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

ZPRACOVÁNÍ DAT ZE SENZORŮ WEARABLE ZAŘÍZENÍ POMOCÍ STROJOVÉHO UČENÍ

PROCESSING SENSOR DATA FROM A WEARABLE DEVICE BY MACHINE LEARNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MARTIN HLAVAČKA**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, PhD.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Hlavačka Martin, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Zpracování dat ze senzorů wearable zařízení pomocí strojového učení**
Processing Sensor Data from a Wearable Device by Machine Learning
Kategorie: Zpracování signálů

Zadání:

1. Prostudujte problematiku nositelných zařízení a využití senzorů k monitorování fyzické aktivity uživatele.
2. Vytvořte aplikaci do nositelného zařízení, která umožní sbírat data ze senzorů; průběžně pořizujte data při různých fyzických/sportovních aktivitách.
3. Prostudujte problematiku strojového učení, zaměřte se na techniky použitelné v zařízení s nízkým výkonem a spotřebou (wearable).
4. Experimentujte s vybranými algoritmy strojového učení nad pořízenými daty; nalezněte použitelná řešení.
5. Navrhněte a vytvořte aplikaci, která umožní analýzu dat z nositelného zařízení a bude identifikovat fyzické činnosti uživatele.
6. Vyhodnoťte vlastnosti aplikace - jak užitelské rozhraní, tak (především) schopnost identifikovat a analyzovat fyzickou činnost uživatele.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Nishant Shukla: Machine Learning with TensorFlow, January 2018, ISBN 9781617293870
- Jeff Tang: Intelligent Mobile Projects with TensorFlow, Packt Publishing Ltd, 22. 5. 2018
- Aurélien Géron: Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, March 2017
- TensorFlow Lite - book - GitBook, https://nlp.gitbook.io/book/tensorflow/tensorflow-lite

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Cieľom práce je analýza problematiky nositeľných zariadení s operačným systémom Android Wear a rozpoznávaní rôznych pohybových aktivít za pomoci neurónových sietí. Prvotným zameraním je teda identifikovanie a popis najvhodnejšieho nástroja pre rozpoznávanie dynamických pohybov s využitím metód strojového učenia na základe dát získaných z tohto typu zariadení. Praktická časť práce následne komentuje implementáciu samostatne fungujúcej aplikácie pre platformu Android Wear schopnej záznamu a naformátovania dát zo senzorov, tréning neurónovej siete v navrhnutom externom nástroji pre desktop a následné spätné použitie natrénovanej neurónovej siete pre možnosti rozpoznania pohybov priamo v zariadení.

Abstract

The goal of this master's thesis is to analyze the situation of wearable devices with the Android Wear operating system and recognition capabilities of various movement activities using neural networks. The primary focus is therefore on identifying and describing the most appropriate tool for recognizing dynamic movements using machine learning methods based on data obtained from this type of devices. The practical part of the thesis then comments on the implementation of a stand-alone Android Wear application capable of recording and formatting data from sensors, training the neural network in a designed external desktop tool, and then reusing trained neural network for motion recognition directly on the device.

Kľúčové slová

nositeľné zariadenia, inteligentné zariadenia, inteligentné hodinky, android wear, neurálna sieť, tensorflow, rozpoznávanie aktivít, accelerometer, python, konvolučné neurónové siete, strojové učenie, detekcia pohybu

Keywords

wearables, smart devices, smart watches, android wear, neural network, tensorflow, activity recognition, accelerometer, python, convolutional neural networks, machine learning, movement detection

Citácia

HLAVAČKA, Martin. *Zpracování dat ze senzorů wearable zařízení pomocí strojového učení.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, PhD.

Zpracování dat ze senzorů wearable zařízení pomocí strojového učení

Prehlásenie

Prehlasujem, že som tento semestrálny projekt vypracoval samostatne pod vedením pána prof. Ing. Adama Herouta, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Martin Hlavačka
19. mája 2019

Poďakovanie

Týmto by som chcel poďakovať vedúcemu práce, prof. Ing. Adamovi Heroutovi, Ph.D., za odbornú pomoc, rady, konzultácie a pripomienky pri jej písaní.

Obsah

Kapitola 1

Úvod

V posledných rokoch dostáva trh s nositeľnými zariadeniami, špecificky inteligentnými hodinkami, o veľa viac pozornosti nielen od potencionálnych užívateľov ale i od technologických výrobcov, ktorý sa snažia do tohto malého formátu zakomponovať čo najvyšší výkon so zachovaním excelentnej výdrže batérie. Táto skutočnosť umožňuje využiť zariadenia i pre komplexnejšie operácie, ktoré boli doteraz výsadou väčších zariadení ako napríklad smartphone.

Na mobilných telefónoch sa už dlhší čas stretávame s implementáciou určitej formy strojového učenia, ktorá pomáha týmto zariadeniam v užívateľsky viac subjektívnych operáciách, ako napríklad s predikciou ďalších slov, identifikáciou objektov skrz integrovanú kameru, či rozpoznávanie gest na ovládanie zariadenia a pripojených periférií.

Nadväzujúc na spomínanú skutočnosť zväčšujúceho sa výkonu inteligentných hodiniek, je možné práve dané strojové učenie presunúť ako nezávisle bežiacu implementáciu i na tieto zariadenia. Táto možnosť otvára veľmi veľký segment možného využitia pre užívateľsky špecifické požiadavky.

