VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYUŽITÍ DAT Z POHYBOVÝCH SENZORŮ PRO ANALÝZU ČINNOSTI UŽIVATELE

UTILIZATION OF MOTION SENSOR DATA FOR USER ACTIVITY ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN ERŠEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Eršek Martin**
Program: Informační technologie
Název: **Využití dat z pohybových senzorů pro analýzu činnosti uživatele**
Utilization of Motion Sensor Data for User Activity Analysis
Kategorie: Zpracování signálů

Zadání:

1. Prostudujte základní metody zpracování signálů a porovnání vzorů v datech z pohybových senzorů. Seznamte se s existujícím využitím těchto dat.
2. Navrhněte celkový postup, dílčí metody a aplikaci, která bude v reálném čase snímat, zpracovávat a klasifikovat data z externích pohybových senzorů.
3. Připravte vhodnou datovou sadu a její anotaci tak, aby obsahovala různé relevantní situace.
4. Implementujte navržený systém pomocí dostupných knihoven. Implementujte základní uživatelské funkce aplikace.
5. Vyhodnoťte systém na připravené sadě dat.
6. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.
- Bishop Ch. M. *Pattern Recognition and Machine Learning.* Springer-Verlag New York Inc., ISBN: 0387310738, 2006.
- Dále dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně body 3 a 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cieľom práce je návrh a implementácia klasifikačného algoritmu pre analýzu činnosti používateľa na základe dát z pohybových senzorov. Práca skúma možnosti klasifikácie a počítania opakovaní 7 základných cvikov s vlastnou váhou – kľukov, drepov, podporov na predlaktiach, sed-ľahov, príťahov kolien v sede, tricepsových kľukov na lavičke a výpadov. Dáta z pohybových senzorov sú zbierané mobilným zariadením umiestneným v hornom vrecku nohavíc cvičiaceho používateľa. Výber použitých metód a ich parametrov, ako aj počtu a druhu extrahovaných príznakov bol vykonaný s ohľadom na ich nízku výpočtovú náročnosť. Pri návrhu riešenia bol kladený dôraz na jeho nezávislosť od natočenia zariadenia vo vrecku používateľa. V rámci práce bola vytvorená dátová sada obsahujúca nahrávky 7 tréningov od 4 rôznych používateľov. Navrhovaný postup bol implementovaný v podobne desktopovej aplikácie s CLI rozhraním a následne overený na vytvorenej dátovej sade. Riešenie bolo schopné dosahovať pri analýze a počítaní opakovaní tréningu nevideného používateľa metriku F1-skóre v rozmedzí 45.3 %-74.9 %. Pri nevidenom tréningu od známeho používateľa dosiahlo metriku F1-skóre s hodnotou až 94 %.

Abstract

This bachelor thesis aims to provide a design and implementation of an algorithm for analysis of user activity based on data from motion sensors. The thesis explores possibilities of classification and counting repetitions of 7 basic body-weight exercises, namely: push ups, squats, planks, sit-ups, seated knee raises, tricep dips and lunges. Data from motion sensors are collected by a mobile device located in top pocket of exercising user's trousers. Selection of used methods and their parameters as well as number and type of extracted features is chosen with regard to low computational complexity. When designing a solution, emphasis was put on the fact that it is irrelevant how the device is positioned in the user's pocket. For the thesis, a dataset containing 7 training sessions from 4 different users was created. Designed method was implemented as a desktop application with Command Line Interface and consequently validated on the created dataset. The solution was able to reach metrics of F1-score in range 45.3 %-74.9 % for analysis and counting repetitions of an unseen user's training session. For an unseen training session of a known user, the metrics of F1-score was up to 94 %.

Kľúčové slová

analýza činnosti používateľa, IMU, klasifikácia, počítanie opakovaní, akcelerometer, cvičenie s vlastnou váhou tela, dátová sada, spracovanie signálu, spektrálna analýza

Keywords

human activity recognition, IMU, classification, counting of repetitions, accelerometer, body-weight exercises, dataset, digital signal processing, spectral analysis

Citácia

ERŠEK, Martin. *Využití dat z pohybových senzorů pro analýzu činnosti uživatele*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Využití dat z pohybových senzorů pro analýzu činnosti uživatele

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vítězslava Berana, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Martin Eršek
28. mája 2020

Poďakovanie

Touto cestou by som rád poďakoval môjmu vedúcemu za jeho ochotu počas konzultácií a trefné pripomienky. Ďalej by som rád poďakoval všetkým štyrom kamarátom a jednej kamarátke za ich ochotu a trpezlivosť pri zbieraní pilotných dát, ktoré boli neskôr použité na tréning a následné overenie mnou navrhovaného riešenia. Nie všetky nahrávky boli použité, no každá z nich nezanedbateľným spôsobom pomohla vylepšiť túto prácu a preto som za všetky vďačný. V neposlednom rade by som rád poďakoval svojej rodine a priateľom za ich morálnu podporu.

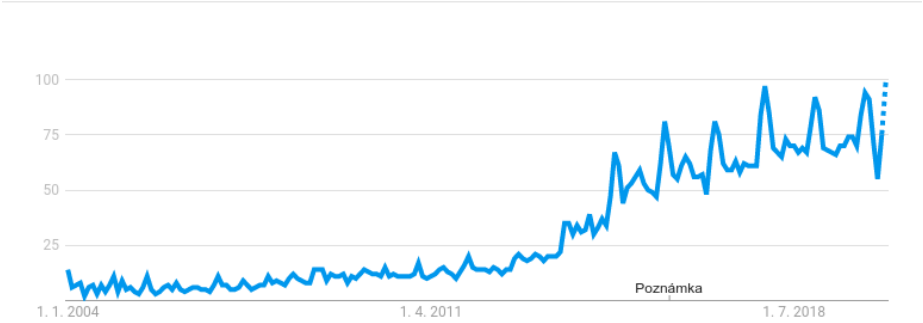
Obsah

Kapitola 1

Úvod

Pohyb a pravidelné cvičenie je nevyhnutnou súčasťou zdravého životného štýlu. S príchodom modernej doby a chytrých zariadení sa zvýšil aj trend sledovania a zaznamenávania si údajov o svojom zdravý viz obrázok 1.1. Spolu s týmto trendom prišli na trh aj zariadenia, ktorých účelom je sledovanie telesnej kondície na základe rôznych metrík – „fitness trackery“. Tieto zariadenia vo forme rôznych náramkov alebo chytrých hodiniek dokážu sledovať niektoré ukazateľe ako tepovú frekvenciu srdca alebo hladinu kyslíku v krvi. Mnohé z nich dokážu automaticky počítať kroky, niektoré dokonca rozpoznať aktivitu používateľa akou je chôdza, beh, plávanie alebo bicyklovanie. Žiadne z komerčne dostupných zariadení (v dobe písania tejto práce) bohužiaľ neponúka možnosť automatického rozpoznávania silových anaeróbnych cvičení a počítania ich opakovaní. Počas môjho prieskumu som našiel iba jedno riešenie, ktoré túto možnosť ponúka – *Exercise Tracker: Wear Fitness*. Táto android aplikácia však k svojej funkčnosti potrebuje chytré hodinky s operačným systémom *Wear OS*. Mojím pozorovaním cvičiacich ľudí na viacerých vonkajších „Street Workout“ ihriskách som zistil, že väčšina z ľudí má pri cvičení svoj smartfón v prednom vrecku ich nohavíc. Prestávky medzi sériami cvičení väčšinou trávia vzájomnou konverzáciou a ich smartfóny tak naďalej ostávajú umiestnené v ich vreckách. Tento fakt spolu s absenciou dostupných riešení bol mojou motiváciou pre túto prácu.

V tejto práci na začiatku zhrniem prehľad existujúcich prác, vedeckých článkov a dátových sád, ktoré sa zaoberajú problematikou rozpoznávania aktivity používateľa na základe dát z pohybových senzorov. Následne navrhnem postup klasifikácie a počítania opakovaní cvikov s vlastnou váhou. Tento postup bude využívať dáta z akcelerometra smartfónu, ktoré budú získané za pomoci mnou navrhnutej a implementovanej android aplikácie. Navrhnutý postup implementujem v podobe desktopovej CLI aplikácie a overím jeho funkčnosť na mnou vytvorenej „pilotnej“ dátovej sade, ktorá vznikla v rámci tejto práce. Zhodnotím výhody a nevýhody mnou navrhovaného riešenia a navrhnem spôsob ako je možné toto riešenie ďalej vylepšiť.



Obr. 1.1: **Trend vyhľadávania slov spojených s fitness tracking-om.** Originálny graf možno nájsť v službe Google Trends po zadaní výrazu „fitness wearable + fitness bracelet + fitness watch + fitness tracking“.

Kapitola 2

Prehľad kľúčových znalostí

Účelom tejto kapitoly je uviesť čitateľa do problematiky rozpoznávania aktivity používateľa a oboznámiť ho so základnými metódami, ktoré budú v tejto práci použité.

2.1 Existujúce práce a metódy riešenia

V tejto podkapitole uvádzam prehľad niekoľkých prác venujúcich sa HAR[1] z ktorých som čerpal. Tieto práce považujem za vhodný zdroj pre štúdium oblasti HAR a štúdium existujúcich postupov pre riešenie rozpoznávania aktivít a počítania opakovaní cvikov s vlastnou váhou. Pri jednotlivých prácach popisujem postup riešenia (mnohokrát aj špecifické implementačné detaily), ktorý zvolili ich autori. Cieľom tohto popisu je vytvorenie prehľadu o možnostiach riešenia na ktoré sa neskôr v práci odkazujem.

V práci **„Push-up Tracking through Smartphone Sensors“** viz [8] sa autori venujú rozpoznávaniu a počítaniu kľukov na základe dát akcelerometru a gyroskopu získaných z mobilného zariadenia umiestneného na ramene cvičiaceho používateľa. Za pomoci kĺzajúceho okna extrahujú príznaky z časovej aj frekvenčnej domény. Tieto príznaky klasifikujú dvoma spôsobmi a to buď použitím SVM[2] alebo MLP[3]. V práci skúmajú vplyv dĺžky kĺzajúceho okna na úspešnosť klasifikácie pri použití oboch klasifikačných metód. Najhorší výsledok úspešnosti klasifikácie bol 97.5 %. Tento výsledok bol dosiahnutý algoritmom MLP s dĺžkou okna pre extrakciu príznakov o veľkosti 1 sekunda. Počítanie opakovaní riešia počítaním vrcholov v signále „hlavnej“[4] osi akcelerometru. Ako príznaky boli zvolené štatistické hodnoty získané z kĺzavých okien pre jednotlivé osi a to konkrétne – priemer, rozptyl, maximum a minimum. Ďalej bola extrahovaná dominantná frekvencia a jej amplitúda z „hlavnej“[5] osi. Práca vychádza z údajov, ktoré boli získané od dvoch cvičiacich subjektov. Autori v práci neriešia nezávislosť ich riešenia od natočenia umiestneného zariadenia.

[1] Human Activity Recognition – Rozpoznávaniu aktivít používateľa.
[2] Support-Vector machine
[3] Multi Layer Perceptron
[4] Ako hlavnú os považujú os s najväčším rozptylom. Anglicky túto os autori označujú ako „main axis“ – nejedná sa teda o použitie PCA a premietnutie do prvého hlavného komponentu.
[5] Viz. vyššie

V práci **„Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data“** viz [12] autori (na rozdiel od väčšiny iných prác v ktorých sa na mobilnom zariadení len zbierajú dáta no ich analýza už prebieha mimo neho) výsledné riešenie implementovali a odskúšali priamo na mobilnom zariadení. Autori skúmajú možnosti klasifikácie 5 typov aktivít v reálnom čase na základe dát získaných z akcelerometra a to – chôdzu, beh, bicyklovanie, riadenie auta a státie na mieste prípadne sedenie s telefónom umiestneným v prednom vrecku používateľa. Sú skúmané dva klasifikačné algoritmy a to k-nn[6] a QDA[7]. Práca pochádza z roku 2012 a už aj s výkonom vtedajších smartfónov sa autorom podarilo dosiahnuť vyťaženie CPU[8] nižšie ako 5 % pri použití klasifikačnej metódy QDA. Pre tréning ich algoritmu boli použité dáta od 8 používateľov. Príznaky boli extrahované na základe 7.5 sekúnd dlhého kĺzajúceho okna. Aby autori predišli vplyvu rotácie zariadenia rozhodli sa túto informáciu o rotácií kompletne odstrániť a to tak, že z 3-dimenzionálnych dát vypočítali iba magnitúdu[9] akcelerácie. Z týchto novo vzniknutých 1-dimenzionálnych dát extrahovali štatistické hodnoty pre jednotlivé okná a to – smerodajnú odchýlku, priemer, minimum a maximum. Ako ďalšie príznaky použili počty vzoriek v určitom percentile a sumu druhých mocnín pozorovaní nad určitý percentil.

V práci **„Counting repetitions of exercises from body worn sensors“** viz [17] autor skúma možnosť klasifikácie a počítania opakovaní 7 typov cvičení v reálnom čase. Práca vychádza z nutnosti znalosti typu cvikov v aktuálne vykonávanej sérii cvikov. Pre každý typ cvičenia je vlastný model, ktorého úlohou je kategorizovať rozoznaný segment cvičenia do dvoch tried – cvičenie alebo voľný pohyb. Dáta z akcelerometru a gyroskopu sú najprv spracované za pomoci metódy VQ[10]. Následne sú za pomoci HMM[11] z dát extrahované segmenty, ktoré by mohli obsahovať opakovanie vykonávaného cviku. Ak je dĺžka týchto segmentov príliš krátka alebo dlhá sú okamžite zahodené. Segmenty sú ďalej podvzorkované z dôvodu šetrenia výpočtového výkonu potrebného na vykonanie DTW[12]. Podvzorkované segmenty sú za pomoci metódy DTW namapované na šablónu opakovania očakávaného typu cviku[13]. Následne sú extrahované príznaky a to – priemer, smerodajná odchýlka, maximum a minimum. Tieto príznaky sú za pomoci MLP binárne klasifikované do triedy cvičenie alebo voľný pohyb. Pokiaľ je segment klasifikovaný ako voľný pohyb je zároveň zvýšene aj počítadlo opakovaní. Autor skúmal vplyv umiestnenia vlastného hardwarového zariadenia na 8 rozličných umiestneniach na tele cvičiaceho používateľa. Práca ďalej kládla dôraz na nízku pamäťovú a výpočtovú náročnosť aby bolo možné tento postup implementovať aj pre jednočipy. Postup bol implementovaný a overený na prototype za použitia vývojovej dosky Arduino Due obsahujúcej 32-bitový ARM procesor. Keďže práca vychádza zo znalosti pevného umiestnenie zariadenia, ktoré zbiera pohybové dáta autor v práci nerieši nezávislosť jeho riešenia od natočenia zariadenia.

[6] k-nearest neighbors
[7] Quadratic Discriminant Analysis
[8] 680Mhz ARM11 processor
[9] $\sqrt{x^2 + y^2 + z^2}$
[10] Vector Quantization
[11] Hidden Markov Models
[12] Dynamic Time Warping
[13] Táto šablóna „ideálneho“ opakovania bola vypočítaná z nameraných dát v dátovej sade, ktorá počas tejto práce vznikla.

V práci **„A Comprehensive Study of Activity Recognition Using Accelerometers“** viz [15] autori skúmajú široké spektrum možností rozpoznávania 69 rôznych činností používateľa od extrakcie príznakov po výber klasifikačného algoritmu. Medzi skúmané činnosti patria dlhodobé aktivity ako chôdza, bicyklovanie, sedenie za počítačom ale aj aktivity ako hranie futbalu, polievanie kvetov či skákanie cez švihadlo. V práci je zhrnutý prehľad o najmodernejších tzv. „state-of-the-art“ metódach HAR za pomoci akcelerometrov. V práci je ďalej skúmaná segmentácia, umiestnenie akcelerometrov, spoľahlivosť ale aj schopnosť generalizácie jednotlivých metód.

V práci **„RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises“** viz [7] autori navrhujú a následne testujú metódu pre automatické rozoznávanie cvičení a počítanie opakovaní bez zásahu používateľa. Táto práca teda narozdiel od práce Michala Šustera, ktorú spomínam vyššie, nevychádza z predpokladu o znalosti aktuálne vykonávaného typu cviku. Práca využíva dáta z akcelerometru a gyroskopu, ktoré sú získavané zariadením umiestneným na zápästí používateľa. Za pomoci kĺzavého okno s dĺžkou 5 sekúnd je extrahovaných až 224 príznakov – 28 pre každú z 8 dimenzií skúmaného signálu. Tieto dimenzie vznikli zo 4 signálov odvodených zvlášť pre akcelerometer a gyroskop. Týmito 4 signálmi sú x-ová os, magnitúda danej vzorky v 3D priestore, projekcia signálu za pomoci analýzy hlavných komponentov do prvého hlavného komponentu, projekcia 2D signálu z osi y a z do prvého hlavného komponentu. Príznaky pozostávajú z príznakov získaných autokoreláciou signálu so sebou samým. Ďalej sú extrahované štatistické a frekvenčné príznaky. Na základe extrahovaných príznakov je dané časové okno za pomoci SVM kategorizované ako cvičenie alebo voľný pohyb. Výsledky takýchto klasifikácií sú agregované do segmentov. Tieto segmenty reprezentujúce série opakovaní určitého cviku. Následne sú segmenty identickým spôsobom avšak za použitia iných príznakov kategorizované sériou SVM klasifikátorov do jednej z kategórií značiacich aký typ cvikov bol vykonávaný. Segment obsahujúci znalosti o type cviku je transformovaný za pomoci PCA do prvého hlavného komponentu. V tomto signále sú spočítané vrcholy reprezentované lokálnymi maximami tohto signálu. Vrcholy sú zoradené podľa ich amplitúdy a postupne označované za opakovania cvikov, pokiaľ je daný vrchol vo vhodnej vzdialenosti od už označeného opakovania cviku. Pre určenie vhodnosti vzdialenosti je využitá znalosť typu cvičenia – je známa minimálna dĺžka trvania jedného opakovania. Ďalej je za pomoci autokorelácie spočítaná odhadovaná perióda jednotlivých opakovaní a táto znalosť je použitá na ďalšie vylučovanie vrcholov, ktoré boli kandidátmi na opakovanie cvičenia. Následne sú vypočítané amplitúdy jednotlivých uznaných vrcholov a nájdená amplitúda reprezentujúca 40. percentil. Všetky vrcholy, ktorých amplitúda je nižšia ako polovica tejto amplitúdy sú vylúčené. Nezávislosť na rotácii je v práci riešená za pomoci redukcie dimenzionality osí[14] y a z pomocou projekcie do prvého hlavného komponentu.

V práci **„Exploratory Data Analysis of Acceleration Signals to Select Light-Weight and Accurate Features for Real-Time Activity Recognition on Smartphones“** viz [5] autori hľadajú výpočtovo a pamäťovo nenáročnú metódu pre rozpoznávanie aktivít za pomoci smartfónov. Vychádzajú z dátovej sady, ktorá obsahuje dáta z akcelerometru reprezentujúce 6 aktivít od 30 používateľov. Ukazuje sa, že tieto aktivity môžu byť generované autoregresívnym procesom a preto sú ako príznaky zvolené auto-regresné koefi-

[14] Na ktoré rotácie okolo zápästia vplýva. Os x je fixná vzhľadom na to, že je umiestnená rovnobežne s predlaktím.

cienty. Porovnaním s inými príznakmi autori zistili, že tieto koeficienty vedú k presnejšej klasifikácií (pri vzorkovaní 20Hz) ako pri použití FFT[15], DCT[16] alebo TD[17] príznakov. Ako klasifikačná metóda je používa neurónová sieť typu MLP[18]. Nezávislosť na rotácií dát bola dosiahnutá použitím metódy KDA[19], ktorou boli extrahované AR príznaky spracované. Funkčnosť tohto riešenia bola overená za pomoci dát z 5 rôznych umiestnení smartfónu na tele používateľa.

V práci **„Activity Recognition Invariant to Sensor Orientation with Wearable Motion Sensors“** viz [16] autori navrhujú dve transformačné metódy na odstránenie efektu rotácie zariadenia za účelom HAR. Tieto metódy sú následne overené na 4 „state-of-the-art“ klasifikátor za použitia 5 voľne dostupných dátových sád. Prvou metódou je heuristická metóda založená na transformácií 3-dimenzionálnych dát do 9-dimenzionálnej reprezentácie, ktorá je nezávislá na prvotnej orientácii. Druhá metóda je založená na použití SVD[20] za účelom nájdenia troch hlavných komponentov do ktorých budú dáta transformované.

V práci **„Recognition and repetition counting for complex physical exercises with deep learning“** viz [13] autori skúmajú použitie hlbokých neurónových sietí, konkrétne konvolučných neurónových sietí za účelom automatického rozpoznávania a počítanie opakovaní 10 cvikov typických pre cvičenie crossfit. Výpočet prebieha offline za použitia grafických kariet. Cieľom práce je dosiahnutie čo najvyššej úspešnosti a nie možnosť implementácie tohto postupu v reálnom čase pre mobilné zariadenia. Dáta boli zbierané za pomoci dvojice[21] chytrých hodiniek, ktorých čas bol vzájomne synchronizovaný. Zbierané boli dáta z akcelerometru a gyroskopu a rotácii zariadenia[22]. Zberu sa zúčastnilo 61 cvičiacich ľudí. Dáta od 54 z nich boli použité na tréning a overenie implementovaného algoritmu. Autorom sa podarilo dosiahnuť úspešnosť až 99.96 % pri použití dát z oboch hodiniek súčasne a 98.91 % úspešnosť pri použití hodiniek umiestnených na zápästí. Dáta z kĺzavého okna boli priamo privádzané na vstup neurónovej sieti, ktorá ich klasifikovalo. Počítanie opakovaní bolo tiež vyriešené za pomoci rovnakej architektúry neurónovej sieti (pre každý typ cvičenia jedna), ktorá bola vytrénovaná na klasifikáciu začiatku opakovania pre určitý cvik. Výsledky tejto siete boli vyhladené algoritmom na to navrhnutým[23]. Týmto spôsobom sa podarilo dosiahnuť 91 % úspešnosť počítania opakovaní.

Uvediem ešte niekoľko voľne dostupných dátových sád.

RealWorld (HAR) 2016
https://sensor.informatik.uni-mannheim.de/#dataset_realworld
Aktivita: chôdza hore/dolu po schodoch, skákanie, ležanie, státie, sedenie, behanie, chôdza
Umiestnenie: Pre každé cvičenie bolo súčastne zaznamenávané zrýchlenie na hrudi, pred-

[15] Fast Fourier transform
[16] Discrete cosine transform
[17] Time Domain – štatistických príznakov z časovej domény ako je priemer, smerodajná odchýlka a pod.
[18] Multi Layer Perceptron
[19] Kernel Discriminant Analysis
[20] Singular value decomposition
[21] Prvé hodinky boli umiestnené na zápästí, druhé nad členkom.
[22] Vypočítané na základe dát z akcelerometra a magnetometra.
[23] Viac v samotnej práci.

laktí, hlave, holeni, stehne, ramene a páse.
Senzory: Akcelerometer, Gyroskop, Magnetometer, GPS, Svetelný senzor, Úroveň hluku

Counting repetitions of exercise using body worn sensors
http://michal.sustr.sk/exercise/
Aktivita: „angličáky“, sed-ľahy, drepy, kľuky, poskoky s odrazom znožmo, zdvíhanie rúk, horné háky
Umiestnenie: zápästie
Senzory: Akcelerometer + Gyroskop

Human Activity Recognition Using Smartphones Data Set
https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
Aktivita: chôdza, chôdza hore schodmi, chôdza dole schodmi, sedenie, státie, ležanie
Umiestnenie: pás
Senzory: Akcelerometer + Gyroskop

2.2 Pohybové senzory

Aktivita používateľa skúmaná v tejto práci bude rozpoznávaná na základe údajov získaných z pohybových senzorov. IMU[24] – inerciálna meracia jednotka je pojem, ktorý označuje elektronické zariadenie, ktoré je schopné zaznamenávať zrýchlenie, uhľovú rýchlosť a niekedy aj orientáciu zariadenia za pomoci akcelerometrov, gyroskopov a niekedy aj magnetometrov. Väčšina smartfónov a mnoho „chytrých“ zariadení takúto meraciu jednotku obsahujú. Takmer všetky zariadenia (nie len smartfóny) s operačným systémom Android obsahujú akcelerometer. Mnohé z nich v dnešnej dobe[25] disponujú aj gyroskopom[26]. Jedným z využití pohybových senzorov je sledovanie pohybu zariadenia akým je napríklad jeho náklon, rotácia, či otrasy.

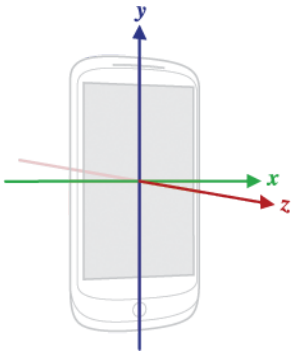
Platforma Android podporuje viaceré hardvérové aj softvérové pohybové senzory. Medzi tieto senzory patria napríklad akcelerometer, gyroskop, senzor počítajúci kroky, senzor merajúci lineárne zrýchlenie[27] a senzor merajúci smer rotácie. Z týchto senzorov sú akcelerometer aj gyroskop vždy implementované za pomoci hardvérového riešenia. Ostatné typy pohybových senzorov môžu byť implementované buď dedikovaným hardvérovým riešením alebo aj softvérovým[28] riešením.

Pohybové senzory sú v platforme Android reprezentované triedou `Sensor` nachádzajúcou sa v „Android sensor framework-u“, ktorý je časťou balíku `android.hardware`. Za pomoci triedy `SensorManager` je možné inštanciovat konkrétny pohybový senzor a zaregistrovať si funkciu typu „event listener“, ktorá bude zachytávať udalosti typu `SensorEvent` produkované konkrétnym pohybovým senzorom. Udalosti typu `SenzorEvent` obsahujú okrem

[24] Z anglického „Inertial Measurement Unit“.
[25] Niektoré staršie zariadenia neobsahujú gyroskop.
[26] Viz https://developer.android.com/guide/topics/sensors/sensors_motion.
[27] Zrýchlenie, ktoré neobsahuje gravitačné zrýchlenie.
[28] V závislosti na konkrétnom senzore softvérové riešenie odvodzuje svoj výstup na základe výstupov hardvérových senzorov akými sú akcelerometer, gyroskop a magnetometer.

iného aj viac-dimenzionálne pole `float` hodnôt reprezentujúce výstupné hodnoty daného senzoru. Tieto udalosti ďalej obsahujú aj časové razítko udávajúce čas vyhotovenia vzorku z daného senzoru v milisekundách. Pri registrácii funkcie typu „even listener“ sa udáva aj parameter `SENSOR_DELAY` určujúci požadované oneskorenie. Toto oneskorenie určuje s akou rýchlosťou by si aplikácia priala dostávať údaje z daného senzoru. Oneskorenie však nie je záväznou hodnotou a Android negarantuje, že bude dodržané. Vplyv na zmenu reálneho oneskorenia môže mať ako samotný systém Android, tak aj iné bežiace aplikácie. Vo vývojárskej príručke sa odporúča toho oneskorenie nastaviť na čo najvyššiu akceptovateľnú hodnotu – najpomalšiu akceptovateľnú rýchlosť akou aplikácia vyžaduje dostávať údaje z daného senzoru. Neexistuje žiadna verejná metóda za pomoci ktorej by bolo možné zistiť reálnu rýchlosť akou `sensor framework` posiela `SenzorEvent` udalosti aplikácii[29]. Táto frekvencia vzorkovania sa však dá odvodiť za pomoci časového razítka nachádzajúceho sa v objektoch typu `SenzorEvent`. Nastavené oneskorenie sa po registrácii „event listener“ funkcie už nedá zmeniť. V prípade potreby zmeny je potrebné túto funkciu odregistrovať a vzápätí znovu zaregistrovať s novou hodnotou parametru `SENSOR_DELAY`. Možné hodnoty tohto parametra sú:

- `SENSOR_DELAY_FASTEST` – najrýchlejší možný mód
- `SENSOR_DELAY_GAME` – oneskorenie 20 000 mikrosekúnd
- `SENSOR_DELAY_UI` – oneskorenie 60 000 mikrosekúnd
- `SENSOR_DELAY_NORMAL` – oneskorenie 200 000 mikrosekúnd, predvolená hodnota



Obr. 2.1: **Súradnicový systém relatívny k pozícii zariadenia používaný jednotlivými senzormi.** Obrázok prevzatý z https://developer.android.com/images/axis_device.png.

Pri vývoji aplikácií využívajúcich dáta z pohybových senzorov pre novšie verzie Androidu treba upozorniť na nasledujúci fakt, citujem a prekladám do slovenčiny z https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-practices:

> Na zariadeniach na ktorých beží Android 9 (API level 28) alebo vyšší, majú aplikácie bežiace na pozadí nasledujúce obmedzenia:

[29] Viz https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-monitor.

- Senzory, ktoré používajú režim neustáleho získavania údajov ako akcelerometre a gyroskopy neprijímajú udalosti.
- Senzory, ktoré používajú režim „pri zmene“ alebo režim jednorázového zachytenia neprijímajú udalosti.

Z citovaného textu vyplýva, že pri novších (>= 9) verziách Android-u je potrebné zabezpečiť aby sa aplikácia po zamknutí obrazovky nedostala do takzvaného „background“ módu. V tomto móde aplikácia nebude dostávať údaje z pohybových senzorov.

Na obrázku 2.1 možno vidieť súradnicový systém relatívny k pozícii zariadenia, ktorý používajú pohybové senzory v platforme Android.

Akcelerometer je v reprezentovaný ako senzor typu `TYPE_ACCELEROMETER`. Jeho výstupom sú hodnoty reprezentujúce zmerané zrýchlenie v jednotlivých osiach zariadenia udávané v jednotke m/s^2.

Gyroskop je reprezentovaný ako senzor typu `TYPE_GYROSCOPE`. Výstupom gyroskopu sú zmerané uhľové rýchlosti udávané v jednotke rad/s. Pri gyroskopoch je potrebné upozorniť na fakt, že aj v dnešnej modernej dobe sa na trhu stále nachádza štatisticky nezanedbateľné množstvo smartfónov (väčšinou takzvaných „low-cost“), ktoré periférium gyroskop neobsahujú.

Ako možno vidieť, akcelerometer a gyroskop sú dve rozdielne periférie zariadenia a preto z nich **nie je možné** získavať údaje v rovnaký časový okamih.

Okrem možnosti vývoja Android aplikácií za pomoci písania natívneho kódu v jazykoch Kotlin/Java, je možné použiť aj iný mobilný framework. Jedným z takýchto frameworkov, ktoré možno použiť pre vývoj Android aplikácií je aj open-source mobilný framework Apache Cordova[30]. Aby bolo v tomto frameworku možné pristupovať k akcelerometru a gyroscopu je potrebné nainštalovať nasledujúce zásuvné moduly:

- `cordova-plugin-device-motion`[31] – prístup k akcelerometru
- `cordova-plugin-device-gyroscope`[32] – prístup k gyroskopu

Inštalácia zásuvných modulov sa vykonáva nasledujúcim spôsobom:
`cordova plugin add cordova-plugin-device-motion`

Tieto zásuvné moduly získavajú údaje z pohybových senzorov za použitia Android-ového „sensors framework-u“ pričom pri volaní metód používajú hodnotu oneskorenia `SENSOR_DELAY_UI`. Hodnotu parametra oneskorenia s ktorou tieto zásuvné moduly volajú metódy Android-ového „sensor framework-u“ bohužiaľ nie je možné zmeniť.

Prístup k akcelerometru je možný za pomoci globálneho objektu `navigator.accelerometer`. Tento objekt podporuje 3 metódy a to:

- `navigator.accelerometer.getCurrentAcceleration`

[30] https://cordova.apache.org/
[31] Viz https://cordova.apache.org/docs/en/7.x/reference/cordova-plugin-device-motion/index.html.
[32] Viz https://github.com/fuegoio/cordova-plugin-device-gyroscope.

- `navigator.accelerometer.watchAcceleration`
- `navigator.accelerometer.clearWatch`

Metóda `navigator.accelerometer.getCurrentAcceleration` vracia akceleráciu získanú v momente jej volania.

Metóda `navigator.accelerometer.watchAcceleration` nastaví callback funkciu `accelerometerSuccess`, ktorá je pravidelne volaná a sú jej predávané údaje získané z akcelerometru. Frekvenciu týchto volaní možno nastaviť za pomoci `accelerometerOptions` objektu. Tento objekt obsahuje kľúč s názvom `frequency`, ktorého hodnota definuje frekvenciu volania callback funkcie s údajmi získanými z akcelerometru. Hodnota tohto parametru je (aj napriek tomu, že sa parameter nazýva `frequency`) čas v milisekundách určujúci periódu volania callback funkcie. Pokiaľ nie je počas periodického volania callback funkcie `accelerometerSuccess` dostupná nová vzorka z akcelerometru, callback funkcia je volaná s dátami reprezentujúcimi poslednú dostupnú hodnotu z akcelerometru – callback funkcia `accelerometerSuccess` teda môže dostávať duplikátne vzorky.

Objekt `Acceleration` predávaný callback funkcii obsahuje štyri kľúče. Kľúče `x`, `y` a `z` obsahujú hodnoty zrýchlenia v jednotlivých osiach. Štvrtým kľúčom je kľúč `timestamp`, ktorého hodnotou je čas vyhotovenia vzorky v milisekundách od epochy[33].

Metóda `navigator.accelerometer.clearWatch` anuluje volanie callback funkcie nastavené volaním metódy `navigator.accelerometer.watchAcceleration`. Jej parametrom je takzvané `watchID`, ktoré bolo získané pri registrácii callback funkcie metódou `navigator.accelerometer.watchAcceleration`.

Prístup ku gyroskopu je možný identickým spôsobom. Používa sa objekt `navigator.gyroscope` a metódy `navigator.gyroscope.watchGyroscope`, `navigator.gyroscope.getCurrentGyroscope`, `navigator.gyroscope.clearWatch`.

2.3 Diskrétna fourierová transformácia

Analyzovaný signál z pohybových senzorov môžno reprezentovať v dvoch doménach, a to v časovej doméne alebo frekvenčnej doméne. Z každej z týchto reprezentácií môžno extrahovať príznaky, ktoré budú následne použité pre klasifikáciu daného signálu. Keďže je signál z pohybových senzorov získavaný v časovej doméne, aby bolo možné extrahovať príznaky z frekvenčnej oblasti je potrebné tento signál transformovať do jeho frekvenčnej domény. Na prevod signálu z jeho časovej domény do jeho frekvenčnej domény slúži Fourierová transformácia. Táto transformácia pracuje so spojitým signálom. Signál získaný z pohybových senzorov je diskrétny v čase a preto bude v práci použitá takzvaná Diskrétna fourierová transformácia – DFT. Táto transformácia pracuje s diskrétnym signálom v čase s konečnou dĺžkou a transformuje ho z jeho časovej domény do frekvenčnej domény.

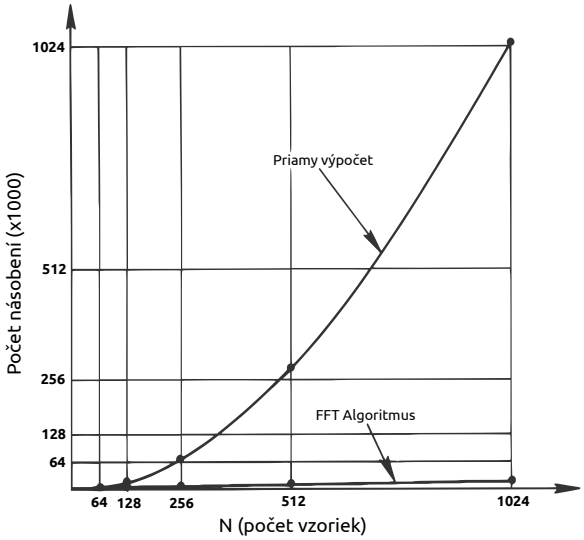
[33] Milisekundový variant UNIX Timestamp-u.

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} \tag{2.1}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{+j2\pi nk/N} \tag{2.2}$$

DFT transformuje postupnosť N komplexných čísiel $x_0, x_1, ..., x_{N-1}$ reprezentujúcich signál v časovej doméne na postupnosť N komplexných čísiel $X_0, X_1, ..., X_{N-1}$ reprezentujúcich signál vo frekvenčnej doméne. Hodnota týchto DFT koeficientov X_k reprezentuje magnitúdu a fázu jednotlivých príslušných frekvencií k, pričom ich počet je rovný počtu vzoriek signálu N. Tieto koeficienty tvoria takzvané „spektrum“ signálu. Postup ich výpočtu možno vidieť vo vzorci 2.1.

Signál možno previesť aj opačným smerom, a to z jeho reprezentácie vo frekvenčnej doméne[34] späť do jeho reprezentácie v časovej doméne. Takúto inverznú transformáciu možno vykonať za pomoci metódy Inverznej Diskrétnej Fourierovej Transformácie – IDFT. Postup tejto transformácie je ilustrovaný vzorcom 2.2.



Obr. 2.2: **Porovnanie počtu násobení počas výpočtu DFT pri použitý priameho výpočtu v porovnaní s použitím algoritmu FFT.** Obrázok prevzatý a preložený z [2].

Zo vzorca 2.1 možno odvodiť aj obtiažnosť výpočtu DFT algoritmu pre N vzoriek. Nech je $N = 2$ potom výpočet jedného DFT koeficientu X_k možno vyjadriť ako:

$$X[k] = \sum_{n=0}^{1} x[n]e^{-j2\pi nk/2} = x[0] \cdot e^{-j\cdot 2\cdot \pi \cdot 0\cdot k/2} + x[1] \cdot e^{-j\cdot 2\cdot \pi \cdot 1\cdot k/2}$$

Pre výpočet jedného DFT koeficientu je teda potrebných N komplexných násobení a $(N-1)$ komplexných sčítaní. Vzhľadom na to, že počet DFT koeficientov je rovný N, obtiažnosť

[34] Signál je reprezentovaný komplexnými DFT koeficientami.

výpočtu DFT teda môžno definovať ako N^2 komplexných násobení a $N \cdot (N-1)$ komplexných sčítaní. Vďaka tejto kvadratickej zložitosti je použitie DFT pre transformáciu signálov väčších dĺžok N výpočtovo neefektívne.

Z tohto dôvodu bol navrhnutý algoritmus rýchlej Fourierovej transformácie – Fast Fourier Transform, FFT. Tento algoritmus je založený na myšlienke rozdelenia výpočtu DFT signálu na viacero častí. Pokiaľ je dĺžka signálu bezozvyšku deliteľná dvomi, je možné tento signál rozdeliť na dve časti a vypočítať zvlášť DFT každej z nich – jeden takýto výpočet bude mať náročnosť $(N/2)^2$ komplexných súčinov a $(N/2)(N/2-1)$ komplexných sčítaní. Dokopy tak bude náročnosť tohto výpočtu $N^2/2$ komplexných súčinov a $N(N/2-1)$ komplexných súčtov. Postup rozdeľovania signálu na menšie časti možno niekoľkokrát zopakovať a výpočet tým ešte viac zefektívniť. Takýmto spôsobom možno celú zložitosť výpočtu zefektívniť až na $O(N \cdot \log_2 N)$. Na obrázku 2.2 možno vidieť porovnanie počtu násobení pri priamom výpočte a pri použití FFT. Existujú dva spôsoby akými je možné rozdeliť postupnosť vzoriek signálu na dve časti a to:

- Prvá časť bude obsahovať vzorky s párnym indexom a druhá časť vzorky s nepárnym indexom. Táto verzia algoritmu sa nazýva FFT algoritmus decimácie v čase.
- Prvá časť signálu bude obsahovať prvú polovicu vzoriek a druhá časť druhú polovicu vzoriek. Táto verzia algoritmu sa nazýva FFT algoritmus decimácie vo frekvencii.

Z dôvodov spomínaných vyššie sa najčastejšie volí dĺžka signálu rovná celočíselnej mocnine čísla 2. Existujú však aj implementácie algoritmu FFT akou je napríklad aj FFTW[35], ktoré dokážu s rovnakou zložitosťou transformovať aj signály inej dĺžky[36].

Podrobnejší popis DFT a FFT možno nájsť v knihách [10] a [2]. Tému spracovania digitálnych signálov a diskrétnej fourierovej transformácie možno nájsť prehľadne spracovanú v slovenskom jazyku aj v bakalárskej práci [4].

2.4 Analýza hlavných komponentov

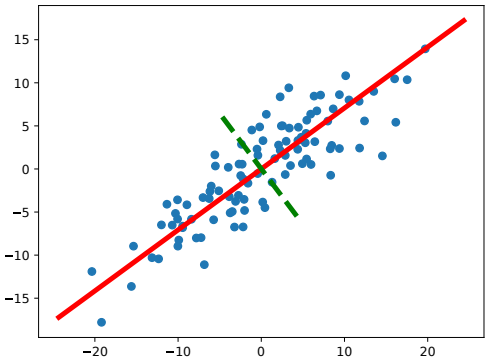
V práci sa spracovávajú rôzne časti signálu reprezentujúceho nahrávku tréningu. Jednou z týchto častí je aj „kalibračný cvik“, ktorý bude použitý na elimináciu vplyvu natočenia smartfónu vo vrecku užívateľa. Bude potrebné nájsť takú transformáciu signálu kalibračného cviku, ktorá z náhodne rotovaného identického signálu zakaždým vytvorí rovnaký signál. Výsledná aplikácia bude okrem klasifikácie cvičenia počítať aj jednotlivé opakovania tohto cvičenia. Aby bolo možné spočítať opakovania cvikov, bude potrebné transformovať 3-dimenzionálny signál konkrétnej cvičebnej série na 1-dimenzionálny signál tak, aby tento 1-dimenzionálny signál reprezentoval najvyššiu mieru rozptylu v pôvodnom 3-dimenzionálnom signále. Pre oba tieto účely je možné použiť Analýzu hlavných komponentov – Principal component analysis, PCA.

Analýza hlavných komponentov je štatistická metóda, ktorej cieľom je prevod množiny N-dimenzionálnych pozorovaní, ktoré môžu byť korelované na 1 až N-dimenzionálne prvky

[35] Viz http://www.fftw.org/.

[36] Signály s dĺžkou, ktorá nie je celočíselnou mocninou čísla 2. Ideálna je dĺžka, ktorej prvočiniteľmi sú malé prvočísla.

inej množiny, ktoré sú vzájomne lineárne nekorelované. K tomuto účelu je použitá ortogonálna transformácia. Pôvodným a hlavným cieľom PCA je redukcia dimenzionality pri čo najmenšej strate informácií (variability). PCA transformuje pôvodné premenné na nové premenné nazývané „hlavné komponenty“, ktoré sú vzájomne lineárne nekorelované a vznikli lineárnou kombináciou pôvodných premenných. Tieto hlavné komponenty sú na seba vzájomne kolmé a zároveň sú zoradené zostupne podľa miery variability pôvodných dát, ktorú vyjadrujú[37]. Hlavné komponenty reprezentujú nový súradnicový systém do ktorého budú pôvodné pozorovania transformované. Prvý hlavný komponent je určený tak, aby reprezentoval smer najvyššej variability pôvodných dát. Druhý komponent je určený tak, aby bol kolmý na prvý komponent a zároveň reprezentoval smer druhej najvyššej variability pôvodných dát. Identickým spôsobom sú získané aj zvyšné hlavné komponenty. Alternatívne je možné definovať prvý hlavný komponent ako priamku, ktorá minimalizuje priemernú kvadratickú vzdialenosť jednotlivých bodov od tejto priamky. Rovnakým spôsobom možno definovať aj ostatné komponenty. Obe tieto definície vedú na rovnaký algoritmus PCA viac v [1]. Na obrázku 2.3 možno vidieť dva hlavné komponenty množiny 2-dimenzionálnych pozorovaní nájdené za pomoci metódy PCA.



Obr. 2.3: **Ukážka hlavných komponentov prislúchajúcich oblaku 2D bodov.** Prvý hlavný komponent je znázornený červenou plnou čiarou, druhý hlavný komponent je znázornený zelenou prerušovanou čiarou.

V prípade premietnutia pozorovaní do všetkých hlavných komponentov nedochádza k strate informácie, ale iba k rotácii pozorovaní okolo ich stredu, ktorý je určený strednou hodnotou týchto pozorovaní. Redukciu dimenzionality pôvodných pozorovaní možno vykonať tak, že sa tieto pozorovania transformujú len do prvých M hlavných komponentov ($M < N$).

Existujú dve rozdielne metódy algebraického riešenia, ktoré sa používajú k hľadaniu hlavných komponentov. Prvou je výpočet kovariančnej matice z originálnych dát a následný rozklad tejto matice na vlastné hodnoty (eigen value decomposition). Druhým a zároveň najpoužívanejším riešením je výpočet singulárneho rozkladu (singular value decomposition) matice obsahujúcej jednotlivé pozorovania. Detailný popis jednotlivých metód riešenia možno nájsť v článku [11]. Popis metódy PCA spolu s ukážkami jej použitia možno nájsť v knižke [1].

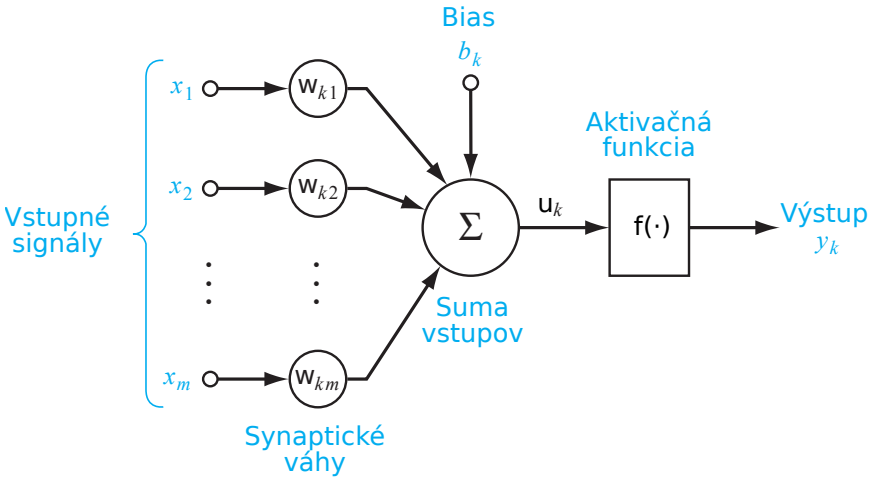
[37] Anglicky sa táto miera označuje ako „explained variance“ – vysvetlená variabilita.

2.5 Multi layer perceptron

Aby bolo možné klasifikovať jednotlivé pohyby na základe extrahovaných príznakov je potrebný vhodný klasifikačný algoritmus. Jedným z takýchto algoritmov je aj Multi layer perceptron – MLP.

Multi layer perceptron je dopredná neurónová sieť, ktorá mapuje vstupný signál na výstupný signál za pomoci skrytých vrstiev[38]. Skryté vrstvy sa skladajú z neurónov typu perceptron viz. [3].

Perceptron je najjednoduchšia forma neurónovej siete pre klasifikáciu vzorov, ktoré sú lineárne separovateľné[39]. Táto sieť pozostáva z jedného neurónu, ktorého vstupom sú synapsie s nastaviteľnou váhou a bias[40]. Algoritmus pre učenie[41] perceptronu nebude popisovaný vzhľadom na to, že v práci je použitá sieť typu MLP, ktorej učiaci algoritmus je odlišný. Algoritmus učenia perceptronu je hlbšie popísaný v [3].



Obr. 2.4: **Nelineárny model neurónu.** Obrázok prevzatý a preložený z [3].

Výpočet váženej sumy u_k vstupných signálov x_j:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.3}$$

Aktivačná funkcia *signum*:

$$f(x) = \begin{cases} 1 & \text{ak x} > 0 \\ 0 & \text{ak x} = 0 \\ \text{-1} & \text{ak x} < 0 \end{cases} \tag{2.4}$$

[38] Jednotlivé vrstvy pozostávajú z neurónov. Medzi sebou sú vzájomne prepojené tak, že výstup každého z neurónov vo vrstve n je pripojený na vstup každého z neurónov vo vrstve n+1.

[39] Vzory, ktoré ležia na opačných stranách nadroviny.
Nadrovina je podpriestor s dimenziou n-1 priestoru s dimenziou n. Napr. v rovine je nadrovinou každá priamka, v trojrozmernom priestore je nadrovinou každá rovina.

[40] Bias je špeciálna hodnota určená pre zvyšovanie alebo znižovanie hodnoty vstupu aktivačnej funkcie.

[41] Hľadanie kombinácie parametrov (váh synapsií a bias-u) siete tak, aby sieť čo najúspešnejšie klasifikovala požadované vzory.

Výstupný signál y_k perceptronu:

$$y_k = f(u_k + b_k) \tag{2.5}$$

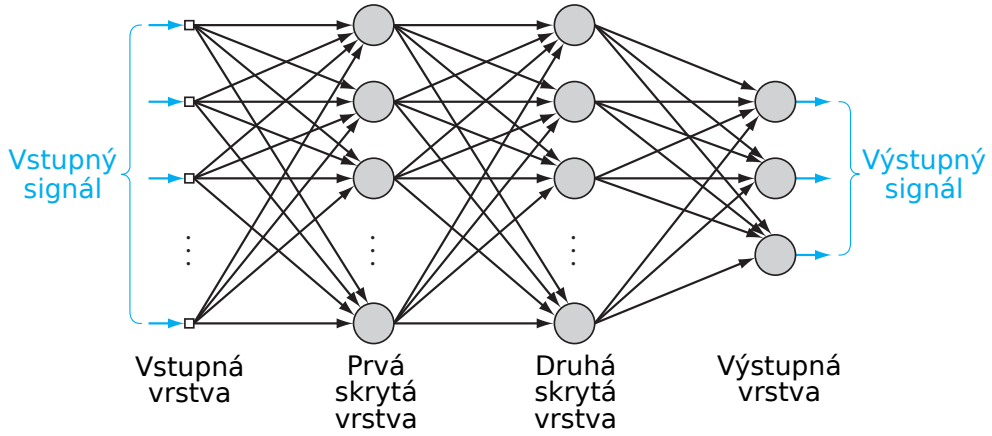
Na obrázku 2.4 možno vidieť všeobecný nelineárny model neurónu s vyznačeným smerom šírenia vstupného signálu. Perceptron[42] spĺňa tento model pričom je ako jeho aktivačná funkcia použitá funkcia *signum* viz rovnica 2.4.

Perceptron spracováva signál tak, že vypočíta vážený súčet u_k jednotlivých vstupných signálov na základe synaptických váh w_{kj} viz 2.3. K tomuto váženému súčtu pripočíta bias b_k a výsledok predá ako argument aktivačnej funkcii f. Výstup tejto aktivačnej funkcie je zároveň aj výstupným signálom y_k perceptronu viz rovnica 2.5.

Perceptron je limitovaný klasifikáciou lineárne separovateľných problémov. Z toho vyplýva, že nedokáže riešiť ani jednoduché lineárne neseparovateľné problémy ako je napríklad logická operácia XOR. Táto vlastnosť bola jedným z hlavných dôvodov, ktoré viedli k návrhu architektúry neurónovej siete typu Multi layer perceptron, ktorá dokáže klasifikovať aj lineárne neseparovateľné problémy.

Základné charakteristiky architektúry MLP:

- Model každého neurónu v sieti obsahuje nelineárnu aktivačnú funkciu, ktorá je diferencovateľná[43].
- Sieť obsahuje aspoň jednu skrytú vrstvu[44].
- Sieť preukazuje vysoký stupeň prepojenosti[45].



Obr. 2.5: **Graf architektúry doprednej neurónovej siete Multi layer perceptron s dvomi skrytými vrstvami.** Obrázok prevzatý a preložený z [3]

Vyššie uvedené charakteristiky sú ale aj zároveň zodpovedné za nedostatky v porozumení toho ako sa táto sieť správa. Teoretická analýza správania sa siete je zložitá vzhľadom na prítomnosť distribuovaných foriem nelinearity a vysokú prepojenosť siete [3]. Z rovnakých

[42] Rosenblatt-ov model percentronu

[43] Funkcia je diferencovateľná pokiaľ existuje derivácia v každom bode domény tejto funkcie. Doména funkcie je množina vstupných argumentov pre ktoré je daná funkcia definovaná.

[44] Skrytá vrstva je taká vrstva, ktorá nie je vstupná ani výstupná.

[45] Miera prepojenosti je odvoditeľná z váh synapsií v sieti.

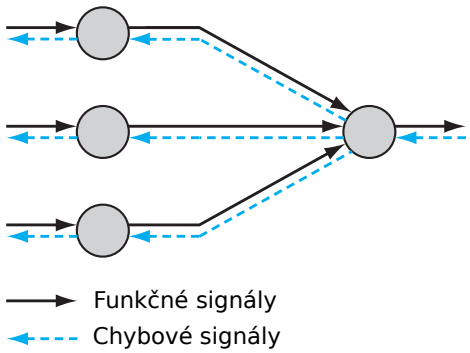
dôvodov je náročná aj vizualizácia procesu učenia sa v siete v porovnaní s vizualízaciou učenia sa perceptronu.

Proces učenia sa siete je náročný, vo svojej podstate sa sieť musí rozhodnúť ako vybrať z obrovského množstva kombinácií nastavení siete (synaptické váhy a bias-y jednotlivých neurónov) také nastavenie, aby sieť čo najpresnejšie klasifikovala požadované vzory.

Populárnym algoritmom pre tréning sietí typu Multi layer perceptron je proces spätného šírenia chyby (back-propagation algorithm). Tento algoritmus pozostáva z dvoch fáz a to:

1. Dopredná fáza - parametre[46] jednotlivých neurónov sú fixné, vstupný signál je propagovaný sieťou vrstvu za vrstvou, neurón za neurónom až kým dosiahne výstupnú vrstvu siete.

2. Spätná fáza - na základe porovnania výstupu siete s požadovaným výstupom[47] je vytvorený chybový signál. Tento signál je ďalej propagovaný sieťou opačným smerom[48] ako funkčný signál. Na základe tohto signálu sa menia parametre neurónom, ktorými tento signál prechádza. Výpočet zmien parametrov je náročnejší ako výpočty potrebné k doprednej propagácií funkčného signálu.



Obr. 2.6: **Ilustrácia smerov šírenia dvoch základných signálov v sieti Multi layer perceptron: Dopredná propagácia funkčných signálov a spätná propagácia chybových signálov.** Obrázok prevzatý a preložený z [3].

V sieti Multi layer perceptron možno identifikovať dva typy signálov (ilustrované na obrázku 2.6) prislúchajúce jednotlivým fázam procesu spätného šírenia chyby a to:

1. **Funkčné signály.** Funkčný signál je vstupný signál, ktorý je dopredne šírený od vstupnej vrstvy, vrstvu za vrstvou (neurón za neurónom) až do výstupnej vrstvy siete. Signál je spracovávaný neurónmi, ktorého menia[49] a tento zmenený signál ďalej propagujú.

2. **Chybové signály.** Chybový signál začína na výstupe sieti a spätne sa šíri na opačným smerom na jej vstup.

[46] Synaptické váhy a bias

[47] Požadovaný výstup je taký signál, aký by mala ideálne sieť produkovať pre konkrétny vstupný signál po skončení tréningu.

[48] Z výstupu na vstup

[49] Vysvetlené vyššie pri popise perceptronu.