Táto práca ma za úlohu nájsť možné využitia tohto typu funkcionality na rozpoznávanie pohybových aktivít pre nositeľné zariadenia, ktoré podporujú operačný systém Android Wear umožňujúci takmer neobmedzenú možnosť využitia funkcionalít zo systému Android určenému pre smartphone. Výsledný produkt poskytuje funkcionalitu pre automatické rozpoznávanie pohybov špecifických pre užívateľa, ktoré boli zaznamenané a transformované pre účely strojového učenia.

Kapitola 2

Rozpoznávanie pohybu

Historicky bolo získavanie dát náročné či už z pohľadu výpočetného výkonu, veľkosti zariadení alebo financií. Okrem toho bola vo väčšine prípadov vyžadovaná prítomnosť dodatočného dedikovaného hardwaru. S príchodom inteligentných zariadení sa senzory umožňujúce záznam tohto typu dát dostali k bežnému spotrebiteľovi, čo spôsobilo masívne rozšírenie ich využiteľnosti či už vo fitness sfére i v každodenných činnostiach.

Detekcia a rozpoznávanie ľudskej činnosti je obšírna oblasť zameraná na identifikáciu konkrétneho pohybu alebo činnosti na základe údajov zo snímačov ako akcelerometer alebo gyroskop. Proces rozpoznávania sa dá rozdeliť do niekoľkých kategórií, podľa typu zamerania:

- **Činnosti**: rozpoznávanie druhu opakujúcej sa aktivity (zväčša športovej) ako napríklad chôdza, beh, cyklistika, plávanie a iné.
- **Cviku**: detekcia unikátneho úseku v periodickej činnosti identifikovaného ako 1 perióda = 1 opakovanie - zväčša sa jedná o cvik a záznam počtu opakovaní.
- **Gesta**: rozpoznávanie špecifického pohybu v krátkom časovom úseku.
- **Polohy zariadenia**: identifikácia polohy zariadenia v kontexte predošlého pohybu využívaná hlavne v spojení s inými senzormi - napríklad zariadenie prevrátené do vertikálnej polohy vo vzťahu k orientácií očí a display-u.

Hlavným problémom detekcie je predpovedanie aktivity z malého a obmedzeného množstva údajov, ktoré pochádzajú z jedného, prípadne niekoľkých senzorov. Tento problém je teda radený medzi jedno alebo viac-rozmerné klasifikačné úlohy časových radov.

Komplikácie taktiež spôsobuje skutočnosť, že neexistujú žiadne zjavné ani priame spôsoby ako priradiť zaznamenané údaje zo snímačov k jednotlivým aktivitám a spôsob vykonávania aktivity sa môže subjekt od subjektu s odchýlkami líšiť.

Základnou úlohou a zámerom je teda zaznamenať čo najviac dát z rôznych snímačov, identifikovať spoločné špecifiká a jedinečné charakteristiky pohybov, tie následne zovšeobecniť a vytvoriť tak model schopný klasifikácie činnosti na doposiaľ neanalyzovaných/nových dátach. [23]

2.1 Triviálne rozpoznávanie

Rozpoznávanie pohybov na základe identifikácie radikálnych zmien v zaznamenávanom pohybe je jedna z možností ako implementovať triviálnym spôsobom detekciu rôznych pohy-

bov. Tento prístup však prináša veľa problémov a nepresností. Práve z dôvodu závislosti na veľmi dobre identifikovateľných zmenách nie je možné rozpoznávať všetky druhy pohybov. Tento prístup taktiež neumožňuje plne automatizované rozpoznávanie viacerých aktivít naraz kvôli prílišnej podobnosti charakteristík zmien pohybov. Ďalším problémom je skutočnosť, že rotácia a poloha zariadenia sa môže oproti navrhovanému riešeniu líšiť, čo prináša ďalšie nepresnosti.

Implementáciu tohto typu rozpoznávania môžeme vidieť v bakalárskej práci [10] na ktorú táto práca nadväzuje. V práci sa podarilo získať dostačujúce výsledky pre reálu použiteľnosť v praxi avšak rozpoznávanie malo niekoľko reštrikcií a problémov ako napríklad:

- poloha zariadenia pri vykonávaní cviku
- dostatočná odlišnosť v cvikoch
- manuálny výber typu rozpoznávaného cviku
- limit dĺžky opakovania cviku
- žiadne dodatočné pohyby (prílišný „šum“ v dátach)
- zameranie sa zväčša na jednu os pohybu

Vyššie uvedené problémy a komplikácie pri rozpoznávaní viedli k pokračovaniu formou tejto práce, ktorej zameraním je využitie strojového učenia a odstránenia väčšiny reštrikcií spôsobených pôvodným typom detekcie pohybu.

2.2 Strojové učenie

Z dôvodov spomínaných problémov v sekcii 2.1 je vhodnejším prístupom k detekcii pohybov práve metóda založená na strojovom učení. Tá umožňuje eliminovať väčšinu predchádzajúcich problémov bez nutnosti komplexnej a neustálej analýzy akýchkoľvek nových dát. Ideálnym by bolo umožnenie automatického extrahovania nových rysov jednotlivých pohybov k dosiahnutiu ešte vyššej presnosti priamo zo surových dát.

Práve neurónové siete a strojové učenie nám toto správanie a funkcionalitu prinášajú. Umožňujú tak automatické učenie a optimalizovanie svojich predikcií na základe nových dát bez dodatočnej intervencie programátora. Svojou presnosťou a všestranným využitím triviálne metódy plne nahradili.