Každý neurón, okrem neurónov vstupnej vrstvy vykonáva dve funkcie:

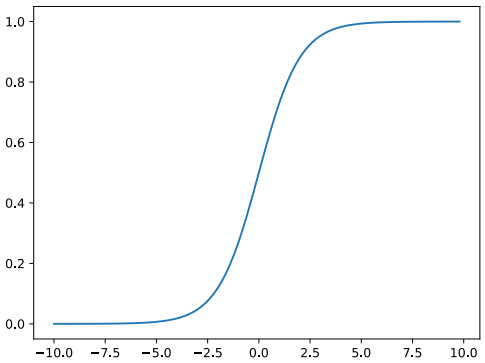
1. Výpočet funkčného signálu - výstupného signálu neurónu na základe jeho vstupov a bias-u.

2. výpočet odhadu gradient vektoru, ktorý je potrebný pre spätnú propagáciu sieťou [2]

Podrobnejší popis algoritmu spätného šírenia chyby a funkcie neurónovej siete Multi layer perceptron možno nájsť v literatúre [3, 1] a mnohých iných zdrojoch a preto v tejto práci nebude uvedený.

Jednou z často používaných nelineárnych aktivačných funkcií pre neuróny[50] v sieti MLP je aj funkcia logistickej sigmoidy. Jej predpis možno vidieť v 2.6 a graf jej priebehu na obrázku 2.7.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$



Obr. 2.7: **Graf priebehu funkcie logistickej sigmoidy.**

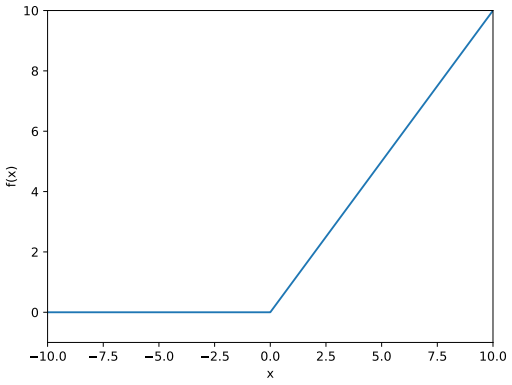
Ďalšou a zároveň jednou z najčastejšie používaných nelinenárnych aktivačných funkcií pre neuróny[51] v sieti MLP je funkcia ReLU[52] – Rektifikovaná lineárna jednotka. Jej predpis možno vidieť v rovnici 2.7 a graf jej priebehu na obrázku 2.8. Táto funkcia je rádovo výpočtovo menej náročná ako funkcia logistickej sigmoidy ako možno vidieť už zo samotného predpisu tejto funkcie. Ďalšou výhodou jej použitia v porovnaní s funkciou logistickej sigmoidy je eliminácia problému miznúceho gradientu[53]. Objasnenie tohto problému a jeho dôsledkov možno nájsť v knihe [3]. Jednou z nevýhod použitia tejto aktivačnej funkcie je fakt, že v sieťach, ktoré túto funkciu používajú prichádza k problému pretrénovania[54] častejšie ako v sieťach používajúcich aktivačnú funkciu logistickej sigmoidy.

Problém pretrénovania spočíva v tom, že sa sieť naučí takmer dokonale rozpoznávať trénovacie dáta, no vďaka tomu stratí schopnosť generalizácie. Dôsledkom je zlyhanie siete pri rozpoznávaní nevidených dát. Jedným z možných riešení ako tomuto problému čiastočne

[50] Najmä pre neuróny umiestnené vo výstupnej vrstve siete MLP viz [9].
[51] Najmä pre neuróny umiestnené v skrytých vrstvách siete MLP viz [9].
[52] Z anglického Rectified Linear Unit
[53] Anglicky „Vanishing gradient problem“.
[54] Anglicky „over-fitting“.

predchádzať je použitie takzvanej „Dropout“ vrstvy. Viac informácií o tejto technike možno nájsť v článku [14].

$$f(x) = \max(x, 0) \tag{2.7}$$



Obr. 2.8: **Graf priebehu funkcie ReLU.**

Kapitola 3

Predspracovanie pohybových dát a ich analýza

Cieľom tejto kapitoly je analyzovať problém automatickej analýzy tréningu s vlastnou váhou a následne navrhnúť postup, ktorým bude možné tento problém vyriešiť. Kapitolu začnem popisom skúmaného problému. Ďalej nadviažem návrhom celkového postupu riešenia problému a následne navrhnem jednotlivé čiastkové kroky, ktoré z tohto celkového postupu vyplývajú.

3.1 Popis problému

Ľudia sa počas svojich životov bežne venujú rôznym aktivitám. Mnohí z nich sa v týchto aktivitách prirodzene snažia zlepšovať. Proces získavania zručností prebieha pomaly a postupne. Jednou z možností ako tento proces zefektívniť a získať nad ním väčšiu kontrolu je aj zavedenie kvalitnej spätnej väzby. Takouto spätnou väzbou je aj sledovanie a zaznamenávanie si napredovania vo vykonávaní konkrétnej aktivity. Pokiaľ je cieľom človeka zlepšiť svoju silu pomocou cvičenia s vlastnou váhou, môžme tento problém špecifikovať ako sledovanie a zaznamenávanie si priebehu vykonaných tréningov. Na základe týchto údajov je daný človek schopný vyhodnotiť či a ako sa v priebehu času mení jeho sila a kondícia. Takéto zaznamenávanie a spracovávanie záznamov o jednotlivých tréningoch však môže byť náročným manuálnym procesom. Cvičiaci človek si tak musí pri každej sérii cvičení počítať opakovania a vzápätí si tento počet niekam poznačiť. Mnoho ľudí cvičí na vonkajších ihriskách v rušnom prostredí. Tento faktor spolu s únavou spôsobenou cvičením vedie k tomu, že mnoho ľudí vďaka náročnosti tohto procesu svoje tréningy nesleduje. Prichádzajú tým však o dôležitú a cennú spätnú väzbu o procese ich (ne)napredovania.

Na trhu sa nachádza mnoho riešení pre automatickú analýzu rôznych aktivít. Bohužiaľ sa takmer žiadne z týchto riešení nevenuje automatickej analýze a počítaniu opakovaní cvičení s vlastnou váhou tela. V aplikačných obchodoch rôznych mobilných platforiem možno nájsť mnoho aplikácií pre manuálne počítanie opakovaní. Manuálne počítanie za pomoci týchto aplikácií znamená, že po vykonaní každého opakovania musí používateľ túto činnosť ručným kliknutím v aplikácii zaznamenať. Niektoré aplikácie dokonca prišli s inovatívnym spôsobom počítania opakovaní kľukov. Zakaždým keď používateľ počas vykonávania kľuku priblíži svoje telo k zemi, tak sa zároveň dotkne špičkou nosu dotykového displeja telefónu

a aplikácia toto opakovanie zaznamená. Tieto riešenia nie sú schopné automaticky počítať opakovania nie to ešte rozpoznávať typ vykonávaného cvičenia. Android-ový aplikačný obchod ponúka iba jedno riešenie, ktoré tento problém adresuje – *Exercise Tracker: Wear Fitness*. Toto riešenie však k svojej funkčnosti vyžaduje chytré hodinky s operačným systémom *Wear OS*. Nákup týchto drahých hodiniek len pre účelom sledovania tréningov môže väčšinu cvičiacich ľudí od tohto riešenia odradiť. Pritom väčšina z týchto ľudí má počas celého tréningu svoj smartfón nehybne vo vrecku. Prečo túto skutočnosť nevyužiť?

Cieľom tejto práce je navrhnúť postup za pomoci ktorého bude možné spomínaný problém vyriešiť s využitím smartfónu umiestneného vo vrecku cvičiaceho používateľa. Riešeným problémom je sledovanie a zaznamenávanie priebehu tréningu – automatické rozpoznávanie typu vykonávaných cvičení a následné počítanie vykonaných opakovaní. Pre vyriešenie tohto problému je potrebné navrhnúť aplikáciu, ktorá bude tento tréning zaznamenávať ako aj následný postup, ktorým bude tento tréning automaticky spracovaný a vyhodnotený. Táto práca si ďalej kladie tieto otázky:

- Je možné použiť dáta z akcelerometru získané zo smartfónu, ktorý je umiestnený vo vrecku používateľa k plne automatizovanej analýze tréningu s vlastnou váhou? (Analýza zahŕňa klasifikáciu aj počítanie opakovaní týchto cvičení.)
- Je možné toto riešenie implementovať tak, aby nebola potrebná znalosť umiestnenia (rotácie) zariadenia vo vrecku používateľa?

3.2 Analýza úlohy a návrh jednotlivých častí riešenia

K realizácii riešenia, ktoré bude spĺňať požiadavky popísané vyššie je potrebné navrhnúť spôsob akým sa budú dáta z pohybových senzorov získavať, ukladať, spracovávať, klasifikovať a počítať opakovania jednotlivých cvikov.

Jednou z kľúčových úloh je návrh zberu dát a následné vytvorenie dátovej sady s ktorou budem neskôr pracovať. Zmyslom tejto úlohy je získať dáta (spolu s ich anotáciou) reprezentujúce jednotlivé činnosti používateľa, ktoré budú cieľom mojej analýzy. Na zozbieraných dátach bude možné mnou navrhovaný a implementovaný model natrénovať a následne vyhodnotiť jeho úspešnosť. Vyhodnotením modelu na nevidených dátach bude možné odhaliť a adresovať jeho nedostatky, ktoré bude možné ďalej vylepšiť. Vo finálnom riešení určenom pre klasifikáciu a počítanie opakovaní budem využívať iba dáta z akcelerometru. Tieto dáta sa ukazujú na základe viacerých štúdií viz. [12],[15], a [5] ako dostatočné na rozoznávanie typu vykonávanej aktivity. Zároveň sa týmto prístupom redukuje výpočtová a pamäťová náročnosť výsledného algoritmu ako aj potreba párovania[1] údajov z týchto periférií. Aplikácia pre zber dát však bude zaznamenávať aj dáta z gyroskopu a výsledná dátová sada bude obsahovať aj tieto údaje.

Pre splnenie tejto úlohy je potrebné aby som navrhol a implementoval aplikácie, ktoré budú slúžiť na zber a anotáciu dát získaných z cieľového zariadenia obsahujúceho pohybové senzory. Z tejto úlohy ďalej vyplýva potreba navrhnúť vhodný formát pre ukladanie takto získaných dát.

[1] Ako popisujem v sekcii 2.2 akcelerometer a gyroskop sú rozdiélne zariadenia a údaje z nich nedostáva aplikácia v rovnaký časový moment.

Aby bolo možné s takto pozbieranými dátami ďalej pracovať, je žiadúce aby som navrhol a implementoval nástroje pre prácu s dátami a ich vizualizáciu.

Finálne riešenie určené pre klasifikáciu typu vykonávaného cvičenia a počítanie opakovaní možno rozdeliť do niekoľkých fáz:

1. Zber dát z pohybových senzorov
2. Predspracovanie dát
3. Extrakcia príznakov
4. Klasifikácia
5. Agregácia výsledkov a segmentácia
6. Počítanie opakovaní
7. Vizualizácia výsledkov modelu

Dáta pochádzajúce z pohybových senzorov nemusia mať rovnomerné vzorkovanie, môžu obsahovať duplikáty a šum. Záznamy jednotlivých tréningov sú rôzne otočené[2]. Z tohto dôvodu je potrené získané dáta vhodne predspracovať.

Z predspracovaných dát sa extrahujú vhodné príznaky tak, aby niesli dostatok informácií potrebný pre klasifikáciu prebiehajúcej činnosti. Takto extrahované príznaky sa za pomoci vhodného klasifikačného algoritmu klasifikujú. Výsledkom tejto klasifikácie bude informácia o triede[3] do ktorej tieto dáta s najvyššou pravdepodobnosťou patria.

Výsledky klasifikácie budú použité na ohraničenie úsekov obsahujúcich opakovania jedného typu cvikov – takzvanú tréningovú sériu. Takto vzniknutý segment obsahujúci sériu cvikov bude použitý ako vstup pre algoritmus počítania opakovaní. Výsledkom tohto algoritmu je počet opakovaní spolu s ich umiestnením v rámci vstupného segmentu.

Nájdené segmenty spolu s označením jednotlivých opakovaní budú vizualizované v grafe zobrazujúcom priebeh tréningu vo všetkých 3 osiach akcelerometru. Užívateľovi sa zobrazí chronologický zoznam vykonaných sérií[4] spolu s agregovanou štatistikou celého tréningu – celkový počet[5] vykonaných opakovaní pre jednotlivé cviky.

3.3 Formát dátovej sady a anotačný systém

Dáta z pohybových senzorov majú charakter časovej postupnosti, pričom každý člen tejto časovej postupnosti sa skladá z hodnôt rovnakého typu a to zrýchlení a uhľových rýchlostí pre jednotlivé osy a času kedy bola táto dátová vzorka zaznamenaná.

Z vyššie spomínaných vlastností vyplýva, že pre reprezentáciu týchto dát je vhodné použiť formát tabuľky. Aby bolo možné s týmito dátami jednoducho pracovať, pre ich reprezentáciu

[2] Dáta boli získané z mobilného zariadenia, ktoré bolo uložené v hornom vrecku nohavíc cvičiaceho používateľa. Toto zariadenie bolo vložené do vrecka ľubovoľne rotované.

[3] Type vykonávaného cviku

[4] Typ cvikov v sérii a ich zistení počet.

[5] Pri podpore na predlaktiach celkový čas v sekundách.

vo forme tabuľky volím formát CSV[6]. Vo formáte CSV použijem ako oddeľovač bodkočiarku „ ; “ miesto štandardného oddeľovača – čiarky „ , “. Dôvod tejto zmeny je objasnený nižšie.

Prvý riadok tohto CSV súboru bude obsahovať názvy jednotlivých stĺpcov tabuľky. Ďalšie riadky budú obsahovať už samotné namerané údaje. Ukážka:

```
ax;ay;az;at;time;label
0.3;4.5;10.2;11.2;112;112;-
```

Zo získaných dát z akcelerometru a gyroskopu budú uložené ich jednotlivé osi – ax, ay, az pre akcelerometer a gx, gy, gz pre gyroskop. Ako aj magnitúdy[7] zrýchlení (at) a uhlových rýchlostí (gt).

Vzhľadom na to, že budú jednotlivé nahrávky obsahovať niekoľko rôznych aktivít používateľa, bude potrebné vedieť tieto aktivity od seba vhodne oddeliť. Z tohto dôvodu je potrebné navrhnúť systém akým budú dané dáta anotované.

Vhodný anotačný systém musí spĺňať niekoľko požiadaviek:

- musí vedieť jednoznačne označiť začiatok a koniec anotovanej aktivity
- podporovať vnorovanie sa anotácií – dátová vzorka môže patriť viacerým anotáciám
- podporovať vzájomné prekrývanie sa anotácií

Na základe týchto požiadaviek som navrhol jednoduchý anotačný systém. Dátové vzorky v dátovej sade budú okrem nameraných hodnôt veličín a času obsahovať aj stĺpec *label*, ktorý bude obsahovať anotačný reťazec.

Anotačný reťazec reprezentuje ľubovoľný[8] počet anotácií. Anotácie sú tvorené anotačnými značkami. Tieto značky môžu byť dvoch typov.

Typy značiek a ich definícia za pomoci regulárneho výrazu[9]:

- značka symbolizujúca začiatok anotácie
  Regulárny výraz: `([a-zA-Z]{1}[0-9a-zA-Z_]*)`
- značka symbolizujúca koniec anotácie
  Regulárny výraz: `(\/([a-zA-Z]{1}[0-9a-zA-Z_]*)`

Pokiaľ anotačný reťazec reprezentuje práve 0 anotácií, je tvorený pomlčkou „ - “. V prípade že anotačný reťazec reprezentuje aspoň jednu anotáciu tak je tvorený zoznamov anotačných značiek, ktoré sú od seba oddelené znakom čiarka[10] „ , “ pričom za poslednou anotačnou značkou sa už čiarka nenachádza. V prípade, že bola anotácia ukončená ukončovacou anotačnou značkou, môže sa neskôr znovu opakovať za použitia začiatočnej anotačnej značky.

[6] CSV – Comma separated values viz. https://en.wikipedia.org/wiki/Comma-separated_values

[7] $magnituda(x, y, z) = \sqrt{x^2 + y^2 + z^2}$

[8] Ľubovoľný – 0 až N

[9] https://en.wikipedia.org/wiki/Regular_expression

[10] Keďže je čiarka štandardným oddeľovačom vo formáte CSV, použijeme formát CSV s oddeľovačom bodkočiarka viz. vyššie.

Ukážka[11] formátu dátovej sady:

```
ax;ay;az;at;time;label
1;1;1;1;20;-
1;1;1;1;21;plank_rep1
1;1;1;1;22;-
1;1;1;1;25;/plank_rep1
1;1;1;1;40;
1;1;1;1;47;test
1;1;1;1;51;abc
1;1;1;1;72;-
1;1;1;1;74;/test
1;1;1;1;81;/abc
1;1;1;1;87;pushups
1;1;1;1;89;pushups_rep1
1;1;1;1;93;-
1;1;1;1;103;/pushups_rep1,/pushups
1;1;1;1;111;-
```

Jednotlivé cviky budú anotované názvom cviku nasledovaným reťazcom `_repX`, kde `X` značí poradové[12] číslo opakovania daného cviku. Série cvikov (napríklad séria 20 kľukov) budu anotované názvom cviku napríklad `pushups`. Pokiaľ sa v tréningu vyskytuje viacero sérií rovnakého cviku budú anotované taktiež **len** názvom cviku – anotácia názov cviku teda nie je jedinečná v rámci jedného tréningu.

3.4 Spôsob zberu dát

Cieľom práce je vytvoriť program slúžiaci na analýzu činnosti používateľa na základe dát z pohybových senzorov. Aby bolo možné tento cieľ realizovať, je potrebné vytvoriť vhodnú dátovú sadu. Táto dátová sada bude obsahovať relevantné situácie – nahrávky aktivít, ktoré majú byť analyzované spolu s nahrávkami iných činností.

Aby bolo možné dáta použiť za účelom analýzy činnosti používateľa, musia byť tieto dáta zbierané rovnakým spôsobom a z rovnakých zariadení ako budú zbierané vo finálnom riešení, ktorého účelom bude ich analýza. Rozhodol som sa, že cieľovým zariadením pre zber dát bude smartfón s operačným systémom Android. Z tohto dôvodu je pre zber dát potrebné vyvinúť mobilnú aplikáciu pre operačný systém Android.

Väčšina moderných mobilných zariadení disponuje niekoľkými pohybovými senzormi a to konkrétne akcelerometrom a gyroskopom. Z tohto dôvodu bude aplikácia pre zber dát zbierať práve dáta z akcelerometru a gyroskopu mobilného zariadenia, ktoré bude umiestnené vo vrecku cvičiaceho používateľa. Tieto dáta bude ukladať vo formáte CSV do vnútornej pamäti tohto zariadenia. Keďže akcelerometer a gyroskop sú dve rozdielne periférie mobilného zariadenia, dáta z nich sa nedajú získavať z identického časového momentu. To je

[11] Text slúži ako názorná ukážka použitia navrhnutého anotačného systému, pričom obsahuje aj vnorené a prekrývajúce sa anotácie. Text ukážky neobsahuje reálne namerané dáta.

[12] Poradové čísla začínajú od 1. Zároveň platí že sú jedinečné pre celý anotovaný súbor tj. určujú absolutné poradie cviku v rámci celého tréningu, nie jeho poradie v rámci série rovnakých cvikov.

dôvod prečo budú dáta z akcelerometru a gyroskopu ukladané oddelene každé do vlastného súboru. Pre každú dátovú vzorku sa bude ukladať čas jej vyhotovenia uvádzaní v uplynutých milisekundách od začiatku nahrávania. Navrhujem použiť vzorkovaciu frekvenciu 10Hz. Vzhľadom na charakter vykonávaných aktivít skúmaných v tejto práci, by mala byť táto frekvencia dostatočná pre účel ich klasifikácie.

Cvičiaci človek bude počas tréningu vykonávať jednotlivé požadované cviky – aktivity, ktoré majú byť analyzované. Zozbierané dáta v jeho zariadení však nebudú anotované. Preto je potrebné vyvínúť druhú mobilnú aplikáciu, ktorá bude slúžiť na anotáciu dát. Je potrebné, aby si tieto dve aplikácie na začiatku medzi sebou synchronizovali čas, respektíve aby obe začali nahrávať v rovnaký okamih. Zatiaľ čo používateľ cvičí, pozerá sa na neho druhý človek s otvorenou anotačnou aplikáciou na svojom zariadení. Za pomoci tejto aplikácie anotuje činnosť cvičiaceho používateľa. Jednotlivé anotácie budú ukladané do príslušného súboru spolu s časom ich vyhotovenia reprezentovaným ako uplynutým časom v milisekundách od začiatku nahrávania. Viac o formáte dát v sekcii 3.3.

Človek anotujúci tréning bude pomocou aplikácie označovať aktuálne vykonávanú aktivitu tak, aby boli zaznamenané jednotlivé série cvikov ako aj jednotlivé opakovania týchto cvikov v rámci danej série.

Aby boli výstupné súbory reprezentujúce dáta z jednotlivých pohybových senzorov a anotácie medzi sebou jednoducho rozlíšiteľné navrhujem nasledujúci formát zápisu názvov súborov.

Názov tréningu `Miroslav_pondelok`:

- `Miroslav_pondelok_a_.csv` - súbor obsahujúci dáta z akcelerometru
- `Miroslav_pondelok_g_.csv` - súbor obsahujúci dáta z gyroskopu
- `Miroslav_pondelok_annot_.csv` - súbor obsahujúci anotácie jednotlivých aktivít

K nameraným údajom z výstupných súborov z akcelerometru a gyroskopu je potrebné následne priradiť zaznamenané anotácie. Týmto spôsobom vzniknú súbory, ktoré budú okrem údajov z jednotlivých pohybových senzorov obsahovať aj jednotlivé anotácie priradené do stĺpca `label`.

3.5 Predspracovanie dát

Dáta získané z akcelerometra[13] obsahujú duplikáty a šum. Duplikáty aj šum sú nežiadúce keďže sa v dátach vyskytujú náhodne a majú priamy vplyv na extrahované príznaky. Z tohto dôvodu je potrebné ich eliminovať.

Na odstránenie duplikátov navrhujem nasledovný algoritmus viz Algoritmus 1.

Za účelom odstránenia šumu a vyhladenia dát je vhodné použiť dolno-priepustný filter s vhodnou medznou frekvenciou. Pri výbere frekvencie treba mať na pamäti, že filtrácia

[13]Mobilná aplikácia zbiera aj dáta z gyroskopu no pre účel finálneho riešenia som sa rozhodol použiť iba dáta z akcelerometru. Viac v sekcii 3.2.

Algoritmus 1: RemoveDuplicates

Input: ax, ay, az, time
Output: out_x, out_y, out_z, out_time

```
1:  out_x = [ax[0]]
2:  out_y = [ay[0]]
3:  out_z = [az[0]]
4:  out_time = [time[0]]
5:  last_time = [time[0]]
6:  rec_no = 0
7:  for t in time do
8:      if t == last_time then
9:          rec_no += 1
10:         continue
11:     end if
12:     out_x.append(ax[rec_no])
13:     out_y.append(ay[rec_no])
14:     out_z.append(az[rec_no])
15:     out_time.append(time[rec_no])
16:     last_time = t
17:     rec_no += 1
18: end for
```

dát týmto filtrom priamo ovplyvní frekvenčnú doménu signálu[14]. Navrhujem dáta filtrovať dolno-priepustným filtrom prvého rádu s medznou frekvenciou 4Hz. Túto frekvenciu navrhujem vzhľadom na veľkosť kĺzajúceho okno a jeho trvanie, ktoré navrhujem nižšie v sekcii 3.6. Po implementácii algoritmu sa vyhodnotí vplyv tohto parametra modelu a frekvencia bude prípadne upravená.

Telefón s nahrávacou aplikáciou bol vo vrecku používateľa náhodne otočený. Vzhľadom na to, že natočenie dát má priamy vplyv na extrahované príznaky, je potrebné navrhnúť spôsob ako tento vplyv obmedziť. Naskytujú sa 2 riešenia:

- Rotáciu neeliminovať → potreba nájsť príznaky na ktoré táto rotácia nemá vplyv
- Rotáciu eliminovať vhodnou transformáciou

Transformácia, ktorá má eliminovať vplyv rotácie musí spĺňať jednu požiadavku a to, že z rovnakého signálu, ktorý je náhodne rotovaný po transformácii vždy vznikne rovnaký výstupný signál.

Najjednoduchšia transformácia spĺňajúca túto požiadavku je prevod 3-dimenzionálneho signálu na 1-dimenzionálny signál reprezentujúci magnitúdu zrýchlenia. Týmto spôsobom sa kompletne eliminuje informácia o smere zrýchlenia. Samotná magnitúda zrýchlenia však nie je dostatočná pre rozpoznanie jednotlivých cvičení s vlastnou váhou. Tento fakt je možné aj vizuálne overiť za pomoci mnou navrhovaného nástroja `spectrogram`. Zatiaľ čo človek

[14]V prípade extrakcie príznakov z frekvenčnej domény ich táto zmena priamo ovplyvní. Filtrácia ovplyvňuje aj časovú doménu signálu no nie tak výrazne ako ovplyvňuje doménu frekvenčnú

dokáže vizuálne jasne identifikovať jednotlivé série cvičení vo vizualizácií 3-dimenzionálneho signálu z tréningu. Vizualizácia magnitúdy signálu mu pripomína skôr šum. V sekcii 2.1 uvádzam aj prácu [12] v ktorej autori použili iba magnitúda zrýchlenia. Táto práca sa však venuje rozpoznávaniu dlhodobých aktivít, ktoré sa medzi sebou líšia v magnitúde zrýchlenia a je ich preto možné na základe magnitúdy rozpoznať.

Ďalšou z možných transformácií, ktoré spĺňajú tento predpoklad je aj transformácia signálu do jeho hlavných komponentov získaných metódou PCA. Výsledkom premietnutia náhodne rotovaného identického signálu do jeho hlavných komponent je vždy rovnaký výstupný signál. Akákoľvek rotácia signálu rotuje rovnakým spôsobom aj jeho hlavné komponenty. Vychádzajú z tejto vlastnosti navrhujem nasledovný postup rotácie signálu. Z tréningu sa extrahuje signál zodpovedajúci opakovaniu/opakovaniam kalibračného cviku. Nad týmto signálom sa za pomoci metódy PCA vypočíta transformácia do všetkých[15] z jeho hlavných komponent. Touto transformáciou sa transformuje signál celého tréningu. Za predpokladu, že je kalibračný cvik v rôznych tréningoch takmer identický, bude táto transformácia rotovať náhodne otočené tréningy vždy na rovnaký súradnicový systém. Ako kalibračný cvik navrhujem použitie cviku drep. Tento cvik je vykonávaný v hojnom počte počas tréningov a zároveň ho zvládnu vykonať aj jedinci so slabšou fyzickou kondíciou. Cvik zo svojej podstaty zasahuje pri akejkoľvek rotácii zariadenia do viacerých osí. Táto vlastnosť je žiadúca pre správnu funkciu eliminácie vplyvu rotácie. Takmer identický spôsob riešenia problému s rotáciou navrhujú aj autori v práci [16], konkrétne v druhej nimi navrhovanej transformácii. Hlavné komponenty signálu získavajú metódou SVD[16] a následne signál transformujú do týchto komponent. Rozdiel v nimi navrhovanom riešení a mnou navrhovanom riešení je použitie kalibračného cviku. Pokiaľ by bol náhodne rotovaný rovnaký signál dá sa použiť aj metóda navrhovaná v ich práci. Pokiaľ sú však náhodne rotované signály tréningov, ktoré obsahujú iné série a typy cvikov, je použitie kalibračného cviku nevyhnutné.

3.6 Extrakcia príznakov

Predspracovaný signál je časovou postupnosťou údajov o zrýchleniach v jednotlivých osiach. Aby bolo možné na základe tohto signálu klasifikovať aktivitu používateľa je potrebné získať časové intervaly, ktoré tento signál rozdeľujú medzi jednotlivé aktivity a voľný pohyb. Celý signál teda nemožno analyzovať naraz, ale je potrebné ho analyzovať po menších úsekoch. Dĺžka týchto úsekov závisí od dĺžky aktivít, ktoré majú byť analyzované. V prípade počítania opakovaní a rozpoznávania typu vykonávaných cvikov by teda mala byť menšia ako dĺžka najkratšieho možného opakovania cviku. Signál bude analyzovaný za pomoci takzvaného kĺzajúceho sa okna. Signál je orezaný na veľkosť tohto okna. Z tohto orezaného signálu sú následne extrahované príznaky. Aby bolo možné lepšie podchytiť klasifikáciu jednotlivých opakovaní je vhodné aby sa tieto kĺzajúce okná pri posune čiastočne prekrývali. Vzhľadom na charakter rozpoznávanej aktivity preto navrhujem kĺzajúce okno s dĺžkou 10 vzoriek[17]. Ďalej navrhujem aby sa tieto okná prekrývali na 80 % ich dĺžky. Kĺzajúce okno sa teda bude posúvať po analyzovanom signále s krokom 2 vzorky.

V závislosti na použitom klasifikačnom algoritme je možné použiť dva prístupy k extrakcii príznakov z kĺzajúceho sa okna:

[15] Nepríde k redukcii dimenzionality a tým pádom ani k strate informácií.
[16] Singular Value decomposition
[17] Pri vzorkovacej frekvencii 10Hz je teda dĺžka okná rovná časovému intervalu 1 sekundy.

- Neextrahovať príznaky – vzorky nachádzajúce sa v kĺzajúcom okne budú priamo privedené na vstup klasifikačného algoritmu
- Nájsť vhodné príznaky a extrahovať ich

Pri použití konvolučných neurónových sietí v úlohe klasifikačného algoritmu, ktoré vynikajú v rozoznávaní vzorov v dátach sa o extrakciu príznakov starajú konvolučné vrstvy týchto sietí. V prípade, ak by som sa rozhodol implementovať riešenie za pomoci nich, žiadnu aditívnu extrakciu príznakov by nebolo potrebné vykonávať – s vysokou pravdepodobnosťou by mohla byť dokonca aj kontraproduktívna. Tento prístup priameho použitia signálu bez extrakcie príznakov ako vstupu do konvolučnej neurónovej siete možno vidieť aj v práci [13]. Keďže si ale nekladiem za cieľ dosahovať tak vysokú úspešnosť klasifikácie ako autori práce rozhodol som sa, že konvolučné neurónové siete v záujme šetrenia výpočtovej a pamäťovej náročnosti mnou navrhovaného riešenia nepoužijem.

Ako klasifikačný algoritmus som sa rozhodol použiť MLP[18]. Tento výber odôvodňujem v nasledujúcej podkapitole. Pri použití MLP narozdiel od konvolučných neurónových sietí môže byť extrakcia príznakov prínosná[19]. Pri analýze signálu je možné sa na signál pozerať cez jeho dve domény:

- Časovú[20]
- Frekvenčnú[21]

V rámci časovej domény signálu môžno ako príznaky použiť jeho štatistické vlastnosti ako sú priemer, rozptyl, smerodajná odchýlka, medián, maximum, minimum, geometrický priemer a podobne. V rámci frekvenčnej domény môžno ako príznaky použiť napríklad amplitúdy jednotlivých frekvencií, amplitúdu dominantnej frekvencie, hodnotu dominantnej frekvencie a podobne.

Z časovej domény signálu som zvolil ako príznaky jeho aritmetický priemer, smerodajnú odchýlku a medián. Z frekvenčnej domény som vybral ako príznaky amplitúdy prvých 5 frekvencií[22]. Vyšší počet frekvencií nemá zmysel analyzovať vzhľadom na dĺžku kĺzajúceho sa okna[23]. O vhodnosti môjho výberu príznakov a návrhu klasifikačného postupu ma utvrdili aj práce s podobným riešením ako navrhujem a to [8],[17].

Vo výslednom riešení navrhujem možnosť voľby príznakov, ktoré budú použité a to z nasledujúcich variant (aby bolo následne možné vyhodnotiť vplyv jednotlivých príznakov na výsledok klasifikácie a počítanie opakovaní):

- Žiadne – vzorky z kĺzajúceho okna budú privedené priamo na vstup MLP
- Iba frekvenčné – amplitúda prvých 5 frekvencií získaných za pomoci metódy FFT
- Všetky – frekvenčné príznaky + priemer, smerodajná odchýlka a medián

[18] Multi Layer Perceptron viz 2.5.

[19] Pri konvolučných neurónových sieťach môže byť extrakcia príznakov prínosná tiež, no vo väčšine prípadov dokáže trénovací algoritmus nájsť vhodné konvolučné jadrá, ktoré dokážu vystihnúť charakter dát z pôvodného signálu lepšie ako z takzvaných „hand-crafted“ príznakov.

[20] Anglicky TD – Time Domain

[21] Anglicky FD – Frequency Domain

[22] Frekvencií 0Hz-4Hz.

[23] Dôvodom je dôsledok Nyguist-Shannonovho vzorkovacieho teorému https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem.

3.7 Klasifikácia

Pre účel klasifikácie prebiehajúcej aktivity na základe extrahovaných príznakov je potrebné zvoliť vhodný klasifikačný algoritmus. Medzi najčastejšie využívané klasifikačné algoritmy v oblasti HAR[24] patria algoritmy k-NN, SVM a algoritmy založené na neurónových sieťach – MLP, CNN, RNN.

Ako klasifikačný algoritmus som sa rozhodol zvoliť algoritmus MLP viz 2.5 s ktorého použitím a implementáciou už mám určité skúsenosti. V rozhodnutí o použití MLP ma utvrdili aj práce viacerých autorov a to konkrétne práce [17] a [8]. Autori v týchto prácach použili pri riešení podobného problému a s extrahovanými podobnými príznakmi práve klasifikačný algoritmus MLP. K výberu tohto algoritmu ma priviedla aj vylučovacia metóda. Algoritmus k-NN som vylúčil z dôvodu nižšej robustnosti v porovnaní s ostatnými algoritmami. Ďalej som vylúčil algoritmus SVM, keďže sa jedná o binárny klasifikačný algoritmus a cieľom tejto práce je klasifikácia 8 rôznych tried[25]. Pre účel viac triednej klasifikácie by bolo možné použiť aj sériu SVM klasifikátorov, tak ako je to uvádzané v práci [7]. Tento prístup by ale viedol k zvýšeniu zložitosti implementácie a pravdepodobne by nepriniesol lepšie výsledky ako použitie MLP ako možno vidieť v [8]. Na výber ostali algoritmy založené na využití neurónových sietí z ktorých som z dôvodu šetrenia výpočtového výkonu a pamäťovej náročnosti zvolil práve algoritmus MLP.

Pri použití algoritmu MLP je potrebné navrhnúť architektúru tejto neurónovej siete – počet skrytých vrstiev, počet neurónov v jednotlivých vrstvách a použité aktivačné funkcie pre neuróny v jednotlivých vrstvách.

K návrhu architektúry neurónových sietí možno pristupovať viacerými spôsobmi. V prípade, že je možné navrhované neurónové siete v rozumnom čase[26] vytrénovať a následne overiť na testovacej dátovej sade, je možné zvoliť postup iteratívneho návrhu architektúry siete. Naskytujú sa dve možnosti:

1. Navrhnúť sieť s vyšším počtom vrstiev a neurónov v nich a postupne tieto počty redukovať.
2. Navrhnúť sieť s nižším počtom vrstiev a neurónov v nich a postupne tieto počty zvyšovať.

Pri použití prvej varianty je navrhnutá zložitejšia architektúra siete. Táto sieť sa následne vytrénuje a overí na trénovacích dátach – cieľom je aby sieť dokázala úspešne klasifikovať videné dáta. Pokiaľ to sieť nedokáže, architektúru je možné rozšíriť o ďalšie vrstvy a neuróny[27] Následne sa vyhodnotí schopnosť generalizácie siete na nevidených testovacích dátach. Pokiaľ nastal problém pretrénovania, architektúra sa postupne zmenšuje pokým sa nedosiahne požadovaných výsledkov. Pri použití druhej varianty sa postupuje opačne. Architektúra sa postupne zväčšuje pokým nie je sieť schopná dosahovať požadované výsledky. Navrhujem teda architektúru siete reprezentovanú tabuľkou 3.1.

[24] Human Activity Recognition

[25] 7 typov cvičení s vlastnou váhou a voľný pohyb.

[26] Rádovo minúty, desiatky minút až jednotky hodín.

[27] Pokiaľ sieť presiahne určité rozmery a stále nedokáže úspešne klasifikovať videné údaje, je vhodné prehodnotiť vhodnosťou extrahovaných príznakov. Je možné, že tieto príznaky nie sú dostačujúce pre klasifikácie trénovacích dát.

Poradie vrstvy	Typ	Počet neurónov	Aktivačná funkcia
1.	Vstupná	24	–
2.	Skrytá	12	ReLU
3.	Skrytá	10	ReLU
4.	Výstupná	8	Sigmoid

Tabuľka 3.1: **Navrhovaná architektúra siete MLP**

Počet neurónov vo vstupnej a výstupnej vrstve je pevne daný počtom extrahovaných príznakov a počtom požadovaných tried, medzi ktoré má byť pozorovaný signál zaradený. Každému príznaku teda prislúcha jeden neurón vo vstupnej vrstve. Celkový počet neurónov vstupnej vrstvy je 24 (8 príznakov pre 3 osi). Pri implementácii je požadované aby bol tento počet odvodený v závislosti od aktuálne zvoleného módu extrakcie príznakov (počet neurónov teda bude rovný počtu extrahovaných príznakov krát 3 osi) viz navrhovaná voľba príznakov v sekcii 3.6. Počet neurónov 2. vrstvy som zvolil ako polovicu počtu neurónov vstupnej vrstvy. Počet neurónov 3. vrstvy som zvolil ako stred medzi počtom neurónov v 2. a 4. (výstupnej) vrstve. Ako aktivačnú funkciu skrytých vrstiev navrhujem použiť funkciu `ReLU` viz 2.5. Táto funkcia je výpočtovo menej náročná v porovnaní s funkciou `sigmoid`. Vďaka tejto vlastnosti bude tréning ako aj samostatná klasifikácia za pomoci tejto siete rýchlejšia v porovnaní s prípadom použitia funkcie `sigmoid`. Pre výstupnú vrstvu navrhujem použiť funkciu `sigmoid`. Alternatívne by bolo možné použiť aj funkciu `softmax` viz [9] – táto funkcia by previedla hodnoty výstupných signálov jednotlivých neurónov na pravdepodobnosti príslušnosti klasifikovaného vzorku k daným triedam, pričom by bol súčet týchto pravdepodobností rovný 1. Funkcia `softmax` je často využívaná v neurónových sieťach, ktoré klasifikujú dáta do viacerých tried. Cieľ tejto práce vyžaduje tvrdé rozhodnutie o zaradení vzorku do jednej z 8 tried. Z tohto dôvodu je znalosť pravdepodobnosti príslušnosti vzorku medzi jednotlivé triedy nepotrebná a keďže je funkcia `softmax` výpočtovo náročnejšia ako funkcia `sigmoid` navrhujem vo finálnom riešení použiť funkciu `sigmoid`. Výsledkom klasifikácie bude trieda[28] pohybu, ktorá bude určená ako trieda pohybu prislúchajúca k neurónu s najvyššou hodnotou výstupného signálu.

Navrhovaná architektúra bude overená tréningom a následným testovaním na nevidených dátach. Môže nastať prípad, že táto sieť bude nedostatočná pre klasifikáciu skúmaného signálu. V tomto prípade bude potrebné architektúru upraviť rozšírením tejto siete. Druhým možným (zároveň pravdepodobnejším) scenárom je, že táto sieť bude vedieť úspešne klasifikovať videné dáta no zlyhá v generalizácii pri testovaní na nevidených dátach. V tomto prípade navrhujem do siete pridať špeciálne „Dropout“ vrstvy viz [14], ktoré budú umiestnené medzi 2.-3. a 3.-4. vrstvou tejto siete. Hodnotu parametru týchto vrstiev navrhujem určiť experimentálne.

Okrem návrhu architektúry siete MLP je potrebné navrhnúť aj spôsob akým bude táto sieť trénovaná. Preto navrhujem nasledujúci postup:

1. Z pilotnej dátovej sady budú vystrihnuté jednotlivé opakovania cvikov[29].
2. Každé z týchto opakovaní bude predspracované.

[28] Druh vykonávaného cviku.

[29] Vystrihnuté budú aj „opakovania“ voľného pohybu. Tieto „opakovania“ budú reprezentovať časť signálu nachádzajúcu sa medzi jednotlivými sériami cvikov. Anotáciu týchto „opakovaní“ možno získať nástrojom `annotation_add_free_motion`.

3. Z predspracovaného opakovania budú extrahované príznaky za pomoci kĺzajúceho sa okna[30] – pre každú pozíciu okna bude vytvorená nová položka v poli extrahovaných príznakov do ktorej budú uložené extrahované príznaky z tohto okna.

4. Do poľa reprezentujúceho požadovaný výstup siete bude n-krát vložený element, kde n je rovné počtu položiek ktoré boli v predchádzajúcom kroku vytvorené. Tento element bude obsahovať 8 miestne pole čísiel. Na poradovom mieste, ktoré zodpovedá aktuálnej triede do ktorej cvik patrí bude hodnota 1. Na ostatných miestach budú 0.

5. Tieto dve polia budú použité za účelom tréningu siete MLP. Prvé pole reprezentuje vstupy siete, druhé pole reprezentujú požadované výstupy siete prislúchajúce k daným vstupom.

Veľký časový úsek nahrávok tréningov je tvorený oddychom medzi jednotlivými sériami cvičení. Vďaka tomu bude trieda voľný pohyb tvoriť rádovo väčšiu časť z celkového počtu vstupov určených pre tréning siete MLP v porovnaní s ostatnými triedami. Signál reprezentujúci voľný pohyb obsahuje široké spektrum rôznych pohybov vďaka čomu je jeho vnútro-triedny rozptyl oveľa vyšší ako vnútro-triedny rozptyl iných tried. Z týchto dvoch vlastností vyplýva, že sieť MLP bude pri klasifikácii signálu preferovať triedu voľný pohyb v porovnaní s inými triedami. Extrahované príznaky zo skúmaného signálu teda musia byť oveľa viac podobné trénovacím príznakom triedy konkrétneho cviku, aby bol signál do tejto triedy zaradený. Táto vlastnosť je žiadúca, keďže do veľkej miery pomáha predchádzať zaradeniu neznámeho pohybu/cvičenia do jednej z tried reprezentujúcich konkrétne známe cviky.

Potrebné je navrhnúť aj priradenie tried jednotlivých aktivít ku konkrétnym výstupným neurónom siete MLP. Z tohto dôvodu uvádzam tabuľku 3.2, ktorá reprezentuje návrh tohto priradenia.

Názov cviku	**Index priradeného neurónu**
PushUps (Kľuky)	0
Squats (Drepy)	1
Plank (Podpor na predlaktiach)	2
TricepDips (Tricepsové kľuky na lavičke)	3
Lunges (Výpady)	4
KneeRaises (Príťahy kolien v sede)	5
SitUps (Sed-ľahy)	6
FreeMotion (Voľný pohyb)	7

Tabuľka 3.2: **Návrh priradenia tried aktivít k jednotlivým neurónom výstupnej vrstvy klasifikátora.**

3.8 Agregácia výsledkov a segmentácia

Výsledkom klasifikácie príznakov extrahovaných z jednotlivých pozícií kĺzajúceho sa okna sú informácie o triedach do ktorých boli tieto kĺzajúcim sa oknom orezané časti signálu

[30] Dĺžka okna aj prekrytie okna bude rovnaké ako pri extrakcii príznakov za účelom analýzy tréningu.

zaradené. Ako navrhujem v sekcii 3.6 dĺžka kĺzajúceho sa okna je 10 vzoriek a okno sa posúva tak, aby sa nová pozícia okna kryla s tou predchádzajúcou na 80 % jeho dĺžky. Dôsledkom je, že prvé dve vzorky signálu obsiahnuté v konkrétnej pozícii kĺzajúceho sa okna sa už v jeho ďalšej pozícii nachádzať nebudú. Z tohto dôvodu vyplýva, že je možné priradiť triedu získanú klasifikáciou signálu tejto pozície okna práve jeho prvým 2 vzorkám. Týmto spôsobom možno vytvoriť nový signál, ktorý bude reprezentovať výstup klasifikátora. Tento signál budem označovať pojmom „Výstupný signál klasifikátora“. Pre tento signál platí, že jeho i-tá vzorka označuje triedu do ktorej bola zaradená $(2i)$-tá a $(2i+1)$-tá vzorka analyzovaného signálu. Ukážku takéhoto signálu možno vidieť v tabuľke 3.3.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Hodnota signálu	7	7	7	**0**	**0**	**0**	**2**	**0**	7	**0**	**2**	**0**	7	7	7

Tabuľka 3.3: **Ukážka možného výstupného signálu klasifikátora.** Hrubým písmom a podčiarknutím sú zvýraznené hodnoty patriace do niektorej z tried reprezentujúcich 7 typov rozpoznávaného cvičenia.

Ako možno vidieť z tabuľky 3.3 výstupný signál klasifikátora nemusí byť rovný hodnote triedy aktuálne vykonávaného pohybu v celej dĺžke signálu, ktorá tento pohyb reprezentuje. Z tohto dôvodu je potrebné navrhnúť spôsob ako výstupné dáta z klasifikátora agregovať a vytvoriť tak segmenty reprezentujúce rozpoznané vykonávania jednotlivých cvičení. V prípade signálu zobrazeného v tabuľke 3.3 to znamená rozpoznať segment vykonávania cvičenia (konkrétne cvičenia kľuky – hodnota 0), ktorý sa nachádza v intervale danom indexami 3 až 11. Cieľom je teda navrhnúť algoritmus, ktorý bude schopný oddeliť úseky v ktorých sa vykonávala nejaká rozpoznávaná aktivita – hodnota výstupného signálu nie je rovná hodnote 7, ktorá reprezentuje voľný pohyb. Trieda cvičenia, ktoré je reprezentované týmto segmentom bude určená ako najčastejšie sa vyskytujúca trieda v danom segmente. V prípade, že majú dve alebo viac tried rovnaký a zároveň najčastejší výskyt, bude zvolená trieda, ktorej číselná hodnota[31] je vyššia. Jednou z požadovaných vlastností tohto algoritmu je vynechanie segmentov, ktoré nie sú dostatočne dlhé a mohli by byť tvorené hodnotami, ktoré vznikli chybou klasifikácie signálu. Pri návrhu tohto algoritmu vychádzam z predpokladu, že dlhá séria hodnôt, ktoré nereprezentujú voľný pohyb znamená, že bolo v tomto úseku vykonávané cvičenie. Druhým predpokladom je to, že sa hodnota triedy vykonávaného signálu v tomto segmente nachádza s vyššou frekvenciou ako 50 % vzhľadom k počtu ostatných tried reprezentujúcich iný typ cvičenia – nie voľný pohyb.

Navrhujem teda nasledujúci algoritmus viz algoritmus 2. Vstupom algoritmu je výstupný signál klasifikátora. Jeho výstupom je pole segmentov. Jednotlivé položky tohto poľa pozostávajú z poľa troch elementov – čísla klasifikovanej triedy, index-u začiatku a konca segmentu v rámci analyzovaného signálu tj. originálneho signálu tréningu. Parametrami algoritmu sú hraničná hodnota čítača `segmentator_cntr_threshold` a hodnota dekrementačného kroku `segmentator_cntr_decrement`. Hodnota `segmentator_cntr_threshold` určuje počet po sebe idúcich ne-voľnopohybových tried potrebný k detekcii segmentu. Pri každej ne-voľno pohybovej hodnote je čítač `sg_cntr` inkrementovaný. Segment je detekovaný pokiaľ hodnota čítača dosiahne alebo presiahne nastavenú hraničnú hodnotu. Po dosiahnutí hraničnej hodnoty sa už čítač ďalej neinkrementuje. Začiatok segmentu je daný indexom na ktorom sa nachádza prvá ne-voľnopohybová hodnota, ktorá viedla k prvej inkrementácii čítača zo

[31] Hodnota určujúce poradie neurónu vo výstupnej vrstve, ktorý k tejto triede prislúcha. Hodnoty možno nájsť v tabuľke 3.2.

stavu nulovej hodnoty. Hodnota `segmentator_cntr_decrement` udáva krok, ktorým bude čítač dekrementovaný pri triede voľný pohyb. Koniec segmentu je detekovaný potom ako čítač po dekrementácii dosiahne hodnotu $<= 0$ a tejto udalosti predchádzala detekcia začiatku segmentu. Koniec segmentu je daný ako -1 plus index prvej voľno pohybovej hodnoty z neprerušenej série voľno pohybových hodnôt, ktoré viedli k dekrementácii čítača na hodnotu $<= 0$.

Ako hraničnú hodnotu čítača navrhujem použiť hodnotu 5. Túto hodnotu volím ako dĺžku jedného okna pôvodného analyzované signálu (keďže každá vzorka výstupného signálu klasikátora zodpovedá 2 hodnotám pôvodného signálu a dĺžka kĺzajúceho sa okna je 10, táto hodnota bude $10/2 = 5$). Ako veľkosť dekrementačného kroku navrhujem použiť hodnotu rovnú 4-násobku dĺžky kĺzajúceho sa okna. Hodnota dekrementačného kroku teda bude 0.25, postup výpočtu vidno v rovnici 3.1. Parameter `dlzka_1_okna` označuje dĺžku, ktorú reprezentuje kĺzajúce sa okno v rámci výstupného signálu klasifikátora $(10/2 = 5)$.

$$\frac{\text{hranicna\_hodnota}}{\text{dlzka\_1\_okna}} \cdot \frac{1}{\text{x\_nasobok\_dlzky}} = \frac{5}{5} \cdot \frac{1}{4} = 0.25 \tag{3.1}$$

Pri vykonávaní sérií niektorých typov cvičení ako sú napríklad drepy sa v tréningovom signále nachádzajú veľké rozdiely medzi signálom v časti prebiehajúceho opakovania cviku a signálom v časti zodpovedajúcej krátkej prestávke medzi opakovaniami. Tento rozdiel vedie k vyššej pravdepodobnosti zaradenia prestávky medzi opakovaniami do triedy voľný pohyb. Segmenty, ktoré vzniknú použitím agregačného algoritmu popisovaného vyššie by teda obsahovali v závislosti na parametroch tohto algoritmu 1 až n opakovaní. Pokiaľ by boli parametre tohto agregačného algoritmu vhodne zvolené (napríklad by sa dekrementačný krok poupravil z 0.25 na hodnotu 1) viedlo by to k situácii, že jeden výstupný segment získaný agregačným algoritmom by bol rovný jednému opakovaniu daného cviku – napríklad drepov. Problém počítania opakovaní by potom spočíval v spočítaní týchto segmentov. Avšak existujú aj iné typy cvikov medzi, ktoré patria napríklad kľuky. Rozdiely medzi signálom v časti prebiehajúceho cviku a časti zodpovednej za prestávku medzi opakovaniami sú u týchto cvikov menšie ako u cvikov typu drepy. Vzhľadom na tento fakt je vysoká šanca že budú tieto prestávky medzi opakovaniami zaradené do rovnakej triedy ako samotné opakovania cvikov. Agregačný algoritmus by v tomto prípade vrátil jeden segment obsahujúci celú sériu kľukov. V tomto prípade by jednoduchý prístup k počítaniu opakovaní ako počítaniu segmentov použiť nešlo. Toto je dôvod prečo je potrebný návrh robustnejšieho algoritmu pre počítanie opakovaní, ktorý popisujem v nasledujúcej sekcii tejto kapitoly. Vstupom tohto algoritmu bude segment obsahujúci celú sériu cvikov.

Aby bolo možné získať segmenty reprezentujúce celú sériu opakovaní konkrétnych cvikov, je potrebné navrhnúť spôsob akým budú menšie segmenty spojené. Preto navrhujem spojiť všetky susedné segmenty, ktoré majú rovnakú triedu a sú od seba vzdialené menej alebo rovnako ďaleko ako hodnota novozavedeného parametra `segmentator_max_time_between_reps_ms`. Hodnotu tohto parametra navrhujem zvoliť na 7 sekúnd = 7000 ms. Tento čas považujem za minimálny čas prestávky medzi dvoma sériami rovnakých alebo podobných[32] cvičení.

Séria cvičení kľuky a cvik podpor na predlaktiach sú si z pohľadu výstupného signálu akcelerometra veľmi podobné. Podpor na predlaktiach vykonáva cvičiaci človek tak dlho,

[32] Podobnými cvičeniami značím cvičenia podpor na predlaktiach a kľuky.

Algoritmus 2: SEGMENTATOR

Input: predicted_classes
Output: segments

```
segmentator_cntr_threshold = 5
segmentator_cntr_decrement = 0.25
sg_cntr = 0
sg_start_idx = 0
sg_end_idx = 0
sg_active = False
sg_pred_class = ExerciseClasses.FreeMotion
last_class_was_exercise = False
segments = []
idx = 0
for pred in predicted_classes do
    if pred == ExerciseClasses.FreeMotion then
        if last_class_was_exercise then
            sg_end_idx = tmp_idx - 1
            last_class_was_exercise = False
        end if
        if sg_cntr > 0 then
            sg_cntr -= segmentator_cntr_decrement
            if sg_cntr <= 0 then
                sg_cntr = 0
                if sg_active then
                    pc_idx_start = int(sg_start_idx/2)
                    pc_idx_end = (int(sg_end_idx/2)+1)
                    part_of_signal =
                    predicted_classes[pc_idx_start:pc_idx_end]
                    eclass = find_most_frequent_value(part_of_signal)
                    if eclass != ExerciseClasses.FreeMotion then
                        segments.append([eclass, sg_start_idx, sg_end_idx])
                    end if
                end if
                sg_active = False
            end if
        end if
    else
        if sg_cntr <= 0 then
            sg_start_idx = tmp_idx
        end if
        if sg_cntr < segmentator_cntr_threshold then
            sg_cntr += 1
            if sg_cntr >= segmentator_cntr_threshold then
                sg_active = True
            end if
        end if
    idx += 2
end for
```

ako kondične vládze. Tesne pred koncom tohto cviku je silovo natoľko vyčerpaný, že sa jeho telo celé trasie. Toto trasenie spolu s rovnakou pozíciou ako pri vykonávaní kľukov môže viesť k zmäteniu klasikátora a milnej zámene medzi týmito triedami. Z tohto dôvodu navrhujem po vykonaní predchádzajúceho spojenia susedným segmentov s rovnakou triedou vykonať ešte jedno spojenie. Tento krát spojenie susedných segmentov, ktoré sú k sebe bližšie ako čas definovaný parametrom `segmentator_max_time_between_reps_ms` a zároveň je trieda jedného z nich rovná triede kľuky, zatiaľ čo trieda druhého z nich je rovná triede podpor na predlaktiach. Ako výsledná trieda tohto spojeného segmentu bude zvolená trieda do ktorej patril segment s väčšou dĺžkou (uvádzanou reálnym časom v ms a nie počtom vzoriek).

Ďalej zavádzam parameter `min_exercise_series_time_in_count_of_reps`. Tento parameter reprezentuje minimálny počet opakovaní cvikov v jednej sérií. Dĺžka každého výstupného segmentu bude porovnaná s hodnotou, ktorá bude vypočítaná ako minimálny počet opakovaní $*1.5*$ minimálna dĺžka trvania cviku patriaceho do triedy danej triedou segmentu. Pokiaľ bude dĺžka segmentu kratšia ako vypočítaná hodnota, segment bude zahodený. Parameter `min_exercise_series_time_in_count_of_reps` priamo neznačí odstránenie všetkých segmentov, ktoré obsahujú menší počet opakovaní ako tento parameter. Parameter slúži k výpočtu minimálnej akceptovanej časovej dĺžky segmentu. Hodnotu 1.5 pri výpočte času som zvolil z predpokladu, že jedno opakovanie cvičenia a po ňom nasledujúca pauza nebudú nikdy kratšie ako 1.5-násobok hodnoty minimálnej dĺžky trvania jedného opakovania cviku.