Na základe použitia v praxi a výsledkov, ktoré boli dosiahnuté s ich pomocou, existujú v dnešnej dobe 2 typy prístupov k strojovému učeniu pre účely rozpoznávania aktivít a to *rekurentné*[1] a *konvolučné* 3.1 neurónové siete. Táto práca sa bude zaoberať vo veľkej miere druhým spomínaným typom, teda *konvolučnými* sieťami.

[1] http://ics.upjs.sk/~novotnyr/home/skola/neuronove_siete/neurony/Rekur.htm

Kapitola 3

Neurónové siete

Táto kapitola obsahuje priblíženie teórie neurónových sietí a základné informácie, ktoré budú východiskovým stanoviskom pri tvorbe samotného algoritmu rozpoznávania a návrhu štruktúry neurónovej siete spolu s prístupom k jej učeniu.

Práca sa zaoberá hlavne konvolučnými neurónovými sieťami, ktoré sa v praxi využívajú pri rozpoznávaní obrazov. Tento typ sietí bol zvolený práve z dôvodu podobnosti štruktúry dát grafického obrazu a dát z akcelerometra. Oba typy obsahujú pre každý „bod“ trojicu dátových informácií, v prípade obrazu sa jedná o farebné zložky RGB a v prípade akcelerometra ide o jednotlivé osi pohybu, menovite x, y a z. To umožnilo rýchlejší a časovo efektívnejší prístup k špecifikácii a výberu správneho typu i zloženia zvolenej siete. Nasledujúce sekcie sú sústredené na ich priblíženie, následnú analýzu ich základných prvkov a v neposlednom rade i proces trénovania vytvorenej siete.

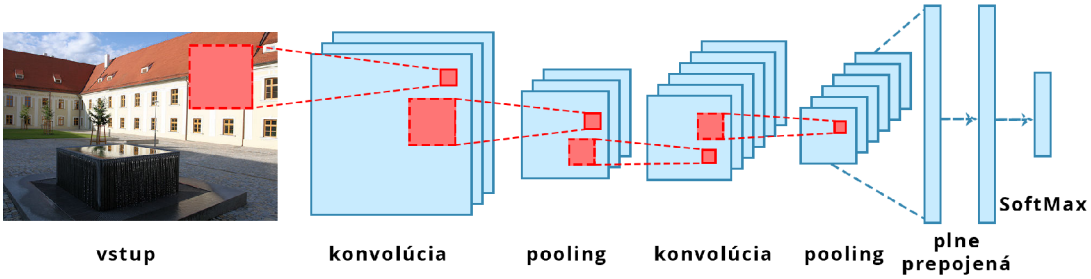
3.1 Konvolučné siete

Informácie v tejto sekcii boli čerpané z [17, 18, 12, 14]. Špecifickým typom neurónových sietí patriacich medzi dopredné viacvrstvové perceptronové sú siete konvolučné, ktorých jednou z charakteristických použití je extrakcia príznakov. Najčastejšie sa s týmto typom sietí môžeme stretnúť pri rozpoznávaní príznakov na 2D obrázkoch bez predošlej hĺbkovej transformácie. Keďže tak ako rozpoznávanie obrazov i rozpoznávanie pohybov by nemalo byť závislé na dĺžke/mierke, posune či čiastočnej deformácii, konvolučné siete túto skutočnosť veľmi vhodne a účinne eliminujú práve svojimi špecifickými vlastnosťami ako:

- priestorové prevzorkovanie
- zdieľanie váh
- extrakcia lokálnych i globálnych príznakov

Schopnosť získania základných príznakov/vzorov z dát bez predošlého predspracovania vyžaduje vyššiu komplexnosť samotnej štruktúry siete z dôvodu nutnosti zachytenia všetkých typov príznakov.

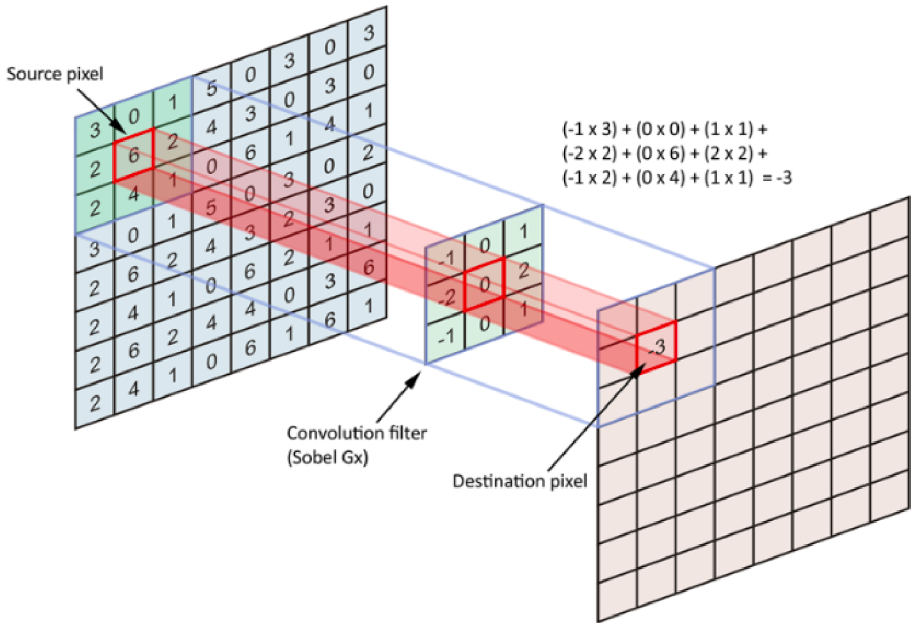
Z obrázka 3.1 a samotnej definície konvolučnej neurónovej siete je možné identifikovať niekoľko typov vrstiev a operácií v nich:



Obr. 3.1: Príklad štruktúry konvolučnej neurónovej siete, ktorej je na vstup privedený farebný RGB obraz, reprezentovaný ako 3 separátne vrstvy pre každú z farieb RGB, a následne je realizovaná extrakcia príznakov za pomoci konvolúcie. Ďalším prvokom štruktúry je pooling slúžiaci na znižovanie priestorovej náročnosti a posledné prvky (plne prepojené) siete vykonávajú samotné klasifikovanie identifikovaných príznakov z predchádzajúcich vrstiev. Vrstva *SoftMax* tieto predikcie následne upravuje do vhodného formátu.

3.1.1 Konvolúcia

Primárna vrstva, na ktorej sú konvolučné siete založené, umožňujúca extrakciu rysov zo vstupného obrazu. Samotná operácia zachováva priestorové závislosti medzi pixelmi práve pomocou postupného aplikovania na malé štvorcové plochy vstupu. Použitie menšieho štvorcového jadra než je vstup umožňuje detekciu menších príznakov ako napríklad v hrany v rozmerne niekoľko násobne väčších vstupných obrazoch. Táto skutočnosť taktiež znižuje celkové nároky operácie.



Obr. 3.2: Príklad aplikovania konvolúcie s 3x3 jadrom (pre detekciu hrán–Sobelov operátor Gx[1]) na 2d maticu, prevzaté z [4]

.

[1] https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

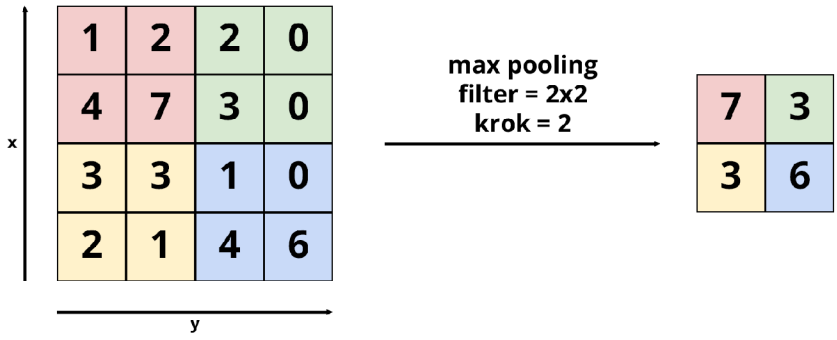
Pri bežných neurónových sieťach je každý neurón matice váh prepojený s neurónom zo vstupu. V prípade konvolučných je však použitá technika zdieľania parametrov umožňujúca neurónom v jednej vrstve zdieľať rovnaké hodnoty parametrov. Konvolučná neurónová sieť využíva techniku zdieľania parametrov. To umožňuje učenie sa len 1 skupiny parametrov použiteľnej pre všetky lokácie namiesto učenia sa osobitných skupín pre každú pozíciu vstupu.

Na obrázku 3.2 je možné vidieť operáciu konvolúcie s 3x3 jadrom v 2D. Okrem veľkosti a podoby filtru je v prípade operácie definovaná i dĺžka kroku, ktorý určuje počet polí pri posune. Aplikovanie danej operácie vyúsťuje ku zmenšovaniu výslednej matice. Tomuto javu sa dá predchádzať použitím zarovnania, ktoré sa vo väčšine prípadov aplikuje ako okolie vyplnené samými 0.

V prípade konvolúcií v konvolučných neurónových sieťach sa však zväčša jedná o 3-rozmernú operáciu keďže obraz je reprezentovaný ako 3D matica (2D matica + reprezentácia každej z farieb RGB).

3.1.2 Pooling

Táto vrstva kombinuje výstupy skupín neurónov z predchádzajúcej vrstvy do jedného neurónu. Hlavnou funkciou je teda progresívne znižovanie priestorovej náročnosti.



Obr. 3.3: Vizuálna reprezentácia operácie max poolingu na 4×4 vstupe s krokom o veľkosti 2, ktorý efektívne zmenšuje priestorový rozmer vstupu na štvrtinu. Hlavnou úlohou je zníženie počtu parametrov a výpočtov čo pozitívne ovplyvňuje tréningový čas a preoptimalizáciu.

Ako možno vidieť i na obrázku 3.3, najčastejšie používaná funkcia určená pre kombinovanie hodnôt je funkcia *max* (zväčša s filtrom 2×2 a krokom o dĺžke 2), no v niektorých prípadoch sa používajú i iné operácie ako napríklad priemer alebo L2-norm[2], no ich výsledky v praxi často zaostávajú.

V niektorých prístupoch ku tvorbe konvolučných sietí sa odporúča nevyužívať túto vrstvu vôbec a namiesto nej využiť v niektorých konvolučných vrstvách väčšiu dĺžku kroku, ktorá čiastočne nahradzuje funkciu pooling vrstvy.