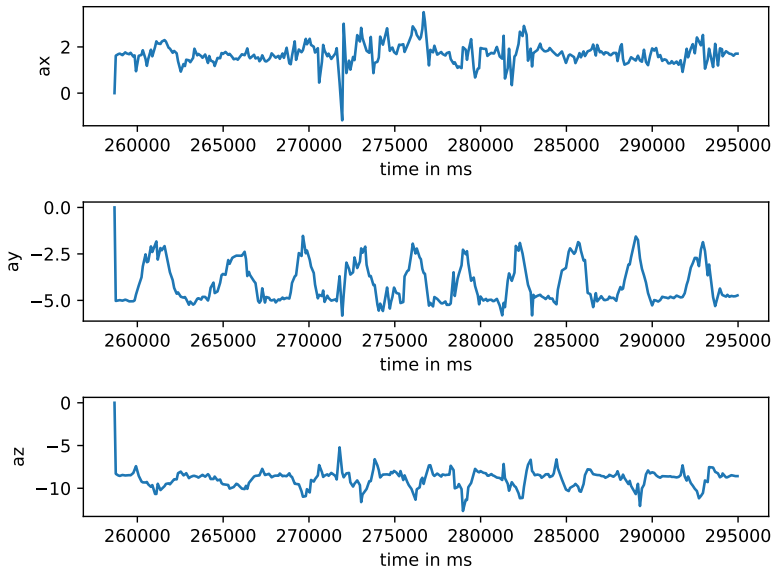
Segmenty, ktoré prešli všetkými krokmi budú ďalej spracované. Pri segmentoch reprezentujúcich podpor na predlaktiach bude spočítaný čas strávený v podpore. Ostatné segmenty budú posunuté algoritmu pre počítanie opakovaní.

3.9 Počítanie opakovaní

Výsledkom kroku segmentácia sú segmenty obsahujúce série jednotlivých cvičení. Každá z týchto sérií obsahuje niekoľko opakovaní rovnakého cviku. Aby bolo možné tieto opakovania spočítať, je potrebné navrhnúť spôsob ako ich v rámci série cvikov identifikovať. Priebeh zrýchlenia počas vykonávania akéhokoľvek cviku sa v čase mení. Každý typ cviku má oblasť v ktorej je jeho zrýchlenie najväčšie (dosiahne vrchol) a následne znovu klesá. Jedno opakovanie môže pozostávať z viacerých takýchto oblastí („vrcholov“) v signále. V rámci priebehu jedného opakovania cvičenia je v jeho signále možné nájsť jeden najvyšší bod. Pokiaľ bude takýto bod prehlásený za znak jedného opakovania, problém počítania opakovaní možno definovať ako hľadanie lokálnych maxím skúmaného signálu. Segment obsahujúci sériu opakovaní konkrétneho cviku je reprezentovaný 3-dimenzionálnym signálom. Na obrázku 3.1 možno vidieť príklad takéhoto signálu – konkrétne sa jedná o sériu 10 opakovaní tricepsových kľukov, ktorá bola vystrihnutá z dátovej sady[33]. Ako vidno na obrázku, jednotlivé opakovania možno jednoducho opticky rozpoznať pohľadom na os ay.

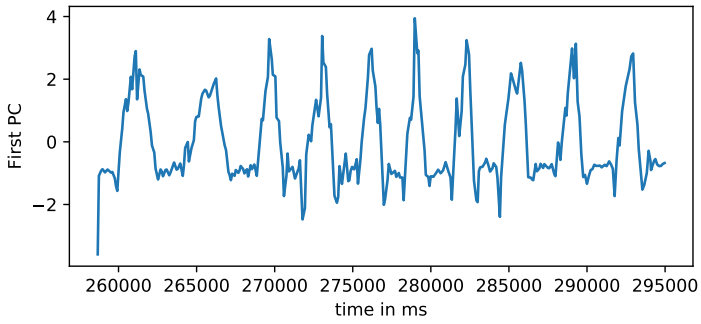
Smartfón zaznamenávajúci zrýchlenie počas tréningu sa nachádza pri vykonávaní rôzneho typu cvikov zakaždým v inej polohe. Dôsledkom je, že sa tieto „vrcholy“ v signále pri každom cviku prejavia v inej osi alebo kombinácii viacerých osí. Z tohto dôvodu je potrebné navrhnúť spôsob ako tento 3-dimenzionálny signál transformovať do 1-dimenzionálneho signálu v ktorom bude možné spočítať jednotlivé lokálne maximá a získať tak počet vykonaných

[33] Dátovej sady, ktorá vznikla v rámci tejto práce.



Obr. 3.1: **Ukážka signálu, reprezentujúca sériu cvičenia tricepsové kľuky.** V rámci série bolo vykonaných 10 opakovaní tohto cvičenia. Signál bol prefiltrovaný dolnopriepustným filtrom prvého rádu s medznou frekvenciou 4Hz.

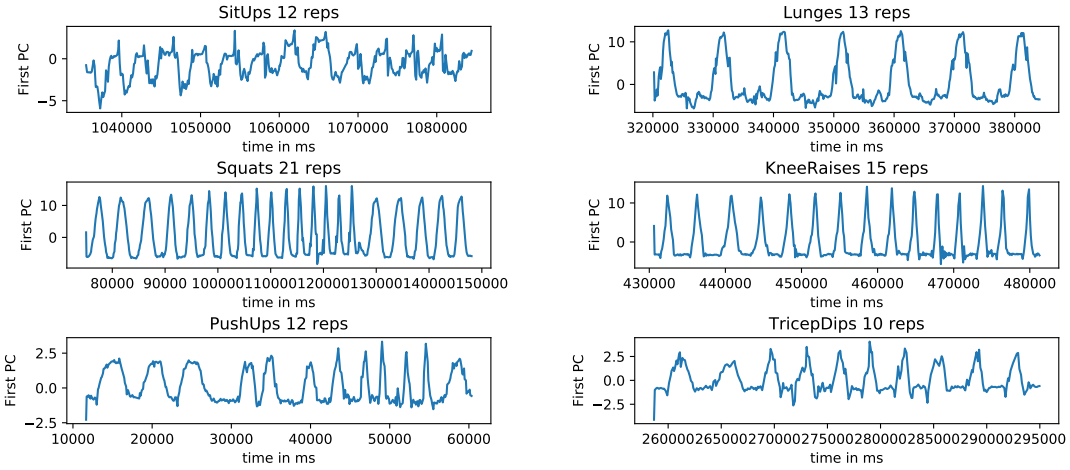
opakovaní. Jednoduchý spôsob ako takýto 1-dimenzionálny signál získať, je nájsť hlavné komponenty tohto signálu a transformovať tento signál do jeho prvého hlavného komponentu, ktorý značí os s najvyšší rozptylom. Výsledok aplikácie tejto transformáciu na signál z obrázku 3.1 možno vidieť na obrázku 3.2.



Obr. 3.2: **Ukážka signálu z obrázku 3.1 po pretransformovaní do jeho prvého hlavného komponentu.** Transformácia bola vypočítaná metódou PCA, pričom jej vstupom bol vyrezaný úsek signálu s dĺžkou 30 % z pôvodného signálu so stredom v strede pôvodného signálu.

Vďaka chybám klasifikácie môžu vzniknúť aj segmenty, ktoré budú na svojom začiatku alebo konci obsahovať aj časť signálu, ktorá už do série cvičenia nepatrí. Napríklad pri sérii kľukov, môže segment obsahovať časť signálu značiacu pohyb používateľa z polohy kľuku do polohy stoja. Zrýchlenie v tejto časti signálu môže byť väčšie ako zrýchlenie spôsobené jednotlivými opakovaniami cvičenia. Pokiaľ by bola metóda PCA (použitá pre nájdenie prvého

hlavného komponentu) aplikovaná nad celým takýmto signálom, mylne by našla transformáciu do osi značiacej smer pohybu z polohy kľukov do polohy stoj. Aby bolo možné tomuto nežiadúcemu efektu predísť, navrhujem hľadať spomínanú transformáciu nad 30 % signálu segmentu so stredom v strede signálu daného segmentu. Tento návrh vychádza z predpokladu, že druhá tretina segmentu bude obsahovať iba opakovania skúmaného cvičenia. Aplikáciu takejto transformácie nad signálmi jednotlivých segmentov všetkých 6 typov analyzovaných cvičení[34] možno vidieť na obrázku 3.3.



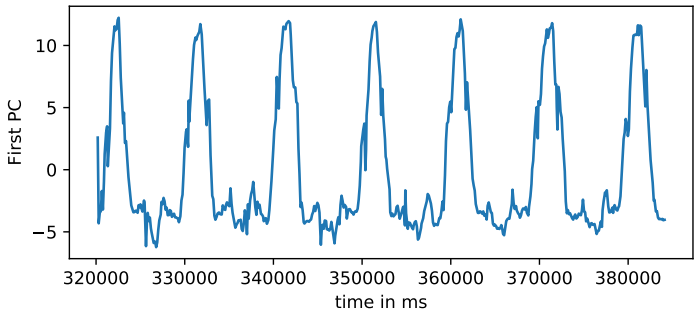
Obr. 3.3: **Ukážka signálu jednotlivých sérií cvičení po ich transformovaní do prvého hlavného komponentu.** Transformácia bola vypočítaná spôsobom popísaným v popise obrázku 3.2.

Výpady (anglicky „Lunges“) sú jediným typom cvičenia, kde po sebe idúce opakovania nemusia vyzerať rovnako. Opakovania ostatných typov cvičení sú z pohľadu tela cvičiaceho človeka symetrické. To znamená, že nezáleží na umiestnení smartfónu – zrýchlenie vplýva na smartfón rovnako nezávisle od toho, či je umiestnený v ľavom alebo pravom vrecku cvičiacej osoby. Výpad patrí medzi takzvané párové cviky – opakovania sa môžu[35] striedavo vykonávať ľavou aj pravou nohou. Opakovanie výpadu sa vykonáva tak, že cvičiaci človek zo stoja spojného vykročí kročnou nohou vpred. Koleno kročnej nohy bude po dokročení zvierať uhol 90 stupňov. Koleno zadnej nohy je spúšťané smerom k zemi až do bodu jemného dotyku s podlahou. Z pohľadu smartfónu to znamená, že namerané zrýchlenie v prípade, že sa smartfón nachádza vo vrecku kročnej nohy bude väčšie a zároveň bude smerovať aj iným smerom ako zrýchlenia namerané pri umiestnení smartfónu vo vrecku zadnej nohy. V praxi to teda znamená, že sa v signále segmentu obsahujúcom sériu výpadov budú nachádzať 2 typy vrcholov. Prvý typ bude reprezentovať opakovania výpadov pri ktorých sa smartfón nachádzal vo vrecku kročnej nohy. Tieto vrcholy budú mať vyššiu amplitúdu a preto bude signál za pomoci PCA transformovaný práve do hlavného komponentu reprezentujúceho smer zrýchlenia týchto opakovaní. Druhý typ bude reprezentovať opakovania výpadov pri ktorých sa smartfón nachádzal vo vrecku zadnej nohy. Amplitúda týchto opakovaní bude

[34] Pri cvičení podpor na predlaktiach sa nepočítajú opakovania. Miesto opakovaní sa počíta čas strávený v polohe tohto cviku.

[35] Táto podmienka nie je nutná, preto používam slovíčko „môžu“. Na tento predpoklad sa teda nemožno spoliehať.

nižšia a zároveň smer ich zrýchlenia bude tiež odlišný. Po transformovaní signálu série výpadov do prvého hlavného komponentu vznikne signál zobrazený na obrázku 3.4.



Obr. 3.4: **Ukážka signálu série 13 výpadov po ich transformovaní do prvého hlavného komponentu.** Transformácia bola vypočítaná spôsobom popísaným v popise obrázku 3.2.

Ako možno vidieť na obrázku 3.4 opakovania výpadov pri ktorých bol smartfón umiestnený vo vrecku zadnej nohy sa v signále prejavili ako lokálne minimá. Pri počítaní opakovaní výpadov možno problém opakovania počítaní definovať ako počítanie lokálnych maxím signálu nasledované počítaním lokálnych miním signálu.

Pre počítanie opakovaní teda navrhujem nasledujúci algoritmus. V skúmanom signále sa za pomoci kĺzajúceho okna extrahujú lokálne maximá. Dĺžka tohto okna musí byť menšia alebo rovná ako dĺžka najkratšieho možného opakovania cviku. Navrhujem aby sa jednotlivé pozície kĺzajúce sa okna prekrývali na 50 % jeho dĺžky. Vďaka tomu bude možné lokalizovať aj menšie lokálne maximá, ktoré by inak mohli byť „zatienené“ vzorkou s vyššou hodnotou, ktorá by ale patrila opakovaniu, ktorého vrhol bol už extrahovaný[36]. Samotnú dĺžku kĺzajúceho sa okna navrhujem zvoliť rovnú 20 vzorkám signálu. Pre každé lokálne maximum sa uloží jeho amplitúda spolu s jeho pozíciou v rámci skúmaného signálu. V prípadne duplikátnej detekcie rovnakého vrcholu sa tento vrchol do poľa uloží len raz. Pole týchto maxím sa zoradí zostupne podľa ich amplitúdy. Vychádzajúc z predpokladu, že vrcholy tohto 1-dimenzionálneho signálu značia opakovania cvičení, bude vrchol s najvyššou amplitúdou označený za opakovanie cvičenia. V druhej tretine transformovaného skúmaného signálu sa zároveň nájde minimum. Z amplitúdy najvyššieho vrcholu a tohto minima sa vypočíta maximálna možná amplitúda opakovania. Následne sa pripočíta dĺžka 40 % z tejto amplitúdy k hodnote nájdeného minima. Výsledná hodnota bude značiť minimálnu požadovanú hodnotu signálu aby mohol byť vrchol uznaný ako opakovanie cvičenia. Pozícia každého vrcholu nachádzajúceho sa v poli vrcholov (iterujúc zostupne podľa amplitúdy týchto vrcholov) je porovnaná s pozíciami už uznaných vrcholov. Pokiaľ je vzdialenosť od najbližšieho uznaného vrcholu väčšia ako minimálneho dĺžka trvania opakovania, tento vrchol je uznaný za opakovanie cvičenia (kontroluje sa ešte predpoklad minimálnej výšky tohto vrcholu, ktorý spomínam vyššie). Týmto spôsobom je možné spočítať opakovania jednotlivých cvičení. Pri cvičení výpady sa následne ešte prevedie detekcia lokálnych miním. Tieto minimá sa zoradia vzostupne a následne sa cez toto pole iteruje a uznávajú sa vrcholy s dostatoč-

[36] V prípade, že by začiatok pozície kĺzajúceho sa okna obsahoval napríklad posledných 40 % opakovania cviku, mohlo ostať nasledujúce opakovanie, ktorého vrchol sa nachádza v rovnakej pozícii kĺzajúceho sa okna z dôvodu nižšej hodnoty jeho vrcholu nedetekované.

nou vzdialenosťou od iných už uznaných opakovaní cvičenia. V prípade lokálnych miním sa ich hodnota výšky nekontroluje. Túto kontrolu nenavrhujem z dôvodu, že hodnota ich výšky v rámci 1-dimenzionálneho signálu získaného transformáciou popisovanou vyššie, nie je v tomto smere (udávanom smerom hlavného komponentu) dostatočne výrazná na to, aby bola takáto výšková kontrola efektívna. Z dôvodu nezavedenia tejto kontroly môže byť detekovaný vyšší počet opakovaní pri neštandardnom[37] cvičení tohto cviku. Podobný postup počítania opakovaní ako navrhujem použili aj autori v práci [7]. Autori však použili k vylučovaniu detekovaných vrcholov odhad periódy cviku za pomoci autokorelácie a pri vylučovaní vrcholov na základe ich amplitúdy zvolili iný prístup v porovnaní s mnou navrhovaným prístupom. Ďalším rozdielom medzi ich a mnou navrhovaným riešením je, že pre nájdenie transformácie za pomoci PCA bola použitá celá dĺžka segmentu. V práci ďalej nebolo riešené počítanie výpadov ani iných párových „nesymetrických“ cvikov.

[37] V prípade, že cvičiaca osoba vykonáva iba opakovania cviku pri ktorých sa smartfón nachádza vo vrecku kročnej nohy. Zároveň musí platiť, že sú medzi týmito opakovaniami vynechané medzery s dĺžkou aspoň 2-násobok minimálnej dĺžky opakovania cviku výpad.

Kapitola 4

Aplikácia pre počítanie opakovaní cvikov s vlastnou váhou tela

Na základe navrhnutého riešenia je potrebné toto riešenie implementovať a overiť jeho funkčnosť. Cieľom tejto kapitoly je popísať tento postup implementácie a overovania funkčnosti navrhovaného riešenia. Na začiatku popíšem implementáciu aplikácií pre zber dát z pohybových senzorov. Ďalej nadviažem popisom nástrojov pre prácu so zozbieranými dátami a ich vizualizáciu. Pokračovať budem popisom metodiky zberu dát, postupom vytvorenia dátovej sady a samotným popisom takto vzniknutej dátovej sady. Popíšem implementáciu finálneho riešenia a následne toto riešenie vyhodnotím za pomoci vzniknutej dátovej sady. Výsledky tohto vyhodnotenia analyzujem a vyvodím závery.

4.1 Nástroje pre nahrávanie a anotáciu dát

Mobilné aplikácie je možné vyvíjať v širokom spektre jazykov a frameworkov pre viacero mobilných platforiem. Ako cieľovú platformu som zvolil Android[1] vzhľadom na to, že je najpoužívanejšou[2] mobilnou platformou v čase písania tejto práce. Pre implementáciu nahrávacej a anotačnej aplikácie som sa rozhodol použiť open-source mobilný framework s názvom Apache Cordova[3]. Cordova umožňuje tvorbu aplikácie v HTML/JavaScript a jej následný preklad do natívneho kontajneru pre jednotlivé platformy. Z tohto dôvodu je možné tieto aplikácie bez väčších zmien preložiť aj pre platformu iOS[4]. Práve táto prenositeľnosť a prehľadnosť programovania v tomto frameworku bola dôvodom prečo som sa rozhodol tento framework použiť.

Aplikácie naprogramované v tomto framework-u možno preložiť príkazom[5] `cordova build`. Výsledkom prekladu je inštalačný súbor s príponou `.apk`, ktorý sa nachádza v adresári `platforms/android/app/build/outputs/apk/debug/`.

[1] Viz https://www.android.com/
[2] Viz https://gs.statcounter.com/os-market-share/mobile/worldwide
[3] https://cordova.apache.org/
[4] Viz https://en.wikipedia.org/wiki/IOS
[5] Spustením v adresári so zdrojovým kódom aplikácie.

4.1.1 Aplikácia `Akcelerak`

Pre vytvorenie dátovej sady je potrebné zaznamenávať dáta z pohybových senzorov smartfónu, ktorý je umiestnený vo vrecku cvičiaceho používateľa. Aby bolo možné tieto dáta zbierať, je potrebné aby v tomto zariadení bežala špeciálna aplikácia. Touto špeciálnou aplikáciou je aplikácia `Akcelerak`. Účelom aplikácie `Akcelerak` je zbieranie dát z pohybových senzorov a ich následne ukladanie vo formáte špecifikovanom v sekcii 3.3 do vnútornej pamäte telefónu. Z nej budú následne presunuté do počítača, kde budú ďalej za pomoci nástrojov spracované.

Táto aplikácia teda musí vedieť získavať a následne ukladať dáta z pohybových senzorov. Pre získavanie týchto dát sú použité zásuvné moduly `cordova-plugin-device-motion` a `cordova-plugin-device-gyroscope`. Parameter `frequency` je nastavený na hodnotu 100, ktorá reprezentuje vzorkovaciu periódu v milisekundách viz 2.2. Pre ukladanie dát je potrebné použiť ďalší zásuvný modul s názvom `cordova-plugin-file`. Pri stlačení tlačítka „Start logging“ viz obrázok 4.1 sú v koreňovom adresári vnútornej pamäte zariadenia vytvorené dva súbory – jeden pre akcelerometer, druhý pre gyroskop. Do týchto súborov je vložený riadok označujúci názvy jednotlivých stĺpcov. Zároveň sa pri stlačení tohto tlačítka uloží počiatočné časové razítko označujúce čas začiatku nahrávania. Ďalej sú inicializované premenné reprezentujúce „vyrovnávaciu pamäť“ a premenné obsahujúce počet prvkov uložených v týchto pamätiach.

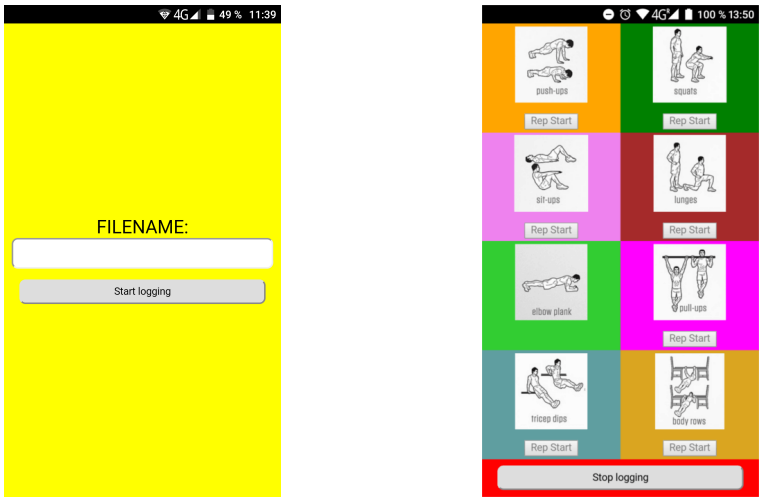
Po každom prijatí nových dát z pohybového senzoru je vypočítaná ich magnitúda (at/gt) a čas relatívny k času začiatku tréningu (čas vyhotovenia vzorky mínus čas začiatku tréningu). Dáta reprezentujúce zrýchlenia/uhľové rýchlosti pre osi x,y a z sú spolu s ich magnitúdou a relatívnym časom vyhotovenia vzorky prevedené do podoby textového reťazca (rešpektujúc formát navrhovaný v 3.3). Tento reťazec je pripojený na koniec reťazca uloženého v premennej reprezentujúcej vyrovnávaciu pamäť pre tento pohybový senzor (jedna pamäť pre akcelerometer, jedna pre gyroskop). Inkrementuje sa počítadlo prvkov vo vyrovnávacej pamäti a keď presiahne hodnotu 50 (ekvivalent 5 sekundám nahrávania pri frekvencii 10Hz) je obsah tejto pamäte uložený na koniec výstupného súboru. Pamäť sa vyprázdni a počítadlo vynuluje. Túto techniku „buffer-ingu“ som zaviedol z dôvodu odľahčenia potreby neustáleho zápisu pri prijatí novej vzorky, ktorý by zbytočne zaťažoval systém neustálymi požiadavkami o prácu so súbormi. Aby bolo možné jednoducho overiť či zariadenie dostáva údaje z pohybových senzorov, rozhodol som sa ich okrem zapisovania do súborov aj zároveň zobrazovať na obrazovke zariadenia viz obrázok 4.1. Týmto spôsobom je možné okrem overenia či zariadenie disponuje jednotlivými pohybovými senzormi (pri niektorých lacných telefónoch sa ukázalo, že neobsahujú gyroskop) overiť aj to, či jednotlivé osi týchto senzorov zodpovedajú osiam, ktoré sú ilustrované na obrázku 2.1.

Posledná vec, ktorú bolo potrebné vyriešiť, je schopnosť aplikácie získavať dáta a zapisovať ich aj počas režimu zamknutej obrazovky. Ako spomínam v sekcii 2.2, získavanie dát v tomto režime môže byť problém od Android-u verzie novšej alebo rovnej verzii 9. Tento problém som vyriešil za pomoci zásuvného modulu `cordova-plugin-background-mode`. Okrem samotného „background módu“ bolo potrebné tiež vypnúť Android-ové optimalizácie za pomoci funkcie `disableWebViewOptimizations` obsiahnutej v tomto module. Týmto spôsobom sa podarilo zaručiť nahrávanie aj pri uzamknutom telefóne (otestované na Android-e verzie 10). Nevýhoda tohto riešenia spočíva v možných problémoch (s porušením podmienok) pri publikácii takejto aplikácie do aplikačného obchodu Google Play.

Obr. 4.1: **Ukážka grafického rozhrania nahrávacej aplikácie `akcelerak`.**

4.1.2 Aplikácia `Anotator`

Aby bolo možné anotovať dáta získané aplikáciou **`Akcelerak`**, je potrebné vyvinúť ďalšiu aplikáciu – aplikáciu, ktorá bude slúžiť na zaznamenávanie anotácií. Touto anotačnou aplikáciou je aplikácia **`Anotator`**. Na obrázku 4.2 možno vidieť grafické rozhranie tejto aplikácie. Obrázky označujúce jednotlivé cviky som prevzal zo stránky `https://darebee.com/muscle-map.html`. Aplikácia si po stlačený tlačítka „Start logging“ uloží počiatočné časové



Obr. 4.2: **Ukážka grafického rozhrania anotačnej aplikácie `anotator`.**

razítko. S každým uloženým anotačným reťazcom[6] je do výstupného súboru uložený aj relatívny čas jeho vytvorenia (aktuálny čas mínus hodnota počiatočného časového razítka). Anotáciu začiatku série cvičenia možno vytvoriť stlačením obrázku reprezentujúceho požadované cvičenie. Aktuálne prebiehajúca séria cvičení je značená zmenou vzoru pozadia ako vidno na obrázku 4.3. Pre vytvorenie začiatku anotácie opakovania cvičenia je potrebné

[6] Viz sekcia 3.3.

Obr. 4.3: **Ukážka symbolizácie prebiehajúcej série kľukov a prebiehajúceho opakovania kľuku v anotačnej aplikácii `anotator`.**

kliknúť na prislúchajúce tlačítko „Rep Start“. Aktívny priebeh anotácie opakovania je značený červeným pásikavým pozadím ako vidno na druhej časti obrázku 4.3. Pri vývoji tejto aplikácie som vybral niekoľko cvikov, ktoré sa bežne vykonávajú na „street workout“ ihriskách, vybrané cvičenia možno vidieť na obrázku 4.2. Neskôr som sa však rozhodol cvičenia zhyby a horizontálne príťahy (bodyrows) z tréningov vynechať[7]. Do tréningov pribudol nový cvik – príťahy kolien v sede (seated knee raises), rozhranie anotačnej aplikácie som však už nemenil. Pre anotáciu tohto novo pridaného cviku som používal možnosť „body rows“.

Aby boli anotácie korektné je potrebné aby obe aplikácie začali nahrávať v rovnaký moment. Tým sa synchronizuje čas[8] začiatku nahrávania. Táto synchronizácia sa dá dosiahnuť viacerými spôsobmi. Rozhodol sa tento proces ponechať na „manuálnej“ synchronizácii. Na dvoch telefónoch sa otvoria aplikácie, na jednom nahrávacia, na druhom anotačná. Jeden človek naraz stlačí tlačítko štart v oboch aplikáciach. Telefón s nahrávacou aplikáciou si cvičiaci vloží do vrecka a môže sa začať tréning. Časový rozdiel medzi stlačením týchto tlačítok je dostatočne malý na to aby nemusel byť braný do úvahy. Túto tézu som vizuálne overil na základe vizualízacie anotovaných tréningov nástrojom `spectrogram`. Takéto vizuálne overenie možno vidieť aj na obrázku C.2.