3.1.3 ReLU

Táto vrstva, ktorá predstavuje nelineárnu funkciu zloženú s dvoch lineárnych súčastí, je ďalším typom aktivačnej funkcie s niekoľkými charakteristickými vlastnosťami:

[2] http://mathworld.wolfram.com/L2-Norm.html

- triviálne vyhodnocovanie
- menšie straty gradientu v hlbokých sieťach/modeloch
- problém s úmrtím vrstvy (vždy vráti rovnakú hodnotu) v prípade príliš vysokej rýchlosti učenia

$$f(x) = \max(0, x) \tag{3.1}$$

ReLU teda aplikuje túto funkciu na všetky elementy vstupného tenzoru bez ovplyvnenia jeho priestorových a hĺbkových informácií.

3.1.4 Plné prepojenie

Neuróny v plne prepojenej vrstve majú kompletné spojenie s predchádzajúcou vrstvou (každým neurónom), tak ako v bežných neurónových sieťach.

Hlavným účelom vrstvy je využitie komplexných vstupných vlastností/rysov z predchádzajúcich konvolučných a pooling vrstiev pre klasifikovanie vstupného obrazu/signálu do niekoľkých skupín na základe tréningových dát.

Celková suma výstupu všetkých pravdepodobností z plne prepojenej vrsty je rovná 1. Vo väčšine prípadov sa používa *Softmax*[3] aktivačná funkcia, ktorá zabezpečuje prevod ohodnotení na vektor hodnôt medzi 0 až 1 so sumou rovnou 1.

3.1.5 Stratová funkcia

Nazývaná i cenovou funkciou, slúži na vyhodnotenie modelu, či určenie rozdielov medzi prognózou a správnou identifikáciou/odhadom. Predstavuje veľmi dôležitú časť samotnej neurónovej siete, s ktorej pomocou sa v kombinácii s backpropagáciou upravujú váhy siete čo spôsobuje jej učenie. Niekoľko príkladov používaných funkcií je možné získať z [9].

Výsledok funkcie predstavuje kladnú hodnotu, ktorej znižovaním sa dosahuje vyššej robustnosti neurónovej siete.

3.2 Tréning neurónovej siete

Pojem „tréning neurónovej siete“ je chápaný ako proces, pri ktorom je hlavnou úlohou optimalizovať váhy vektorov prepojení medzi jednotlivými neurónmi. Tréning teda prebieha v zmysle poskytnutia veľkého množstva tréningových vzoriek na vstup neurónovej siete pričom postupne upravujeme hodnoty váhových prepojení s cieľom minimalizovať celkové množstvo chybných výsledkov. V teórii je teda snaha dostať sa na celkové množstvo chýb rovné nule.

Samotný proces trénovania je závislý na spôsobe implementácie siete a taktiež na výsledkoch, ktoré sú zamýšľané dosiahnuť. Práve z tohto dôvodu je daná časť komplexnejšie rozoberaná v kapitole implementácie 6, kde sú jednotlivé operácie a ich použitie v prípade výslednej aplikácie bližšie špecifikované.

[3] https://en.wikipedia.org/wiki/Softmax_function

Kapitola 4

Android Wear knižnice

Android ako platforma ponúka niekoľko odlišných aplikačných rozhraní na programovanie, trénovanie a interpretovanie neurónových sietí i použitie strojového učenia. Medzi najznámejšie patria:

4.1 Tensorflow

Open-source softwarová knižnica pre data-flow programovanie. Je založená na symbolických výpočtoch a určená najmä pre použitia v aplikáciách využívajúcich strojové učenie [19]. V prípade systému Android a jeho odnoží spoločnosť Google i firmy tretích strán využívajú jej funkcionalitu v niekoľkých aspektoch systému:

- rozpoznávanie gest pre základné ovládanie zariadenia (zväčša Android Wear)
- predikcia a korekcia slov i viet
- inteligentné predpovedanie užívateľských aktivít naprieč dňom/týždňom s odporučeniami ohľadom premávky, jedla či poveternostných podmienok
- rozpoznávanie tváre, obrysov a úsmevu
- inteligentne generované rýchle odpovede
- rozpoznávanie textu a jeho extrakcia
- doporučenia na základe predchádzajúcich výberov
- kustomizované vyhľadávania
- detekcia možných zdravotných komplikácií
- radenie a zoskupovanie rôznych typov dát na základe podobností
- cielená reklama

V prípade tejto knižnice je dostupných niekoľko verzií, ktoré ponúkajú rôznu komplexitu funkcionalít pre špecifické účely a typy použití. Pre operačný systém Android sú dostupné 2 implementácie:

4.1.1 Tensorflow-Mobile

Prvá verzia knižnice určenej pre mobilné zariadenia, ktorej funkcionalita bola len čiastočne optimalizovaná no stále vyžadovala väčší výkon aký bolo možné poskytnúť na nositeľných zariadeniach. Z toho dôvodu sa spoločnosť rozhodla na začiatku roku 2019 podporu pre túto verziu plne ukončiť.

Informácia indikujúca nemožnosť použitia tejto knižnice na zariadeniach s menšou pamäťou RAM však nebola priamo uvedená ani v oficiálnej dokumentácii a preto prvotný vývoj prebiehal práve s touto verziou. Až po vygenerovaní testovacej neurónovej siete a jej nasadení na inteligentné hodiny *Motorola Moto 360 sport*[1] bol odhalený problém s nedostatkom pamäte RAM a nemožnosťou spustenia samotného procesu rozpoznávania.

Práve z vyššie uvedených dôvodov a zistení bolo nutné využiť novšiu implementáciu, ktorá už je priamo optimalizovaná pre zariadenia s malým výkonom a kapacitou, ako napríklad inteligentné hodinky.

4.1.2 Tensorflow-Lite

Táto verzia by podľa informácií mala byť po Q1/2019 jediným oficiálnym *Tensorflow* riešením umožňujúcim beh modelov strojového učenia na mobilných zariadeniach typu smartphone, inteligentné nositeľné zariadenia a vstavané zariadenia. Jedná sa o ďalší vývojový krok po 4.1.1 ponúkajúci efektívny, pamäťovo menej náročný beh aplikácií s nižšou latenciou na operačných systémoch Android[2] a iOS[3]. Zároveň podporuje možnosť využitia hardwarovej akcelerácie s cielením na urýchľovače pomocou *Neural Network API*[4] .

Nevýhodou voči verzii *Tensorflow-Mobile* je skutočnosť, že generovaný model môže využiť len 1 vstupný a 1 výstupný uzol, zatiaľ čo v prípade druhej spomínanej je možné počas behu špecifikovať uzle manuálne a je možné teda využiť i multi-vstupové/výstupové[5] siete. Podporovaná množina funkcionality a operátorov je taktiež mierne zmenšená. To však pre účely vyvíjanej aplikácie nebol rozhodujúci faktor, dôležitejšia bola schopnosť rýchleho a bezproblémového behu i na zariadeniach s nízkym výpočetným výkonom ako napríklad inteligentné hodinky.

4.2 Deeplearning4j

Open-source distribuovaná knižnica (ďalej *DL4J*) a výpočtový framework pre programovací jazyky Java s veľkou kontribúciou firmy Skymind[6]. Obsahuje taktiež podporu GPU a distribuovaných výpočtových softwarov ako Apache Spark[7] a Hadoop[8]. Informácie čerpané v tejto sekcii pochádzajú z [16].

Samotný framework umožňuje skladanie plytkých neurónových sietí ako napríklad:

- konvolučné siete
- obmedzené Boltzmannove stroje

[1] https://www.motorola.com.au/products/moto-360-sport
[2] https://www.android.com/
[3] https://www.apple.com/sk/ios/ios-12/
[4] https://developer.android.com/ndk/guides/neuralnetworks/
[5] https://keras.io/getting-started/functional-api-guide/#multi-input-and-multi-output-models
[6] https://skymind.ai/
[7] https://spark.apache.org/
[8] https://hadoop.apache.org/

- word2vec[9]
- doc2vec[10]
- autoenkódery
- rekurzívne siete

do komplexných hlbokých sietí rôznych typov. Medzi dostupnými nástrojmi sa tiež nachádzajú rôzne vyzualizačné nástroje a výpočtové grafy.

Výhodou DL4J je kompatibilita s *Tensorflow* a *Keras*[11], čo umožňuje importovanie modelov z *Tensorflow* no s podmienkou, že boli vytvorené za pomoci *Keras*.[6]

4.2.1 Android For Deep Learning

I keď vo väčšine prípadov je na tréning neurónových sietí nutný vyšší výkon, existuje i odnož a špecifický návod na sfunkčnenie celého *DL4J* priamo na platforme Android[12]. Keďže táto implementácie nie je do hĺbky optimalizovaná pre beh na výkonovo obmedzenejších zariadeniach, je nutné aby bolo splnených niekoľko výkonnostných štandardov ako 200 MB voľnej pamäte, Android API level 21+ a Android Studio 2.2+.

Ak sa vezme do úvahy samotné nositeľné zariadenia a ich veľmi obmedzenú výpočtovú kapacitu, bolo by veľmi problematické tento framework využiť pre potreby bežného užívateľa. To bol aj jeden z rozhodujúcich faktorov pri výbere medzi *DL4J* a *Tensorflow-Lite* [22].

[9] https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-word2vec
[10] https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-doc2vec
[11] https://keras.io/
[12] https://deeplearning4j.org/docs/latest/deeplearning4j-android

Kapitola 5

Návrh aplikácie

Hlavným zámerom tejto práce bolo vytvoriť plne funkčnú a konkurencie-schopnú aplikáciu pre operačný systém Android Wear, ktorá bude umožňovať bližšie špecifikovanú funkcionalitu:

1. Záznam a ukladanie dát na vyžiadanie užívateľa pri fyzickej aktivite zo senzorov (akcelerometra–lineárna akcelerácia) nositeľného zariadenia vo formáte a za účelom ďalšieho spracovania.

2. Aplikácia umožňujúca analýzu, spracovanie, optimalizáciu a reprezentáciu dát nazbieraných z predchádzajúceho kroku so zámerom tréningu navrhnutej neurónovej siete a s cieľom zlepšenia automatického rozpoznávania zachytených typov pohybov z dát.

3. Spätné využitie natrénovanej neurónovej siete z 2. kroku pre automatickú detekciu typu pohybu a jednotlivých repetícií v kontexte ostatných natrénovaných pohybov neurónovej siete.

Prvotný koncept aplikácie bol založený na princípe *on-board* (tzn. priamo na nositeľnom zariadení) tréningu neurónovej siete i samotného procesu rozpoznávania. Tento prístup by umožnil absolútnu nezávislosť na externom výpočetnom výkone alebo dodatočných zariadeniach. Problém však predstavoval prvý krok, teda samotný tréning neurónovej siete, ktorý je výpočetne veľmi náročný. Pre systém Android je dostupných niekoľko nástrojov a portácií kódu, ktoré by umožnili tréning priamo na zariadení, keďže Tensorflow podporuje i natívny jazyk *C++*, ktorý by umožňoval, narozdiel od jazyka *Python*, jednoduchý beh na platforme Android [5].