4.2 Nástroje pre prácu s dátami a ich vizualizáciu

Dáta získané zberom dát je potrebné ďalej spracovať za účelom vytvorenia dátovej sady. Táto dátová sada bude následne použitá na tréning a vyhodnotenie presnosti samotného klasifikátora, ako aj na vyhodnotenie presnosti finálneho riešenia – vyhodnotí sa aj presnosť samotného počítania opakovaní. Aby bolo možné dáta spracovať, je potrebné naprogramovať niekoľko nástrojov určených pre prácu so získanými dátami. Okrem toho je potrebné vyvinúť aj nástroje určené na vizualizáciu týchto dát. Na základe týchto nástrojov bude

[7] Z dôvodu, že sa nedajú vykonávať kdekoľvek bez potreby cvičebného náradia.

[8] Stĺpec `time` vo výstupných súborov nahrávacích aplikácií značí čas v milisekundách od začiatku nahrávania.

možné vizuálne overiť presnosť získaných anotácií ako aj vizuálne overiť činnosť mnou navrhovaných algoritmov.

Konkrétne je potrebné naprogramovať tieto nástroje:

1. `annotation_add_free_motion` – Nástroj slúžiaci na pridanie anotácie voľného pohybu na miesta kde sa nenachádza žiadna prebiehajúca anotácia.
2. `annotation_joiner` – Nástroj slúžiaci na napárovanie anotácií k jednotlivým dátovým vzorkám a ich spojenie do jedného CSV súboru.
3. `annotation_cutter` – Nástroj slúžiaci na vyrezanie vybraných stĺpcov dát pre konkrétne zvolenú anotáciu. Tento nástroj bude ďalej obsahovať aj možnosť zobraziť všetky použité anotácie v konkrétnom anotovanom[9] súbore.
4. `annotation_cut_all` – Nástroj, ktorý pracuje s nástrojom `annotation_cutter`. Za jeho pomoci získa zoznam anotácií v konkrétnom CSV súbore. Jednotlivé anotované časti za pomoci nástroja vystrihne a uloží do oddelených súborov. Názov týchto súborov bude pozostávať z názvu pôvodného súboru a názvu anotácie, ktorej daný súbor prislúcha.
5. `repetitions_sorter` – Nástroj, ktorý na základe preddefinovaných pravidiel roztriedi jednotlivé súbory, ktoré vznikli použitím nástroja `annotation_cut_all` do preddefinovaných priečinkov.
6. `spectrogram` – Vizualizačný nástroj slúžiaci na zobrazenie nameraných hodnôt v jednotlivých osiach za pomoci grafov. Pod každým grafom sa bude nachádzať spektrogram[10], ktorý bude zobrazovať vývoj frekvenčného spektra signálu pre danú os. Na základe preddefinovaných farieb k jednotlivým anotáciám bude možné vyfarbiť úseky pozadie grafov reprezentujúcich jednotlivé osy tak, aby konkrétne farby pozadia grafov v jednotlivých úsekoch reprezentovali priebeh konkrétnej anotovanej aktivity.

Popis ich implementácie a ich spôsob použitia uvádzam v prílohe C.

4.3 Zber údajov a vytvorenie dátovej sady

Za účelom vytvorenia dátovej sady je potrebné najskôr nahrať a anotovať niekoľko tréningov. Aby boli modely trénované na tejto dátovej sade schopné lepšie generalizovať je vhodné aby obsahovala tréningy od čo najväčšieho počtu rôznych ľudí.

Ako cieľovú aktivitu používateľa, ktorú v tejto práci budem analyzovať som zvolil cviky s vlastnou váhou tela cvičiaceho. Z týchto cvikov som vybral tie cviky, ktoré sú jednoducho realizovateľné v akejkoľvek miestnosti bez použitia žiadnych[11] strojov alebo cvičebného

[9] Súbore, ktorý vznikol spojením anotácií z anotačného súboru spolu s nameranými dátovými vzorkami z výstupného súboru reprezentujúceho dáta z akcelerometru/gyroskopu. Takýto súbor je možné vytvoriť za pomoci nástroja `annotation_joiner`

[10] Získaný za pomoci FFT použitím techniky kĺzajúceho sa okna pre zadanú dĺžku okna a percentuálny prekrytie týchto okien.

[11] Pre vykonávanie cviku „tricepsové kľuky na lavičke“ je potrebná stolička, posteľ alebo iné vyvýšené miesto.

náradia. Tieto cviky budú vykonávané v sériách – niekoľko opakovaní rovnakého cviku za sebou. Medzi jednotlivými sériami sa bude nachádzať ľubovoľne dlhá prestávka, počas ktorej sa bude cvičiaci používateľ voľne pohybovať v miestnosti. V rámci voľného pohybu bude vykonávať akékoľvek pohyby, ktoré sú pre neho pri cvičebných prestávkach bežné – prechádzanie sa, naťahovanie sa, sedenie a podobne.

Cvičiacim používateľom je pred tréningom vysvetlená technika cvičenia, aby bolo zaručené, že budú používatelia vykonávať jednotlivé cvičenia rovnakým spôsobom. Na dvoch smartfónoch sa spustia aplikácie `Anotator` a `Akcelerak`, zvolí sa rovnaký názov tréningu. V rovnaký časový moment sa stlačí tlačítko „Start logging“ v oboch aplikáciách. Cvičiaci používateľ si vloží uzamknutý smartfón s bežiacou aplikáciou `Akcelerak` do predného vrecka nohavíc a tréning sa môže začať. Cvičiaci používateľ prejde napríklad do polohy vykonávania kľukov. Anotujúci používateľ tento fakt zaznamená kliknutím na obrázok kľuky. Zároveň vyzve cvičiaceho používateľa slovom „Poď!“ aby vykonal opakovanie cvičenia. Začiatok a koniec opakovania cviku je značený použitím tlačítka „Rep Start/Rep Stop“ viz obrázok 4.3.

Poradie vykonávaných cvikov môže byť ľubovoľné, každý tréning by však mal obsahovať aspoň jednu cvičebnú sériu pre každý typ cvičenia. Nevyhnutou požiadavkou je aby tréning obsahoval aspoň jedno opakovanie cviku drepy, ktoré bude použité ako kalibračný cvik viz sekcia 3.5.

Dátová sada bude vytvorená tak, že pre každého cvičiaceho používateľa bude vytvorený priečinok s jeho menom. Tento priečinok bude obsahovať priečinky s názvom[12] jednotlivých tréningov, ktoré vykonal. Tréningové priečinky budú obsahovať dva priečinky. Priečinok `raw` obsahujúci dáta z nahrávacích aplikácií. Priečinok `splitted` obsahujúci vystrihnuté opakovania cvikov roztriedené do priečinkov podľa typu cvičenia.

Dáta z nahrávacích aplikácií sa presunú do priečinku `raw`. Za pomoci nástrojov sa spojí nahrávka tréningu s jej anotáciou. Vzniknutá anotovaná nahrávka sa vizuálne overí ako možno vidieť na obrázku C.2. V prípade, že sú anotácie vizuálne posunuté budú manuálne opravené. Z anotovanej nahrávky sa za pomoci nástrojov vyrežú jednotlivé opakovania do priečinku `splitted` a následne sa vytriedia do podpriečinkov. Manuálne sa vytvorí podpriečinok `calib` do ktorého sa umiestní jedno opakovanie cviku drepy. Pre každý tréning sa vytvorí ešte súbor `summary.txt`, ktorý bude obsahovať metadáta[13] daného tréningu.

Pri spracovaní tréningov bolo potrebné iba v jednom prípade manuálne upraviť posunutý čas anotácií. V troch prípadoch[14] bol čas na začiatku tréningu synchronizovaný správne, no v priebehu tréningu sa postupne akumulovala nepresnosť[15] až na konci tréningu dosiahla úroveň +0.3 sekundy. Tieto anotácie boli manuálne upravené tak aby presne zodpovedali vykonávaným opakovaniam v cele dĺžke tréningu. V nahrávkach tréningov sa prejavil aj problém s nepravidelným vzorkovaním a duplikátmi spomínaný v sekcii 2.2. Dôvodom vzniku problému je, že framework Cordova získava vzorky s periódou 100ms z Androidového rozhrania poskytujúceho vzorky s periódou 60ms. Bohužiaľ toto nastavenie nie je možné v danom zásuvnom module zmeniť.

[12] Ako názov tréningu som zvolil názov dňa v ktorom bol tréning vykonaný, vzhľadom na to, že ani jeden používateľ nevykonal dva tréningy v rovnakom dni v týždni.

[13] Trvanie tréningu, počty vykonaných cvikov a orientáciu zariadenia.

[14] Problém sa prejavil pri dlhších tréningoch.

[15] Pravdepodobne spôsobená rozdielnym meraním času dvoch smartfónov.

Číslo triedy	0	1	2	3	4	5	6
Počet opakovaní	152	207	9m 52s	150	195	154	138

Tabuľka 4.1: **Celkový počet opakovaní cvičení nachádzajúci sa v dátovej sade.** Miesto názvu cviku bolo použité číslo triedy (z dôvodu šetrenia priestoru). Názvy cvičení patriace k týmto číslam tried možno nájsť v tabuľke 3.2.

Názov	Pozícia	Dĺžka	C. Počet	0	1	2	3	4	5	6
marek_pondelok	ldv	37m 31s	206	41	41	68s	36	38	18	30
martin_utorok	ldv	09m 42s	66	6	15	35s	7	15	10	12
matej_utorok	pdn	23m 09s	147	10	41	95s	19	34	10	31
miro_nedela	pdn	11m 50s	86	15	11	65s	13	14	20	12
miro_piatok	pdn	18m 31s	158	28	33	99s	25	27	30	13
miro_streda	pdn	12m 16s	103	18	14	64s	8	21	22	19
miro_stvrtok	lhv	26m 00s	241	34	52	139s	42	46	44	21

Tabuľka 4.2: **Prehľad informácií o tréningoch v dátovej sade.** Názov označuje meno cvičiacej osoby a deň vykonania tréningu. Pozícia značí orientáciu telefónu vo vrecku. Dĺžka označuje celkový čas tréningu vrátane páuz medzi sériami cvičení. „C. Počet“ značí celkový počet vykonaných opakovaní (započítané sú aj opakovania cviku podpor na predlaktiach). Nasledujúce stĺpce označujú triedy cvikov viz 3.2 a celkový počet (pri podpore na predlaktiach celkový čas strávený v polohe podporu) opakovaní daných cvikov v rámci tréningu.

Takto vytvorená dátová sada obsahuje 7 tréningov od 4 rôznych osôb. Cvičiacimi osobami boli muži vo veku 20, 21, 22 a 23 rokov. Dve z týchto osôb (miro, marek) boli v silovo nadpriemernej kondícii. Ďalšie dve z týchto osôb (matej,martin) boli v silovo slabšej kondícii (v porovnaní s prechádzajúcimi osobami). Celkový čas zaznamenaných tréningov je 2 hodiny 18 minút a 59 sekúnd. Celkovo bolo vykonaných 1007 opakovaní jednotlivých cvikov.

V tabuľke 4.1 možno vidieť celkový počet zaznamenaných opakovaní pre jednotlivé triedy. V tabuľke 4.2 možno vidieť prehľad jednotlivých tréningov. Pre označenie pozície smartfónu v rámci vrecka používateľa navrhujem formát `ABC`, kde:

- `A` – značí vrecko v ktorom sa smartfón nachádzal. Možné hodnoty sú `l`-ľavé, `p`-pravé.
- `B` – značí pozíciu vrchnej časti displeja. Možné hodnoty sú `h`-hore, `d`-dole.
- `C` – značí natočenie displeja. Možné hodnoty sú `v`-von displej smeruje od nohy, `n`-noha displej smeruje k nohe.

Tréning `matej_utorok` bol zaznamenaný dvomi telefónmi. Druhý telefón však v strede tréningu z vrecka vypadol a bol doň vrátený s odlišnou (neznámou) orientáciou. Z tohto dôvodu boli v dátovej sade použité len údaje z prvého telefónu.

4.4 Finálna aplikácia

Aby bolo možné overiť funkčnosť navrhovaného postupu riešenia, je potrebné tento postup implementovať, vytrénovať klasifikačný algoritmus a následne celé riešenie vyhodnotiť

za pomoci nahrávok z dátovej sady. Navrhované riešenie som sa rozhodol implementovať v podobe desktopovej CLI[16] aplikácie. Táto aplikácia spracuje a vyhodnotí tréningovú nahrávku – jej výstupom bude prehľad vykonaných sérií cvičení spolu s počtom opakovaní cvikov v týchto sériách. Po spracovaní nahrávky bude na štandardný výstup vypísaný chronologický zoznam vykonaných sérií cvikov (typ cvičenia + počet opakovaní) spolu s agregovanou štatistikou celého tréningu (celkový počet opakovaní pre jednotlivé druhy cvikov). Okrem toho sa používateľovi zobrazí graf, ktorý bude zobrazovať priebeh zrýchlenia počas tréningu vo všetkých 3 osiach. V tomto grafe sa za pomoci farby pozadia vyznačia rozpoznané tréningové série tak, že farba pozadia série bude reprezentovať typ vykonávaného cviku. Jednotlivé rozpoznané opakovania sa v tomto grafe vyznačia za pomoci zmeny farby pozadia na čiernu v blízkom okolí miesta detekcie opakovania. Aplikáciu som sa rozhodol implementovať v jazyku Python3 vzhľadom k vhodnosti použitia tohto jazyku pre účely strojového učenia.

Finálna aplikácia pozostáva z dvoch častí. Prvou časťou je program reprezentujúci samotný klasifikátor – obsahuje všetky navrhované časti od predspracovania dát až po klasifikáciu. Tento program slúži pre tréning klasifikačného algoritmu (neurónovej siete) a následné uloženie vytrénovaných parametrov (váh jednotlivých synapsií) do súboru reprezentujúceho vytrénovaný model. Táto časť je reprezentovaná programom `classifier.py`. Druhá časť implementácie vychádza z tej prvej – klasifikátoru. Váhy neurónovej siete sú inicializované zo súboru reprezentujúceho vytrénovaný model. Okrem samotného klasifikačného postupu však táto časť ďalej implementuje aj navrhovaný segmentačný algoritmus a algoritmus pre počítanie opakovaní. Výsledky získané za pomoci týchto algoritmov sú prezentované používateľovi spôsobom popisovaným v predchádzajúcom odstavci. Táto časť je reprezentovaná programom `final.py`.

Na začiatku každého z týchto súborov sa nachádzajú premenné, za pomoci ktorých je možné nastaviť parametre modelu. Týmito parametrami sú napríklad typ extrahovaných príznakov, vlastnosti kĺzajúceho sa okna, zapnutie/vypnutie eliminácie vplyvu rotácie za pomoci PCA, zapnutie/vypnutie filtrácie za pomoci dolno-priepustného filtra prvého rádu a samotná frekvencia tohto filtra, parametre segmentačného algoritmu a parametre algoritmu počítania opakovaní. Testovanie modelu bude prebiehať tak, že sa zvolia parametre tohto modelu v súbore `classifier.py` a spustí sa tréning klasikátora spustením tohto programu. Po skončení tréningu sa zobrazí graf vývoja úspešnosti („accuracy“) klasifikátora zobrazujúci vývoj úspešnosti pri testovaní na tréningových a testovacích dátach. Zároveň sa na štandardný výstup vypíšu 2 takzvané matice zmätenia[17], získané validáciou na trénigových a testovacích dátach. Následne sa rovnaké parametre modelu nastavia aj v súbore `final.py` a spustením tohto programu sa analyzuje požadovaný tréning. Výsledky tohto programu sa manuálne porovnajú s očakávanými výsledkami (na základe známych anotácií) a vyhodnotí sa tak úspešnosť modelu pri analýze konkrétneho tréningu.

V priečinku s týmito dvoma programami sa nachádza priečinok s názvom `dataset`. Do tohto priečinku boli presunuté jednotlivé priečinky `splitted`[18] pričom každý z nich bol premenovaný tak, aby reprezentoval názov tréningu napr. `miro_piatok`. Tréningy použité pre tréning klasifikačného algoritmu a tréningy použité pre jeho testovanie možno nastaviť za pomoci premenných `exercise_train_prefixes` a `exercise_test_prefixes`.

[16] Command line interface.
[17] Z anglického „confusion matrix“.
[18] Popisované v sekcii 4.3.

Tieto premenné sú reprezentované listom obsahujúcim reťazce označujúce cestu k priečinku s tréningovými údajmi[19]. Položka tohto listu môže vyzerať napríklad nasledovne `"./dataset/miro_piatok/"`. Program `classifier` vypočíta pre každý tréning hodnotu kalibračnej transformácie (na základe cviku/cvikov uložených v podpriečinku `calib`). Následne načíta jednotlivé opakovania cvikov z tréningových priečinkov, pričom tieto dáta predspracuje. Ďalej sú z týchto cvikov extrahované príznaky. Pre výpočet týchto príznakov som použil funkcie z knižnice `numpy`. Príznaky z frekvenčného spektra (amplitúda jednotlivých frekvencií) boli vypočítané ako:

```
fft = np.abs(np.fft.fft(signal))
```

Extrahované príznaky sú následne použité pre tréning neurónovej siete MLP. Neurónovú sieť som implementoval za pomoci tried `Sequential` a `Dense` z knižnice `Keras`. Ako backend tejto knižnice som zvolil `TensorFlow`.

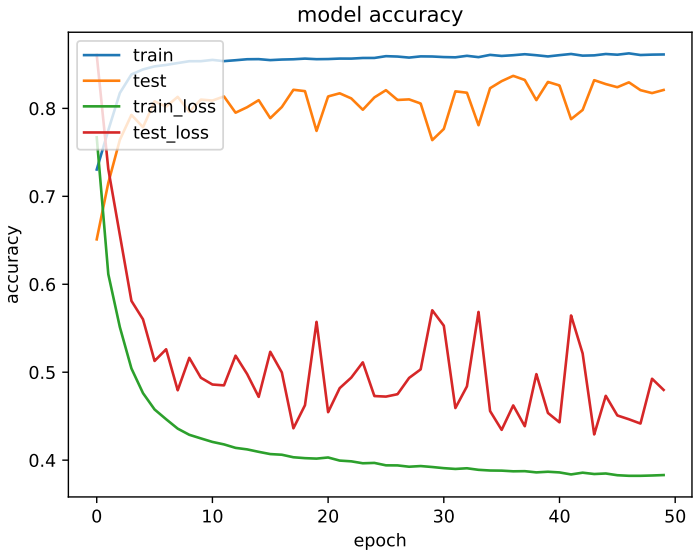
Trénovacie parametre tejto siete som zvolil nasledovne. Hodnotu „batch-size“[20] volím rovnú 8. Táto nízka hodnota sa síce prejaví na pomalšom tréningu siete, no vo všeobecnosti vedú nízke hodnoty k lepšej trénovacej stabilite a lepšej schopnosti výsledného modelu generalizovať viz [6]. Počet tréningových epóch[21] som zvolil na hodnotu 50. Na základe grafu vývoja úspešnosti (zobrazenom po ukončení tréningu) je možné vyhodnotiť či je potrebné tento počet zmeniť. Ako optimalizačný algoritmus som zvolil algoritmus `Adam`. Tento algoritmus narozdiel od algoritmu `SGD`[22] používa rôzne rýchlosti učenia – pre každý parameter siete vlastnú. Tieto rýchlosti sa v priebehu tréningu automaticky prispôsobujú potrebám siete. Nastavenia tohto algoritmu som ponechal na predvolených hodnotách[23] knižnice `Keras`. Váhy neurónovej siete sú po dokončení každej tréningovej epochy uložené do súboru s názvom `model.hdf5`. Veľkosť takto vzniknutého súboru je pri zvolení možnosti všetkých príznakov (viz sekcia 3.6) rovná 34.5kB. Na obrázku 4.4 možno vidieť graf vývoja úspešnosti klasifikátora v čase. Tento vývoj bol zaznamenaný pri tréningu so zvolenými parametrami, ktoré sú popísané v tomto odstavci. V prílohe B možno nájsť ukážku matíc zmätenia, ktoré boli na štandardný výstup vypísané po skončení tohto tréningu.

Po získaní vytrénovaného modelu reprezentovaného súborom `model.hdf5` je možné spustiť aplikáciu `final`, ktorá tento súbor k svojmu chodu potrebuje. Aplikácia `final` inicializuje váhy neurónovej siete z tohto súboru a načíta tréningovú nahrávku spolu s kalibračným cvikom. Signál nahrávky tréningu je predspracovaný a následne sú z neho extrahované príznaky. Tieto príznaky sú neurónovou sieťou klasifikované. Výsledky klasifikácie sú agregované do segmentov reprezentujúcich série cvičení. V týchto segmentoch sa spočítajú opakovania jednotlivých cvikov a výstup tejto analýzy tréningu sa zobrazí používateľovi.

Aplikáciu `final` možno spustiť bez alebo s dvomi argumentmi. Pokiaľ bola aplikácia spustená bez argumentov, tréning je načítaný z prednastavených názvov súborov. Tieto názvy sú definované na začiatku súboru `final.py`. V opačnom prípade budú použité argumenty značiť názvy tréningových súborov (súbor s nahrávkou tréningu, súbor s kalibračným cvikom). Ukážka použitia:

```
python ./final.py  training_file calibration_rep_file
```

[19] Priečinku, ktorý obsahuje vnorené priečinky napr. `pushups` obsahujúce jednotlivé opakovania cvikov.
[20] Počet tréningových vzoriek použití v jednom doprednom/spätnom kroku tréningu.
[21] Jedna tréningová epocha značí jedno prejdenie všetkých tréningových vzoriek touto sieťou.
[22] Stochastic gradient descent – veľmi často používaný optimalizačný algoritmus v sieťach MLP.
[23] Tieto predvolené hodnoty v knižnici `Keras` sú nastavené na hodnoty odporúčané autormi tohto algoritmu.



Obr. 4.4: **Graf vývoja úspešnosti klasifikátora v čase.** Tréning bol testovaný na nahrávke tréningu `miro_stvrtok`. Trénovaní bol na všetkých ostatných nahrávkach obsiahnutých v dátovej sade.

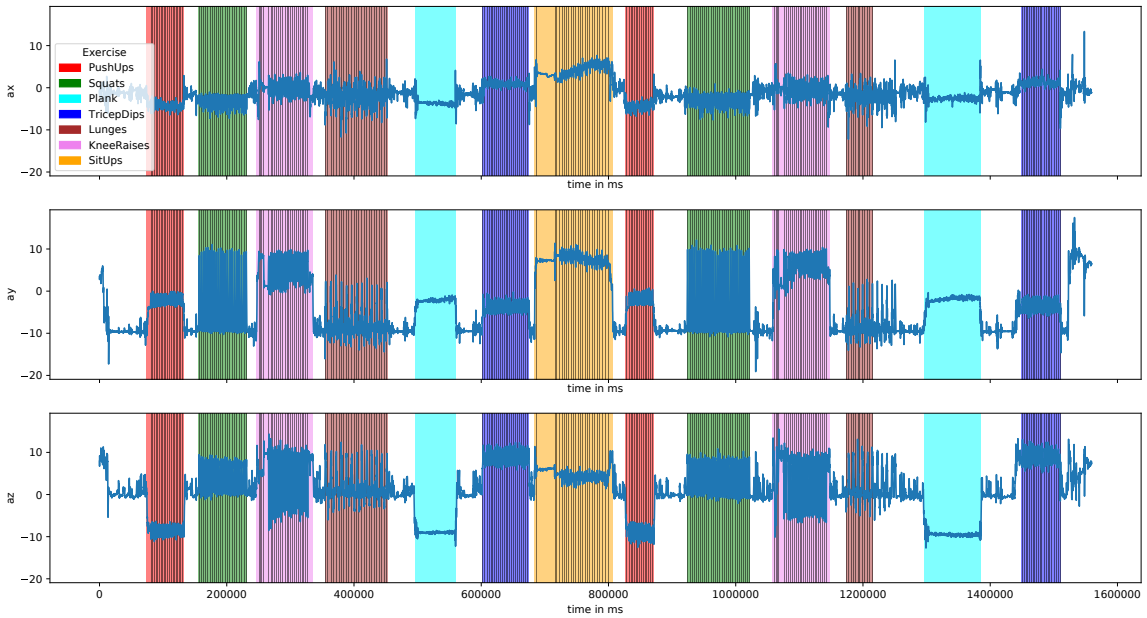
Textový výstup tejto aplikácie pri analýze nevideného tréningu od známeho[24] používateľa možno nájsť v prílohe B. Grafický výstup[25] aplikácie pri tejto analýze možno vidieť na obrázkoch 4.5 a 4.6.

4.5 Vyhodnotenie výsledkov

Implementované riešenie je potrebné vyhodnotiť, a získať tak spätnú väzbu o jeho vlastnostiach. Táto spätná väzba by mala poskytnúť dostatočné informácie o slabých ale aj silných stránkach navrhovaného riešenia. Vďaka týmto informáciám bude možné doladiť rozličné parametre jednotlivých algoritmov a adresovať prípadné problémy. Navrhované riešenie by malo byť schopné presne rozpoznávať a počítať opakovania cvičení s vlastnou váhou. Dáta z pohybových senzorov budú zbierané za pomoci smartfónu umiestneného v prednom vrecku používateľa – tento predpoklad by mal byť jediným predpokladom z ktorého riešenie vychádza. Riešenie by teda malo byť dostatočne robustné na to, aby dokázalo generalizovať na nevidených dátach od nevidených[26] používateľov, ako aj na to aby sa dokázalo vysporiadať s neznámou rotáciou zariadenia.

Implementované riešenie navrhujem vyhodnotiť za pomoci dvoch tréningov. Pomocou prvého overím, či je vôbec riešenie schopné generalizovať na nevidených dátach od známeho používateľa. Z dátovej sady vyberiem tréning tak, aby aj po odobratí tohto tréningu stále obsahovala aspoň jeden tréning od rovnakého používateľa. Tento zvolený tréning by mal byť vyhotovený s inou rotáciou zariadenia ako tréningy obsiahnuté vo zvyšku dátovej sady.

[24] Pre tréning klasifikátoru boli použité medzi inými aj nahrávky tréningov patriace tomuto používateľovi.
[25] Matplotlib graf zobrazený používateľovi.
[26] Takých používateľov od ktorých model „nevidel“ ani jedinú nahrávku tréningu.



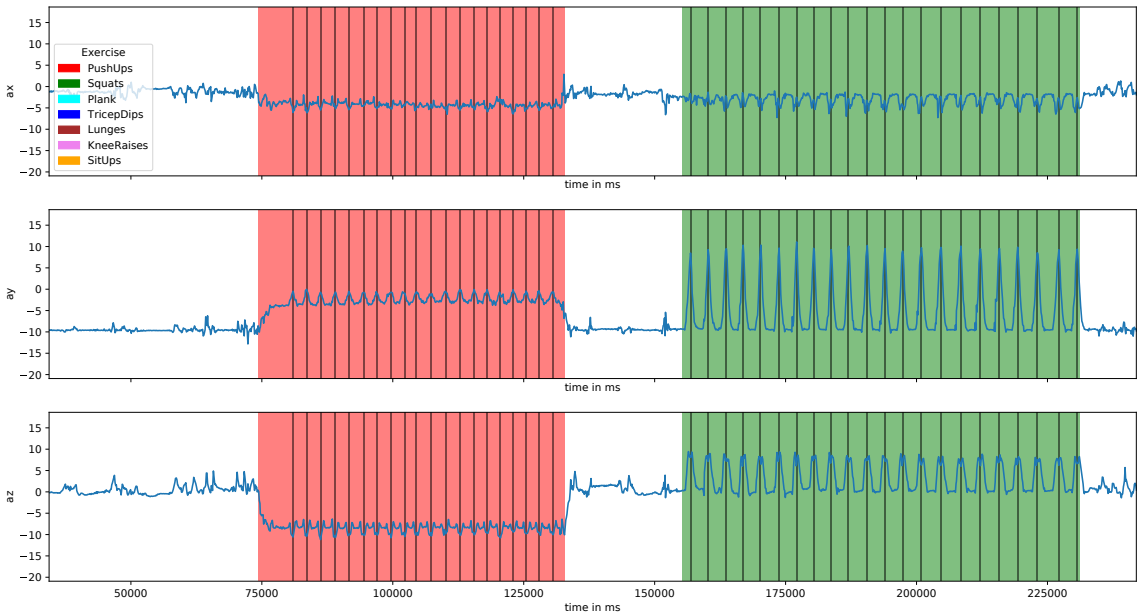
Obr. 4.5: **Ukážka grafického výstupu aplikácie `final` pri analýze tréningu `miro_stvrtok`**. Pre tréning modelu boli použité všetky tréningy z dátovej sady okrem tréningu `miro_stvrtok`.

Týmto spôsobom bude zaručené overenie nezávislosti riešenia od rotácie zariadenia. Za pomoci tohto tréningu možno overiť aj vplyv výberu príznakov na úspešnosť klasifikácie. Rovnako možno overiť správanie modelu pri vynechaní kroku eliminácie vplyvu rotácie za pomoci PCA. Druhým tréningom pre overenie riešenia by mal byť tréning od nevideného používateľa. Na tomto tréningu možno overiť schopnosť generalizácie riešenia na dátach od neznámych používateľov. Potrebné je aj navrhnúť spôsob ako sa bude vyhodnocovať úspešnosť riešenia a aké metriky budú k tomuto vyhodnoteniu použité.

Úspešnosť klasifikátora možno vyhodnotiť samostatne. Rovnako možno samostatne vyhodnotiť aj úspešnosť algoritmu pre počítanie opakovaní – ako vstup by boli použité vystrihnuté tréningové série za pomoci známych anotácií[27]. Vzhľadom na cieľ práce však dáva zmysel vyhodnotiť riešenie ako celok, a získať tak informáciu o tom, ako dobre toto riešenie dokáže analyzovať tréningy. Mnoho autorov pri vyhodnocovaní úspešnosti počítania opakovaní zvolilo jednoduchý a málo výpovedný spôsob analýzy úspešnosti riešenia. Títo autori jednoducho porovnávajú výsledný počet opakovaní jednotlivých cvikov s ich očakávaným počtom a vyjadria tak percentuálnu chybu. Takýto prístup má však vážny nedostatok. Pokiaľ sa napríklad algoritmus „pomýli“ a zamení si sériu 10 kľukov so sériou 10 drepov, z pohľadu celkové počtu vykonaných opakovaní sa bude tréning javiť ako úspešný. Rovnako pokiaľ v jednej sérii kľukov spočíta o 2 opakovania menej a v ďalšej o 2 opakovania viac výsledok sa bude javiť ako 100 % úspešný. V realite má však tento algoritmus oveľa nižšiu úspešnosť. Z tohto dôvodu navrhujem robustnejšiu metriku ako manuálne a spoľahlivo overiť úspešnosť riešenia. Opakovania cvičení možno rozdeliť do 3 kategórií:

- TP – „True Positive“ detekované opakovania, ktoré sa naozaj udiali

[27] Každý tréning obsahuje okrem anotácií označujúcich jednotlivé opakovania, aj anotácie, ktoré ohraničujú celé série cvičení.



Obr. 4.6: **Ukážka vyznačenia rozpoznaných opakovaní.** Na obrázku sa nachádza priblížený graf z obrázku 4.5.

- FP – „False Positive“ detekované opakovania, ktoré sa v tréningu neodohrali
- FN – „False Negative“ opakovania, ktoré sa v tréningu odohrali no algoritmus ich nedetekoval

Z týchto metrík možno odvodiť ďalšie metriky a to presnosť („precision“) a úplnosť („recall“). Metrika presnosť určuje koľko percent z detekovaných opakovaní tvoria opakovania, ktoré sa reálne udiali. Metrika úplnosť značí koľko percent z opakovaní, ktoré sa udiali bolo aj detekovaných. Z metrík presnosť a úplnosť možno odvodiť metriku F1-skóre, ktorá je ich harmonickým priemerom. Spôsob výpočtu jednotlivých metrík možno vidieť v rovniciach 4.1, 4.2 a 4.3.

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

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

$$\text{f1\_skore} = 2 \cdot \frac{\text{presnost} \cdot \text{uplnost}}{\text{presnost} + \text{uplnost}} \tag{4.3}$$

Pre vyhodnotenie riešenia na neviedených dátach od videného používateľa som zvolil tréning `miro_stvrtok`. Ako vidno z tabuľky 4.2 v dátovej sade neexistuje žiaden iný tréning s rovnakou rotáciou zariadenia. Pri tréningu klasifikátora som odskúšal všetky 3 varianty voľby príznakov popisované v sekcii 3.6. Na základe výsledných matíc zmätenia, ktoré uvádzam v prílohe B možno konštatovať, že pre maximalizáciu úspešnosti klasifikátora je najvhodnejšie použiť mód extrakcie všetkých príznakov. Ďalej boli odskúšané všetky 3 varianty voľby príznakov pričom bol vynechaný krok eliminácie vplyvu rotácie za pomoci PCA. Tento experiment dopadol podľa očakávaní – klasifikátor v režimoch všetkých a žiadnych príznakov nebol schopný úspešne klasifikovať jednotlivé vzorky tréningu. Väčšina vzoriek

bola označená ako voľný pohyb – tu sa prejavila aj žiaduca vlasnosť klasikátora a to, že pokiaľ si nie je vzorkou istý, má tendenciu ju zaradiť do triedy voľný pohyb.

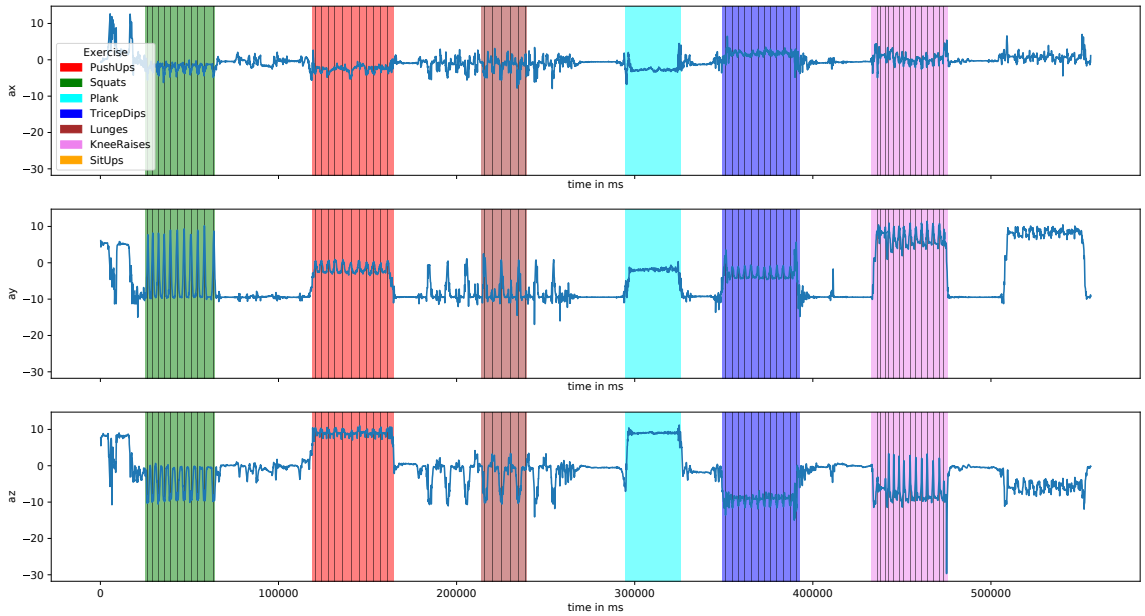
Ukázalo sa, že pri použití iba frekvenčných príznakov je klasifikátor schopný úspešne klasifikovať aj bez použitia kroku pre elimináciu vplyvu rotácie za pomoci PCA. Táto vlastnosť je spôsobená tým, že smartfón nie je vo vrecku otočený úplne náhodne. Vo všeobecnosti možno konštatovať, že sa môže nachádzať v 4 možných pozíciách. Vrch displeja môže smerovať dole alebo hore. Plocha displeja môže byť natočená k nohe alebo od nej. Vzhľadom na osi pohybových senzorov viz obrázok 2.1 to znamená, že rozloženie zrýchlenia medzi jednotlivé osi bude vo všetkých 4 možných pozíciách rovnaké. Frekvenčnými príznakmi sú amplitúdy jednotlivých frekvencií. Pokiaľ ostane rozloženie zrýchlenia medzi jednotlivé osi rovnaké, amplitúdy zostanú rovnaké tiež, nezávisle na tom či napríklad os y smeruje kolmo k zemi alebo kolmo nahor. V reálnych situáciach sa však zariadenie vo vrecku používateľa môže nachádzať vo viacerých než týchto „ideálnych“ pozíciách a preto navrhujem použiť krok eliminácie vplyvu rotácie aj v prípade, že by boli zvolené iba frekvenčné príznaky.

Metrika	**0**	**1**	**2**	**3**	**4**	**5**	**6**
TP	34	52	139	42	37	44	20
FP	1	0	13	3	0	8	4
FN	0	0	0	0	9	0	1
Precision	0.97	1.00	0.91	0.93	1.00	0.85	0.83
Recall	1.00	1.00	1.00	1.00	0.80	1.00	0.95
F1	0.98	1.00	0.95	0.96	0.88	0.92	0.89

Tabuľka 4.3: **Vyhodnotenie metrík počítania opakovaní pri analýze nevideného tréningu známeho používateľa.** Tréning možno vidieť na obrázkoch 4.5 a 4.6. Stĺpce značia čísla tried. Hodnoty v riadkoch TP,FP a FN značia počty opakovaní (v stĺpci 2 označujúcom triedu podpor na predlaktiach označujú hodnoty čas v sekundách). Priemerné hodnoty jednotlivých metrík sú: **presnosť=92.7 %**, **úplnosť=96.4 %** a **F1-skóre=94.0 %**.

Schopnosť generalizácie na nevidených dátach videného používateľa som overil s použitím klasifikátora v režime extrakcie všetkých príznakov. Priebeh úspešnosti tohto klasifikátora počas jeho tréningu možno vidieť na obrázku 4.4. Počet trénovacích epóch s hodnotou 50 sa prejavil ako vhodne zvolený. Klasifikátor neprejavoval známky pretrénovania, takže nebolo potrebné tento problém riešiť. Overenie generalizácie prebiehalo tak, že som spustil analýzu nevideného tréningu programom `final` a manuálne som spočítal metriky pre jednotlivé triedy cvičení. Vzniknuté metriky možno vidieť v tabuľke 4.3. Z týchto metrík som odvodil výslednú metriku F1-skóre, ktorá je v tomto prípade rovná 94 %. Tento výsledok značí, že navrhované riešenie dokáže úspešne generalizovať na nevidených dátach známeho používateľa.

V práci navrhované hodnoty jednotlivých parametrov segmentačného algoritmu sa prejavili ako vhodne zvolené. Odskúšané boli aj iné hodnoty týchto parametrov, viedli však k zhoršeniu výsledkov. Ako parameter minimálneho času trvania jednej série vyjadrenom v počte opakovaní viz sekcia 3.8 som zvolil hodnotu 5, ktorá sa taktiež prejavila ako vhodná voľba. Minimálny čas trvania jednotlivých cvičení (potrebný pre správnu funkčnosť algoritmu počítania opakovaní) bol zvolený na základe anotácií z dátovej sady na približne 70 % z najkratšieho trvania cviku.



Obr. 4.7: **Ukážka analýzy nevideného tréningu od neznámeho používateľa**. Obrázok zobrazuje vyhodnotenie modelu na nahrávke tréningu od používateľa, ktorého tréningy model nevidel. Model bol vytrénovaný na celej dátovej sade.

Za účelom overenia schopnosti generalizácie riešenia na nevidených dátach od nevideného používateľa som klasifikátor ešte raz vytrénoval, tento krát na všetkých tréningoch obsiahnutých v dátovej sade. Nevidenému používateľovi (mužovi vo veku 24 rokov) bola za pomoci video-hovoru[28] vysvetlená technika vykonávania cvikov. Následne tento používateľ vykonal a nahral tréning tak, aby bolo vykonaná jedna cvičebná séria z každého druhu cvičenia. Prvú sériu tvorili drepy. Z tejto série som manuálne extrahoval kalibračný cvik – jedno opakovanie drepu. Tento tréning som za pomoci programu `final` analyzoval ako vidno na obrázku 4.7 a následne v ňom manuálne spočítal jednotlivé metriky. Tieto metriky možno vidieť v tabuľke 4.4. Úspešnosť samotného klasifikačného algoritmu na tomto tréningu nebolo možné vyhodnotiť, keďže k tomuto tréningu nebola z vyššie uvedených dôvodov vytvorená anotácia pomocou aplikácie `Anotator`. Výsledná hodnota metriky F1-skóre celého riešenia overená na tréningu nevideného používateľa je 74.9 %. Na základe tejto metriky a vizuálneho výsledku, ktorý možno vidieť na obrázku 4.7 možno skonštatovať, že model dokáže čiastočne generalizovať na nevidených používateľoch.

Keďže sú však výsledné váhy klasifikačného algoritmu po jeho tréningu za každým iné (aj pri rovnakých parametroch tréningu – spôsobené je to ich prvotnou inicializáciou na náhodnú hodnotu), rozhodol som sa klasifikačný algoritmus vytrénovať rovnakým spôsobom ešte 9 krát. Zakaždým som overil schopnosť generalizácie výsledného riešenia na tréningu nevideného používateľa. Najhoršia dosiahnutá metrika F1-skóre bola 45.3 %. Z tohto dôvodu možno skonštatovať, že finálne riešenie trénované na malej dátovej sade, ktorá vznikla v tejto práci, nie je schopné stabilne (replikovatelne) dobre generalizovať na dátach nevidených používateľov. Tento fakt môže byť spôsobený viacerými faktormi. Dátová sada je malá a zároveň obsahuje tréningy od malého množstva rozličných používateľov. Preto nemusí

[28] Práca bola písaná počas koronavírusovej pandémie v roku 2020.

Metrika	0	1	2	3	4	5	6
TP	11	11	27	10	6	10	0
FP	0	0	4	2	0	4	0
FN	0	0	0	0	9	0	10
Precision	1.00	1.00	0.87	0.83	1.00	0.71	0.00
Recall	1.00	1.00	1.00	1.00	0.40	1.00	0.00
F1	1.00	1.00	0.93	0.91	0.57	0.83	0.00

Tabuľka 4.4: **Vyhodnotenie metrík počítania opakovaní pri analýze nevideného tréningu neznámeho používateľa.** Tréning možno vidieť na obrázku 4.7. Stĺpce značia čísla tried. Hodnoty v riadkoch TP,FP a FN značia počty opakovaní (v stĺpci 2 označujúcom triedu podpor na predlaktiach označujú hodnoty čas v sekundách). Priemerné hodnoty jednotlivých metrík sú: **presnosť=77.3 %**, **úplnosť=77.1 %** a **F1-skóre=74.9 %**.

byť dostatočná na to, aby dokázala dostatočne abstraktne reprezentovať jednotlivé triedy cvičení.

Ďalším problémom je aj nerovnomerné vzorkovanie. V závere tejto práce sa mi podarilo objaviť upravenú verziu knižnice `cordova-plugin-device-motion`, ktorá volá metódy Android-ového „sensors framework-u“ s hodnotou parametru delay rovnou `SENSOR_DELAY_GAME`. Vďaka tomuto môže framework Cordova získavať vzorky s periódou 20ms. Pri volaní funkcií z Cordova frameworku je teda možné získavať vzorky s periódou 100ms oveľa presnejšie v porovnaní s predchádzajúcim riešením, kde Android poskytoval vzorky s periódou 60ms (keďže 100 nie je celočíselným násobkom čísla 60, viedol tento prístup k nerovnomernému vzorkovaniu). Tréning nevideného používateľa bol získaný za pomoci novej verzie aplikácie, ktorá používala túto novú knižnicu. Tento fakt mal pravdepodobne tiež vplyv na schopnosť generalizácie na tomto tréningu. Zdrojové kódy aplikácie na priloženom médiu obsahujú novú verziu aplikácie `Akcelerak`, ktorá využíva upravenú verziu knižnice `cordova-plugin-device-motion` dostupnú z:
https://github.com/bayhall-digital/cordova-plugin-device-motion-fast.

V priebehu testovania finálneho riešenia na rozličných dátach som zistil, že toto riešenie máva niekedy problém s rozoznávaním cviku výpady. Tento problém môže byť spôsobený práve vyšším vnútrotriednym rozptylom, keďže opakovanie výpadu môže mať v porovnaní s inými cvikmi dve rôzne podoby. Tento problém by bolo možné riešiť tým, že by sa trieda výpady rozdelila na dve triedy – triedu výpad kročnou nohou a triedu výpad zadnou nohou, podľa toho vo vrecku ktorej nohy sa smartfón pri vykonávaní opakovania nachádzal. Segmentačný algoritmus by tieto triedy považoval za jednu triedu konkrétneho cviku. Ďalej sa pri niektorých tréningoch preukázal aj očakávaný problém, ktorý som načrtol v sekcii 3.8. Triedy kľuky a podpor na predlaktiach boli medzi sebou zamieňané. Tento problém by šiel vyriešiť s pomocou využitia algoritmu pre počítanie opakovaní. Pri detekcii segmentu s triedou podpor na predlaktiach by sa neodmeral čas tohto segmentu (tak ako sa to robí teraz) no spočítali by sa v ňom opakovania kľukov. Ak by sa v tomto segmente podarilo nájsť dostatočný počet opakovaní, znamenalo by to, že tento segment je pravdepodobne sériou kľukov, ktorá bola mylne označená za podpor na predlaktiach. Rovnako by sa pri sériách kľukov po spočítaní ich počtu porovnal tento počet s „očakávaným“ počtom, a pokiaľ by bol výrazne nižší, tak by to znamenalo, že sa jedná pravdepodobne o sériu cviku podpor na predlaktiach, ktorá bola mylne označená za kľuky. Riešenie by ďalej išlo signifikan-

tne vylepšiť za pomoci vytvorenie väčšej dátovej sady od viac používateľov s pravidelnejším vzorkovaním – obsahovala by tréningy nahraté novou verziou aplikácie `Akcelerak`. Následne by bolo možné vylepšiť aj tréning klasifikátora tak, aby z každej triedy videl počas tréningu rovnaký počet vzoriek.

Výsledné navrhované riešenie je možné implementovať aj ako android aplikáciu. Navrhovaný algoritmus možno implementovať aj tak, aby bol schopný vyhodnocovať tréning počas jeho priebehu v reálnom čase. Počet opakovaní v sérii by bol však vyhodnotený až po jej skončení (vzhľadom na spôsob akým je kvôli robustnosti riešenia navrhnutý algoritmus počítania opakovaní). Riešenie v reálnom čase však nie je potrebné, pretože sa smartfón počas celého tréningu nachádza vo vrecku používateľa. Po skončení tréningu teda môže byť tento tréning vyhodnotený aj offline – riešenie v reálnom čase by v prípade analýzy zloženia[29] tréningu preto neprinieslo žiadnu pridanú hodnotu.

Na základe získaných výsledkov zodpovedám na otázky, ktoré si táto práca kládla:

- „Je možné použiť dáta z akcelerometru získané zo smartfónu, ktorý je umiestnený vo vrecku používateľa k plne automatizovanej analýze tréningu s vlastnou váhou? (Analýza zahŕňa klasifikáciu aj počítanie opakovaní týchto cvičení.)“
  Áno, dáta z akcelerometru získané popisovaným spôsobom sa ukázali ako dostatočné pre riešenie tohto problému.

- „Je možné toto riešenie implementovať tak, aby nebola potrebná znalosť umiestnenia (rotácie) zariadenia vo vrecku používateľa?“
  Áno, použitím kalibračného cviku je možné eliminovať vplyv rotácie. Pokiaľ informácia o takomto kalibračnom cviku nie je dostupná, je možné použiť mód klasifikátora v ktorom sú použité iba frekvenčné príznaky. Treba však brať do úvahy, že zariadenie v tomto prípade nemôže byť úplne ľubovoľne rotované.

[29] Typov cvičení v jednotlivých vykonaných sériách a počtu vykonaných opakovaní v nich.

Kapitola 5

Záver

Cieľom práce bol návrh a implementácia riešenia pre automatickú klasifikáciu a počítanie opakovaní 7 základných cvikov s vlastnou váhou tela s využitím dát z pohybových senzorov. Tieto dáta boli získané z mobilného zariadenia umiestneného v hornom vrecku nohavíc cvičiaceho používateľa.

Práca na začiatku oboznámi čitateľa s prehľadom niekoľkých existujúcich prác, vedeckých článkov a dátových sád. Následne ho oboznámi so základnými metódami, ktoré budú v práci použité. Text práce pokračuje analýzou problému, ktorý táto práca rieši a návrhom jeho riešenia. Najprv je navrhnutý celkový proces riešenia a následne sú navrhnuté jednotlivé časti z ktorých pozostáva. Práca ďalej popisuje realizáciu týchto čiastočných krokov a samotné finálne riešenie. Text práce končí overením tohto riešenia na dátovej sade, ktorá vznikla v rámci tejto práce a vyhodnotením dosiahnutých výsledkov. Na základe získaných výsledkov sú navrhnuté možnosti vylepšenia riešenia navrhovaného v tejto práci.

V rámci riešenia práce vznikli 2 mobilné Android aplikácie slúžiace pre zber a anotáciu dát z pohybových senzorov smartfónu. Spolu s nimi vzniklo aj 6 nástrojov pre prácu s týmito dátami a ich vizualizáciu. Za pomoci týchto aplikácií a nástrojov bola vytvorená dátová sada obsahujúca nahrávky 7 tréningov s vlastnou váhou tela od 4 rôznych používateľov. Celková dĺžka týchto tréningov je 2 hodiny 18 minút a 59 sekúnd, pričom bolo dokopy vykonaných 1007 opakovaní skúmaných cvičení. Finálne riešenie implementované v podobe desktopovej CLI aplikácie bolo schopné dosiahnuť pri analýze a počítaní opakovaní tréningu nevideného používateľa metriku F1-skóre v rozmedzí 45.3 %-74.9 %. Pri nevidenom tréningu známeho používateľa bolo riešenie schopné dosiahnuť hodnotu metriky F1-skóre až 94 %.

Ukázalo sa, že údaje z akcelerometru smartfónu, ktorý je umiestnený v hornom vrecku používateľa je možné úspešne použiť k automatickej analýze tréningu a počítaniu opakovaní. Ďalej sa ukázalo, že za pomoci použitia kalibračného cviku je možné dosiahnuť nezávislosť riešenia od rotácie smartfónu vo vrecku používateľa.

V práci možno ďalej pokračovať vytvorením väčšej dátovej sady, ktorá bude obsahovať tréningy širšieho spektra užívateľov. Vďaka tomu bude model na nej trénovaný schopný oveľa lepšie generalizovať na dátach od nevidených užívateľov. Finálne riešenie je možné implementovať aj v podobe Android aplikácie, ktorá by vykonávala analýzu v reálnom čase. Nahrávaciu aplikáciu je možné rozšíriť o automatické zaznamenanie kalibračného cviku. Túto aplikáciu je ďalej možné rozšíriť o automatické zaslanie nahrávky za účelom jej analýzy tréningu na server, na ktorom by bežala desktopová implementácia realizovaná v tejto práci.

Týmto spôsobom by bolo možné zaručiť automatickú výmenu klasifikačného modelu za novšiu verziu bez nutnosti používateľov aktualizovať ich aplikácie.

Nahrávacie aplikácie a nástroje pre prácu so vzniknutými dátami je možné použiť ako základ pre vytvorenie dátovej sady pre akýkoľvek iný projekt pracujúci s dátami z pohybových senzorov Android-ového smartfónu. Navrhované riešenie možno s menšími úpravami taktiež použiť aj na riešenie iných podobných problémov. Samotný algoritmus počítania opakovaní vracia okrem počtu opakovaní vo vykonanej sérii cvičenia aj ich pozíciu v rámci tejto série. Túto vlastnosť možno využiť napríklad k odhadu kondície používateľa na základe rozdielov času medzi jednotlivými opakovaniami – predlžujúce sa rozdiely môžu indikovať fyzické vyčerpanie používateľa.

Literatúra

[1] Bishop, C. M. *Pattern Recognition and Machine Learning*. 1. vyd. Springer-Verlag New York, 2006. ISBN 0-387-31073-8.

[2] Brigham, E. O. *The Fast Fourier Transform and Its Applications*. USA: Prentice-Hall, Inc., 1988. ISBN 0-13-307505-2.

[3] Haykin, S. S. *Neural Networks and Learning Machines*. 3. vyd. Pearson Education Upper Saddle River, 2009. ISBN 0-13-147139-2.

[4] Jašurková, M. *Základy spracovania digitálnych signálov - Diskrétna Fourierova transformácia* [online]. 2014. [cit. 2020-02-10]. Bakalárska práca. Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky. Dostupné z: http://www.iam.fmph.uniba.sk/efm/bakalarky/2014/jasurkova/bakalarka.pdf.

[5] Khan, A., Siddiqi, M. a Lee, S.-W. Exploratory Data Analysis of Acceleration Signals to Select Light-Weight and Accurate Features for Real-Time Activity Recognition on Smartphones. *Sensors (Basel, Switzerland)*. Október 2013, zv. 13, s. 13099–13122. DOI: 10.3390/s131013099.

[6] Masters, D. a Luschi, C. Revisiting Small Batch Training for Deep Neural Networks. *ArXiv*. Apríl 2018, abs/1804.07612.

[7] Morris, D., Saponas, T., Guillory, A. a Kelner, I. RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises. *Conference on Human Factors in Computing Systems - Proceedings*. Apríl 2014. DOI: 10.1145/2556288.2557116.

[8] Nilsson, M. a Wilén, H. *Push-up Tracking through Smartphone Sensors* [online]. 2016. [cit. 2020-01-28]. Diplomová práca. KTH Royal Institute of Technology - School of Computer Science and Communication. Dostupné z: https://pdfs.semanticscholar.org/9df2/81b745cde1bc0390f6acc12d4db32144986a.pdf.

[9] Nwankpa, C., Ijomah, W., Gachagan, A. a Marshall, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. November 2018.

[10] Oppenheim, A. V., Schafer, R. W. a Buck, J. R. *Discrete-Time Signal Processing*. Second. Prentice-hall Englewood Cliffs, 1999. ISBN 0-13-754920-2.

[11] Shlens, J. A Tutorial on Principal Component Analysis. *Educational*. Apríl 2014, zv. 51.

[12] Siirtola, P. a Röning, J. Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data. *International Journal of Interactive Multimedia and Artificial Intelligence*. Jún 2012, zv. 1, s. 38–45. DOI: 10.9781/ijimai.2012.155.

[13] Soro, A., Brunner, G., Tanner, S. a Wattenhofer, R. Recognition and repetition counting for complex physical exercises with deep learning. *Sensors*. Multidisciplinary Digital Publishing Institute. 2019, zv. 19, č. 3, s. 714.

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. Jún 2014, zv. 15, s. 1929–1958.

[15] Twomey, N., Diethe, T., Fafoutis, X., Elsts, A., McConville, R. et al. A Comprehensive Study of Activity Recognition Using Accelerometers. *Informatics*. Máj 2018, zv. 5, s. 27. DOI: 10.3390/informatics5020027.

[16] Yurtman, A. a Barshan, B. Activity Recognition Invariant to Sensor Orientation with Wearable Motion Sensors. *Sensors*. August 2017, zv. 17. DOI: 10.3390/s17081838.

[17] Šustr, M. *Counting repetitions of exercises from body worn sensors* [online]. 2015. [cit. 2020-01-15]. Diplomová práca. Czech Technical University in Prague. Dostupné z: http://michal.sustr.sk/DP.pdf.