5.1 Existujúce riešenia

Jednou z oblastí, pre ktoré boli prvotne nositeľné zariadenia určené bola práve fitness a z toho dôvodu v dnešnej dobe existuje niekoľko desiatok aplikácií umožňujúcich záznam a sledovanie štatistík počas cvičenia. Väčšina týchto aplikácií je zameraná na stredne až dlho trvajúce opakujúce sa aktivity typu beh, chôdza, cyklistika a podobne. Sledovanie počtu opakovaní a iných štatistík pri rezistenčnom cvičení je viac ojedinelé práve z dôvodu veľkej rozmanitosti spôsobu vykonávania pohybov a taktiež vyššej náročnosti na detekciu komplexných pohybov narozdiel od sledovania malého počtu charakteristík v prípade spomínaných dlhotrvajúcich aktivít.

V dobe písania tejto práce je pre Android Wear (v prípade Android a smartphone sa počet dostupných líši práve kvôli zvýšenému výkonu a dĺžke existencie na trhu) dostupných len pár exemplárov schopných automaticky detekovať opakovania vykonávaných rezistenčných cvičení.

Najkomplexnejšiu funkcionalitu ponúka *Google Fit*[1] od samotného tvorcu systému. Aplikácia ponúka veľmi presné rozpoznávania založené na strojovom učení a i napriek tomu, že je zabalené v jednoduchom užívateľskom prostredí ponúka veľké množstvo funkcionality. Najväčšou nevýhodou aplikácie bola však v dobe analýzy neschopnosť plne pracovať bez pripojenia k internetu. Toto bola jedna z hlavných motivácií tejto práce a tvorby aplikácie schopnej pracovať plne offline.

Okrem *Google Fit* je dostupných ešte niekoľko ďalších menej známych aplikácií, v ktorých prípade je však možné nájsť niekoľko nedostatkov ako napríklad:

- využívanie nekomplexných, primitívnych, algoritmov na rozpoznávania
- aktualizovanosť aplikácie
- chýbajúca podpora nových verzií operačného systému
- obmedzené množstvo typu rozpoznávaných cvikov
- generalizované rozpoznávanie
- závislosť na pripojenom smartphone

Všetky predchádzajúce nedostatky sa praktický výstup tejto práce, aplikácia na nositeľné zariadenia, snaží aktívne odstrániť a priniesť užívateľom chýbajúcu funkcionalitu spolu s ďalšími vylepšeniami čím sa zvýši celková použiteľnosť pri tomto type aktivít.

5.2 Architektúra

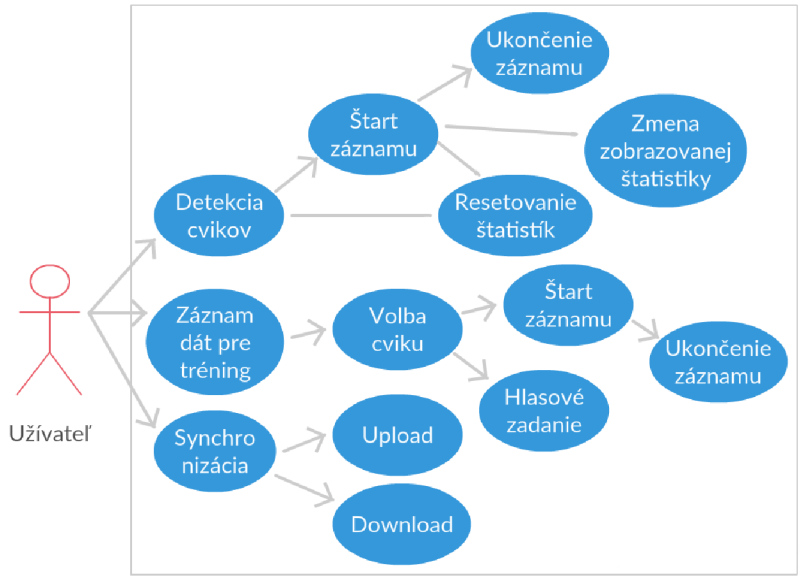
Na základe predchádzajúceho návrhu boli identifikované 2 samostatne fungujúce zložky, komponenty, umožňujúce vykonanie požadovanej funkcionality ako celku:

1. Aplikácia pre nositeľé zariadenie umožňujúca:
   - Zaznamenávanie fyzických aktivít užívateľa na vyžiadanie.
   - Intepretácia bežiacej neurónovej siete a výsledkov jej rozpoznávania.
2. Aplikácia pre *desktop* umožňujúca analýzu a spracovanie zaznamenaných dát pre účely tréningu neurónovej siete.

Dôvodom rozdelenia funkcionality a zavedenia externého nástroja pre tréning neurónovej siete vychádzal z predošlej analýzy dostupných nástrojov pre platformu Android 4 a problémov vyskytujúcich sa so samotným behom 4.1.1 knižnice na zariadeniach s malou výpočetnou kapacitou. Táto informácia bola spolu s nižšou výdržou a tepelným vyžarovaním pri extrémnej záťaži rozhodujúcim faktorom.