Príloha A

Obsah priloženého pamäťového média

Priložené médium (DVD) obsahuje nasledujúce priečinky a súbory:

- `android_apps` – Priečinok obsahujúci zdrojové súbory Android aplikácií, ako aj ich preložené verzie.
- `app` – Priečinok obsahujúci kód finálnej desktopovej aplikácie.
- `dataset` – Priečinok obsahujúci vytvorenú dátovú sadu.
- `models` – Priečinok obsahujúci jednotlivé modely tj. váhy neurónovej siete, ktoré boli použité pri vyhodnotení riešenia.
- `poster` – Priečinok obsahujúci plagát vo formáte PDF, ako aj zdrojové kódy k jeho prekladu v systéme LaTeX.
- `projekt.pdf` – Súbor obsahujúci text bakalárskej práce.
- `python_env` – Priečinok obsahujúci izolované virtuálne prostredie Python3.6.3 v ktorom sú predinštalované všetky potrebné knižnice pre beh finálnej aplikácie.
- `README` – Súbor obsahujúci popis súborov nachádzajúcich sa na pamäťovom médiu.
- `tools` – Priečinok obsahujúci zdrojové kódy ako aj preložené nástroje pre prácu s dátami a vizualizáciu.
- `tz` – Priečinok obsahujúci zdrojové kódy tejto technickej dokumentácie.
- `validation_trainings` – Priečinok obsahujúci tréningy, ktoré boli použité pre vyhodnotenie riešenia.
- `video` – Priečinok obsahujúci video prezentujúce riešenie v krátkej a dlhej verzii.

V jednotlivých priečinkoch sa v prípade potreby nachádzajú súbory `README` popisujúce ich obsah/použitie.

Príloha B

Ukážka výstupov aplikácií

Ukážka výstupu aplikácie `final` pri analýze tréningu `miro_stvrtok`. Model bol trénovaný na všetkých tréningoch z dátovej sady s vynechaním tohto analyzovaného tréningu.

```
##################################################
Training consisted of these exercise series:

PushUps with: 20 reps
Squats with: 22 reps
KneeRaises with: 25 reps
Lunges with: 25 reps
Plank with time: 64.129 seconds.
TricepDips with: 24 reps
SitUps with: 24 reps
PushUps with: 15 reps
Squats with: 30 reps
KneeRaises with: 27 reps
Lunges with: 12 reps
Plank with time: 88.768 seconds.
TricepDips with: 21 reps
--------------------------------------------------
+--------------------------------+
|  Aggregated training results:  |
+--------------------------------+
|  Exercise  |   Count of reps   |
+------------+-------------------+
| PushUps    |           35 reps |
| Squats     |           52 reps |
| Plank      |       152 seconds |
| TricepDips |           45 reps |
| Lunges     |           37 reps |
| KneeRaises |           52 reps |
| SitUps     |           24 reps |
+------------+-------------------+
##################################################
```

Ukážka textového výstupu programu `classifier` po dokončení tréningu pri použití všetkých(režim `AllFeatures`) príznakov. Klasifikátor bol testovaný na tréningu `miro_stvrtok` a trénovaný na zvyšku dátovej sady. Zobrazené sú matice zmätenia, ktoré vznikli testovaním vytrénovaného klasifikátora na tréningových a testovacích dátach. Jednotlivé stĺpce označujú predpovedané triedy, riadky označujú skutočné triedy. Uvediem príklad na prvom riadku druhej matice. Z tohto riadku možno vyčítať, že testovacie dáta obsahovali 121 vzoriek prislúchajúcich nultej triede. Pri klasifikácii bolo 120 z nich zaradených do nultej triedy a jedna vzorka bola zaradená do siedmej triedy. Jednotlivé triedy prislúchajú indexu výstupného neurónu. Tieto indexy možno previesť na názvy cvičení za pomoci tabuľky 3.2.

```
Confusion matrix - validated on Train set:

[[  797     0   114     8     3     0     0    20]
 [    0  1274     0    19   101     1     5   237]
 [   13     0  2091     1     2     0     6     7]
 [   14     7     8   829     5     0    49    88]
 [    0   208     0    70   633     0     0  1051]
 [   29    19    16    29     3   491    67   226]
 [    3     1   153    16     0     2  1579   110]
 [  111   214   142   377   441    93   373 19876]]

Confusion matrix - validated on Test set:

[[ 120    0    0    0    0    0    0    1]
 [   0  355    0    0    2    0    0    1]
 [  48    0  513    0    0    0    0    0]
 [   1    0    0  194    0    0    0    3]
 [   0   40    0    0  260    0    0  100]
 [   0    6    0    1    0  212    1    5]
 [   0    0    0    0    0    4  131   40]
 [  30   32   14   75   90   25  284 1897]]
```

Nasledujúce ukážky sú ukážkami úspešnosti rovnakého tréningu klasifikátora pri zvolení rôznych režimov extrakcie príznakov. Režim `NoFeatures`:

```
Confusion matrix - validated on Train set:

[[  584     0   268    23     2     4    16    45]
 [    0  1267     0     8   137     1     5   219]
 [   63     0  2019     2     0     0     3    33]
 [    7     5    21   731     6     2    39   189]
 [    0   112     3    16   814     6     8  1003]
 [   13     8    20    35    18   491   110   185]
 [   20     4    70     8     0     2  1555   205]
 [  100   131   145   197   445    52   419 20138]]

Confusion matrix - validated on Test set:

[[   5    0  106    7    0    0    0    3]
 [   0  344    0    0   14    0    0    0]
 [   1    0  559    1    0    0    0    0]
 [   0    0    0  101   15    0    0   82]
 [   0   56    2    7  110    0    0  225]
 [   0   17    0    3    3  182   10   10]
 [   0    0    0    0    0    1   11  163]
 [   3   38   40   82   80    6  127 2071]]
```

Režim `FFTFeaturesOnly`:

```
Confusion matrix - validated on Train set:

[[  791     0    84     9     0     0     5    53]
 [    3  1268     0    35    21    11     1   298]
 [   15     0  2074     0     0     0     8    23]
 [   50    10     1   693     0     1    39   206]
 [    0   147     0    89   280     5    25  1416]
 [   55    17    17    32     1   378    65   315]
 [    8     0    78    43     0     1  1332   402]
 [   57   159    83   222   119    27   368 20592]]

Confusion matrix - validated on Test set:

[[ 118    0    0    1    0    0    0    2]
 [   0  348    0    0    0    0    0   10]
 [   5    0  556    0    0    0    0    0]
 [   4    0    1  168    0    0    0   25]
 [   0   59    0   11  131    0    1  198]
 [   0   15    0    0    1  181    0   28]
 [   7    1    6    0    0    0   51  110]
 [  30   26   14   31   15   10  134 2187]]
```

Režim `NoFeatures` bez použitia eliminácie vplyvu rotácie za pomoci PCA:

```
Confusion matrix - validated on Train set:

[[  802     9    83    12     0     0     4    32]
 [    9  1104     1    29    72     4     0   418]
 [   45     1  2070     0     0     0     0     4]
 [    0    20     0   857     0    17    46    60]
 [    5    92    18    29   466     2     0  1350]
 [    4    19     1    38     3   472   139   204]
 [    0     0     0     3     0    25  1746    90]
 [   50   132    77   251    90    84   452 20491]]

Confusion matrix - validated on Test set:

[[   0    0    0    0  109    0    0   12]
 [   2  294    0    6    0    7    5   44]
 [   0    0    0  167  394    0    0    0]
 [   0    1   59    0    0    0    0  138]
 [   9  153    0    4    5    7   12  210]
 [   4   59    0    0   34    0    0  128]
 [   0    0    0    0    0    0  134   41]
 [  19   66   28    9   77    3   68 2177]]
```

Režim `FFTFeaturesOnly` bez použitia eliminácie vplyvu rotácie za pomoci PCA:

```
Confusion matrix - validated on Train set:

[[  793     1    41    26    11     7    23    40]
 [    3  1127     0     3    89    54    12   349]
 [   18     0  2092     0     0     1     0     9]
 [   60     5    55   679     0    19    66   116]
 [   54   126     1    14   611    36    12  1108]
 [   36    59    14    71    34   413    49   204]
 [    0     0    63    10     9     8  1558   216]
 [  111   215   224   142   308   125   309 20193]]

Confusion matrix - validated on Test set:

[[ 119    0    0    0    2    0    0    0]
 [   0  340    0    0   16    0    0    2]
 [   9    0  550    0    0    0    0    2]
 [   1    0    1  174    0    0    0   22]
 [   1  108    0    0   93   11    0  187]
 [   0    1    0    5    0  191    3   25]
 [   0    0    0    1    0    0   84   90]
 [  46   37  100   15   55   10  160 2024]]
```

Režim `AllFeatures` bez použitia eliminácie vplyvu rotácie za pomoci PCA:

```
Confusion matrix - validated on Train set:

[[  797    12    63    12     0     0     0    58]
 [    1  1084     0    20    46    16     0   470]
 [    1     0  2109     1     0     0     0     9]
 [    0     7    13   832     0    33    37    78]
 [    0    88     7    10   706     0     0  1151]
 [   17    20     0    72     6   526    82   157]
 [    0     0     4     0     0     4  1820    36]
 [   57   180    69   194   237    90   460 20340]]

Confusion matrix - validated on Test set:

[[  39    0    0    0   82    0    0    0]
 [   0  346    0    0    0    0    0   12]
 [ 265    0    0    0   28    0    0  268]
 [   0    0    9    0    0    0    0  189]
 [   1  181    0    0   71    0    0  147]
 [   0   17    0    0    0   11    0  197]
 [   0    0    0    0    0    4  141   30]
 [  17   64   15    4   51   16   70 2210]]
```

Príloha C

Popis implementácie nástrojov pre prácu s dátami a ich vizualizáciu

Súbory ktoré vznikli použitím aplikácií `Akcelerak` a `Anotator` je potrebné ďalej spracovať. Za týmto účelom som vyvinul niekoľko nástrojov pre prácu s týmito dátami. Nástroje som implementoval v jazykoch C++, BASH a Python3.

Nástroj `annotation_add_free_motion`

Úloha nástroja je upraviť súbor obsahujúci anotácie[1] tak, že do neho budú vložené anotácie reprezentujúce voľný pohyb. Tieto anotácie budú vložené na miesta (časové intervaly) kde sa nenachádza žiadna iná anotácia.

Pre implementáciu nástroja som rovnako ako pre ostatné nástroje pracujúce priamo s výstupnými súbormi nahrávacích aplikácií zvolil jazyk C++.

Vstupom programu je obsah anotačného súboru privedený na jeho štandardný vstup[2]. Ukážka vstupu:

```
time;label
11685;pushups
12637;pushups_rep1
17402;/pushups_rep1
17801;/pushups
```

Výstupom programu na jeho štandardný výstup[3] je obsah anotačného súboru s pridanými anotáciami reprezentujúcimi voľný pohyb. Ukážka:

```
time;label
0;free_motion_rep1
```

[1] Súbor získaný z anotačnej aplikácie
[2] Viz https://en.wikipedia.org/wiki/Standard_streams#Standard_input_(stdin).
[3] Viz https://en.wikipedia.org/wiki/Standard_streams#Standard_output_(stdout).

```
11684;/free_motion_rep1
11685;pushups
12637;pushups_rep1
17402;/pushups_rep1
17801;/pushups
17802;free_motion_rep2
```

Nástroj `annotation_joiner`

Úloha nástroja je priradiť anotácie z anotačného súboru k dátovým vzorkám[4].

Program som implementoval v jazyku C++. Princíp funkcie programu je jednoduchý – pre každú zo vstupných anotácií nájde časovo najbližšiu dátovú vzorku a túto anotáciu k nej priradí.

Vstupom programu sú argumenty určujúce cestu k súborom s dátami a anotáciami. Uvádzam ukážky obsahov súborov[5].

Súbor s dátami:

```
ax;ay;az;at;time;label
-0.3765;4.5813;10.2270;11.2126;112;-
...
-2.2982;-2.6731;9.0363;9.6996;75972;-
...
```

Súbor s anotáciami:

```
time;label
0;free_motion_rep1
75965;/free_motion_rep1
75966;pushups
...
```

Výstupom programu na jeho štandardný výstup sú anotované dáta. Ukážka[6] výstupu:

```
ax;ay;az;at;time;label
-0.3765;4.5813;10.2270;11.2126;112;112;free_motion_rep1
...
-2.2982;-2.6731;9.0363;9.6996;75972;/free_motion_rep1,pushups
...
```

Použitie programu: `./annotation_joiner data_filename labels_filename`.

[4] Reprezentovaných výstupným súborom z nahrávacej aplikácie.

[5] Tri bodky v ukážkach nahrádzajú ľubovoľný počet riadkov. Desatinné čísla boli orezané na 4 desatinné miesta z dôvodu šetrenia priestoru v ukážke.

[6] Ukážky pre nástroj `annotation_joiner` sú vyhotovené z reálne nameraných dát a anotácií. Ako možno vidieť v 4. riadku ukážky, anotácie boli napárované k najbližšej časovej vzorke a tou bola pre obe anotácie (`/free_motion_rep1`, `pushups`) práve vzorka s časom `75972`.

Nástroj annotation_cutter

Úloha nástroja je vyrezanie stĺpcov dát pre konkrétnu anotáciu z anotovaného dátového súboru[7]. Sekundárnou úlohou je zobraziť zoznam použitých anotácií vo vstupnom súbore. Nástroj som implementoval v jazyku C++.

Program podporuje 3 režimy a to `trimm`, `labels` a `labels-raw`.

Vstupom programu je obsah tréningového CSV súboru[8] privedený na štandardný vstup.

Výstupom programu je text privedený na jeho štandardný výstup. Obsah tohto textu záleží od zvoleného režimu. Režim sa dá zvoliť na základe vstupných parametrov. Ukážky použitia:

`./annotation_cutter --trimm --columns ax,ay,az,time --label pushups_rep1`
Výstupom programu budú vyrezané stĺpce `ax`, `ay`, `az` a `time` obsahujúce dáta prislúchajúce anotácii `pushups_rep1` vo formáte CSV.

`./annotation_cutter --labels`
Výstupom bude prehľad anotácií spolu s časovým intervalom, ktorý im prislúcha:

```
Label [pushups]
Start: 11685; End: 17801
Label [pushups_rep1]
Start: 12637; End: 17402
```

`./annotation_cutter --labels-raw`
Výstupom je zoznam anotácií oddelených znakov nového riadku `\n`. Ukážka:

```
pushups
pushups_rep1
```

Nástroj annotation_cut_all

Úloha nástroja je vyrezať z tréningového CSV súboru časti nameraných dát prislúchajúce konkrétnym anotáciám a uložiť ich do separátnych súborov. Pre každú anotáciu tak vznikne súbor obsahujúci namerané údaje, ktoré tejto anotácii prislúchajú. Program využíva k svojmu chodu nástroj `annotation_cutter`. Najskôr za pomoci režimu `labels-raw` získa názvy všetkých anotácií v tréningovom súbore. Ďalej použije režim `trimm` nástroja `annotation_cutter` za pomoci ktorého získa vyrezané údaje prislúchajúce jednotlivým anotáciám vo formáte CSV. Tieto údaje sú následne uložené do jednotlivých súborov. Z dôvodu potreby opakovaného spúšťania nástroja `annotation_cutter` a následnej práce s jeho výstupom som pre uľahčenie implementácie riešenia použil jazyk BASH.

Názvy výstupných súborov sú vo formáte `X_Y.csv` kde `X` prislúcha názvu tréningového súboru a `Y` prislúcha názvu konkrétnej vyrezanej anotácie.

Vstupom programu sú argumenty. Tie určujú priečinok pre ukladanie výstupných súborov, stĺpce, ktoré majú byť vyrezané a názov vstupného súboru, ktorý obsahuje tréningové dáta.

[7] Súboru, ktorý je výstupom nástroja `annotation_joiner`.
[8] Súboru ktorý obsahuje nahrávku tréningu s už priradenými anotáciami.

Výstupom programu sú vytvorené súbory vo výstupnom priečinku a priebežne vypisovaný text na štandardný výstup, ktorý informuje používateľa o aktuálne prebiehajúce činnosti.

Použitie nástroja:
`bash ./annotation_cut_all [output_directory] [columns] [input_filename]`

Príklad použitia:
`bash ./annotation_cut_all out "ax,ay,az,time" trening.csv`
Výstupom bude napríklad súbor `trening_pushups_rep1.csv` umiestnený v priečinku `out`.

Nástroj `repetitions_sorter`

Úloha nástroja je vytriediť CSV súbory vo vstupnom priečinku do priečinkov[9] vo výstupnom priečinku na základe ich názvov. Cieľom je prekopírovať jednotlivé opakovania cvičení získané vystrihnutím za pomoci nástroja `annotation_cut_all` do priečinkov podľa typu vykonaného cviku. Napríklad všetky opakovania kľukov `Y_pushups_repX.csv`[10] budú prekopírované do priečinku `OUT/pushups/` kde `OUT` značí názov výstupného priečinku. Z dôvodu uľahčenia a prehľadnosti implementácie som zvolil jazyk Python3.

V súbore `repetitions_sorter.py` sa nachádza definícia slovníku `belongs_to_folder`, ktorá určuje mapovanie názvov súborov na výstupné priečinky. Kľúčmi sú názvy výstupných priečinkov a hodnotami sú listy[11] obsahujúce reťazce korešpondujúce „strednej“ časti názvu triedených súborov. Ukážka položky slovníku: `"pushups": ["pushups_rep"],`

Program hľadá triedené súbory za pomoci porovnávania názvu súboru s regulárnymi výrazmi, ktoré sa skladajú z:

1. Začiatku názvu aj s cestou: `^.*\/[a-zA-Z]+[a-zA-Z0-9_]*_`
2. Strednej časti názvu napr. `pushups_rep`
3. Konca názvu: `[1-9]+[0-9]*\.csv$`

Pokiaľ sa nepodarilo názov súbory napárovať na ani jeden z vytvorených[12] regulárnych výrazov, tak je súbor skopírovaný do priečinku s názvom uloženým v premennej `other_folder`.

Vstupom programu sú argumenty, ktoré určujú priečinok so vstupnými súbormi a priečinok v ktorom budú vytvorené priečinky s výstupnými súbormi.

Použitie programu:
`python ./repetitions_sorter.py folder_with_files sort_into_folder`

[9] Ak neexistujú tak ich program vytvorí.
[10] Kde `Y` reprezentuje názov tréningu z ktorého bol cvik vystrihnutý a `X` reprezentuje poradové číslo opakovania cviku.
[11] Dátový typ v jazyku Python3.
[12] Pre každý element v listoch, ktoré sú hodnotami pre jednotlivé kľúče v slovníku `belongs_to_folder` je vytvorený regulárny výraz. Následne sa názvy súborov porovnávajú so všetkými takto vytvorenými regulárnymi výrazmi.

Nástroj `spectrogram`

Úloha nástroja je zobraziť nahrávku[13] tréningu v podobe grafu. Tento graf je tvorený 8 vnorenými grafmi – dvomi pre každú zo štyroch osí[14]. Prvým vnoreným grafom pre konkrétnu os je graf zobrazujúci vývoj hodnoty meranej veličiny danej osi v čase. Druhým vnoreným grafom je spektrogram zobrazujúci vývoj frekvenčného spektra signálu danej osi v čase. Pre výpočet spektrogramov je použité kĺzajúce sa okno s dĺžkou 10 vzoriek a 80 % prekrývaním sa pričom je nastavená vzorkovacia frekvencie 10Hz (potrebná pre výpočet hodnôt osi y jednotlivých spektrogramov). Pokiaľ nahrávka tréningu obsahuje anotácie, pozadia jednotlivých grafov (nie spektrogramov) sú v úsekoch zodpovedajúcich konkrétnej anotovanej aktivite vyfarbené farbou podľa predom definovaných pravidiel.

Za pomoci slovníku `exercise_has_color` možno prideliť konkrétnu farbu jednotlivým názvom anotovaných aktivít. Kľúčmi sú názvy anotácií a hodnotami sú názvy pridelených farieb. Ukážka prvku slovníku:
`"pushups_rep": "red",`
Na základe tohto slovníku je vytvorený slovník `exercise_has_color_regexes`, ktorého kľúčmi sú regulárne výrazy a hodnotami jednotlivé farby. Tieto regulárne výrazy vznikli spojením regulárneho výrazu `^` značiacim začiatok reťazca s kľúčmi[15] predchádzajúceho slovníku a s regulárnym výrazom `[1-9]+[0-9]*$` označujúcim poradové číslo opakovania. Pri načítaní súboru obsahujúceho nahrávku tréningu sú extrahované jednotlivé anotácie spolu s časmi ich začiatku a konca. Každá z týchto anotácií je porovnaná s vytvorenými regulárnymi výrazmi a v prípade zhody je časový interval grafov zodpovedajúci priebehu anotovanej aktivity vyfarbený príslušnou farbou.

Vstupom programu sú argumenty a súbor obsahujúci nahrávku tréningu. Výstupom programu je graf (ukážku grafu možno vidieť na obrázkoch C.1 a C.2), ktorý sa zobrazí používateľovi. Nástroj možno spustiť v troch režimoch:

1. S dvomi argumenty:
   `python ./spectrogram.py a workout_a.csv`
   Nástroj vizualizuje dáta z akcelerometru zo súboru `workout_a.csv`. Prvý argument programu značí druh pohybového senzoru, môže nadobúdať dvoch hodnôt a to „`a`“ pre zobrazenie dát z akcelerometru a „`g`“ pre zobrazenie dát z gyroskopu. Na základe tohto argumentu vie nástroj aký typ dát (aké osi) má v súbore očakávať.

2. S jedným argumentom:
   `python ./spectrogram.py g`
   Nástroj vizualizuje dáta z vybraného pohybového senzoru zvoleného prvým argumentom nachádzajúce sa v súbore s názvom definovaným premennými `default_filename_a` a `default_filename_g`.

3. Bez parametrov:
   `python ./spectrogram.py`
   Nástroj vizualizuje dáta z pohybové senzoru zvoleného premennou `default_mode`

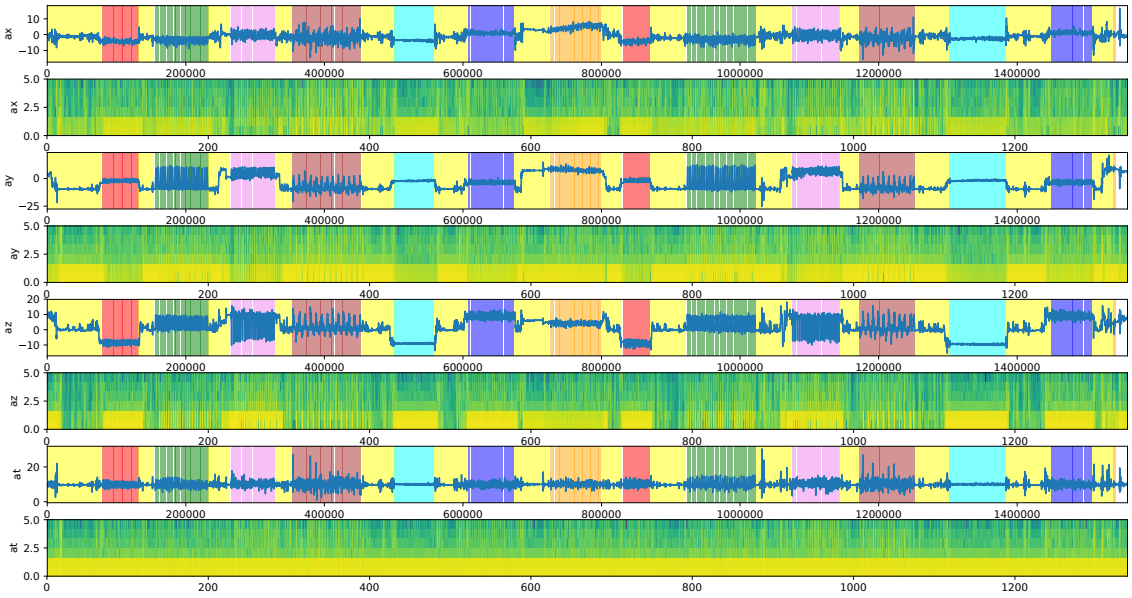
[13] Súbor obsahujúci dáta z akcelerometra/gyroskopu. Táto nahrávka môže a nemusí byť anotovaná, musí však obsahovať stĺpec `label`.

[14] Osi x, y, z symbolizujúce súradnicový systém pohybového senzoru a os t reprezentujúca magnitúdu meranej veličiny.

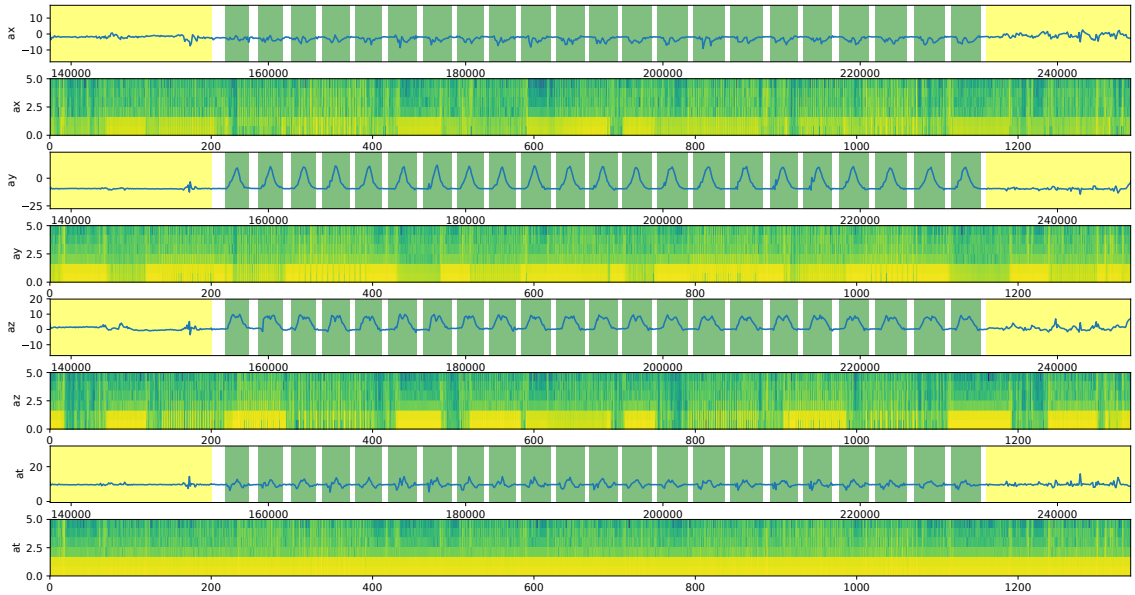
[15] Reprezentujúcimi názov anotovanej aktivity.

nachádzajúce sa v súbore s názvom daným predvolenými názvami (viz predchádzajúca možnosť). Táto premenná môže nadobúdať dve hodnoty a to:
`SpectrogramModes.Accelerometer` a `SpectrogramModes.Gyroscope`.

V nástroji možno povoliť alebo zakázať zobrazovanie popisu x-ovej osi grafov za pomoci premennej `show_x_label`. Zakázaním zobrazovanie popisu x-ovej osi je možné získať väčší vertikálny priestor pre zobrazenie grafov.



Obr. C.1: **Ukážka výstupu nástroja `spectrogram`. Graf zobrazuje tréningovú nahrávku obsahujúcu dáta z akcelerometra.** Zobrazenie popisu x-ovej osi bolo zakázané. X-ová os grafov značí uplynulí čas od začiatku tréningu v milisekundách.



Obr. C.2: **Ukážka výstupu nástroja `spectrogram` po priblížení. Na obrázku možno vidieť anotácie jednotlivých opakovaní cviku drepy.** Nad časovaním anotácií nebola vykonaná žiadna korekcia. Po priblížení jednej z osí x, y, z alebo t sa automaticky ostatné osi grafov (nie spektrogramov) priblížia na rovnaký úsek dát. Tento úsek je definovaný rozsahom x-ovej a y-novej osi priblíženého grafu.