Existuje však niekoľko indikácií (odpovede[2] od inžinierskeho vedenia na čele s *Rajat Monga*[3] , že samotná implementácia knižnice *Tensorflow Lite* 4.1.2 bude v budúcnosti

[1] https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=sk
[2] https://github.com/tensorflow/tensorflow/issues/17328
[3] https://www.linkedin.com/in/rajatmonga



Obr. 5.1: Diagram prípadov použitia aplikácie určenej pre nositeľné zariadenia zachytávajúci všetky funkčné moduly a interakcie s nimi.

ponúkať funkcionalitu tzv. *on-board* tréningu neurónových sietí optimalizovaného pre výpočetne obmedzené zariadenia, čo by umožnilo zjednotenie celej funkcionality do jednej komplexnej aplikácie.

5.2.1 Komunikačné rozhranie medzi komponentami

Keďže komponenty pracujú nezávisle na sebe bolo nutné stanoviť rozhranie a formát dát, ktorý bude požadovaný pre komunikáciu a kompatibilitu.

Prvým požadovaným aspektom je formát výstupu zaznamenaných dát z akcelerometra v zariadení. Pre čo najnižšiu náročnosť na predspracovanie a nárok na výpočetný výkon nositeľného zariadenia bol zvolený formát CSV[4]obsahujúci 5 hodnôt pre každý riadok záznamu:

1. slovná reprezentácia typu vykonávanej aktivity/cviku
2. časový odtlačok reprezentovaný pomocou *unixového*[5] *časového formátu*
3. Prirodzené číslo reprezentujúce hodnotu lineárnej akcelerácie pre os X s presnosťou na 6 desatinných miest.
4. Prirodzené číslo reprezentujúce hodnotu lineárnej akcelerácie pre os Y s presnosťou na 6 desatinných miest.
5. Prirodzené číslo reprezentujúce hodnotu lineárnej akcelerácie pre os Z s presnosťou na 6 desatinných miest.

[4] https://tools.ietf.org/html/rfc4180#page-2
[5] https://en.wikipedia.org/wiki/Unix_time

Kód 5.1: ukážka výstupného záznamu z aplikácie pre záznam z nositeľného zariadenia

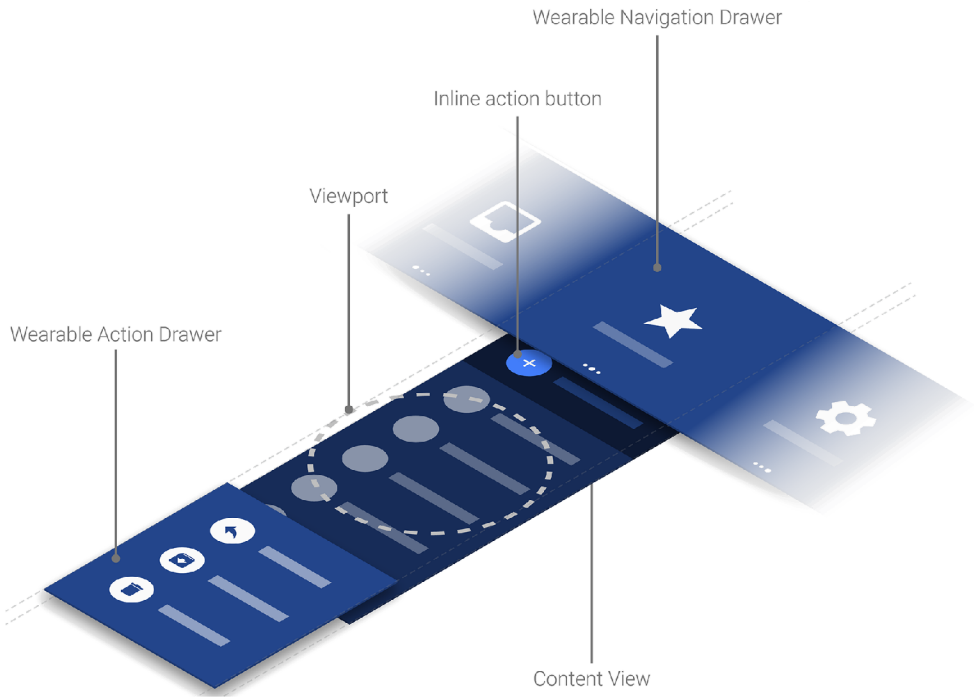
```
Squats,1541603654329,6.708553,0.225055,9.179369
Squats,1541603654349,6.997553,1.115056,8.079300
```

V prípade rozhrania pre akceptovanie natrénovanej neurónovej siete je už vopred vyžadovaný súbor **.tflite* [20] obsahujúci zdrojové súbory pripravenej siete.

Okrem tohto súboru je nutné poskytnúť informácie o všetkých dostupných aktivitách/-cvikoch, ktoré daná sieť dokáže rozpoznávať zoradené v správnom poradí. Toto poradie je kľúčové pre identifikáciu výstupu z procesu rozpoznávania, kedy je návratová hodnota reprezentovaná poľom reálnych čísiel percentuálne prezentujúcich istotu, že rozpoznávaný záznam je identifikovaný ako daný cvik/aktivita.

Z dôvodu rozpoznávania opakujúcich sa pohybov je nutné taktiež stanoviť približný alebo priemerný počet záznamov akcelerometra pre jedno opakovanie pohybu, čo umožní rozpoznávanie dát v zariadení v dávkach. Táto informácia predstavuje 1 celé číslo, ktoré je možné pridať k informáciám o dostupných cvikoch pre rozpoznávanie z hľadiska prenosu dát po spracovaní z desktop-ovej komponenty.

5.3 Užívateľské prostredie



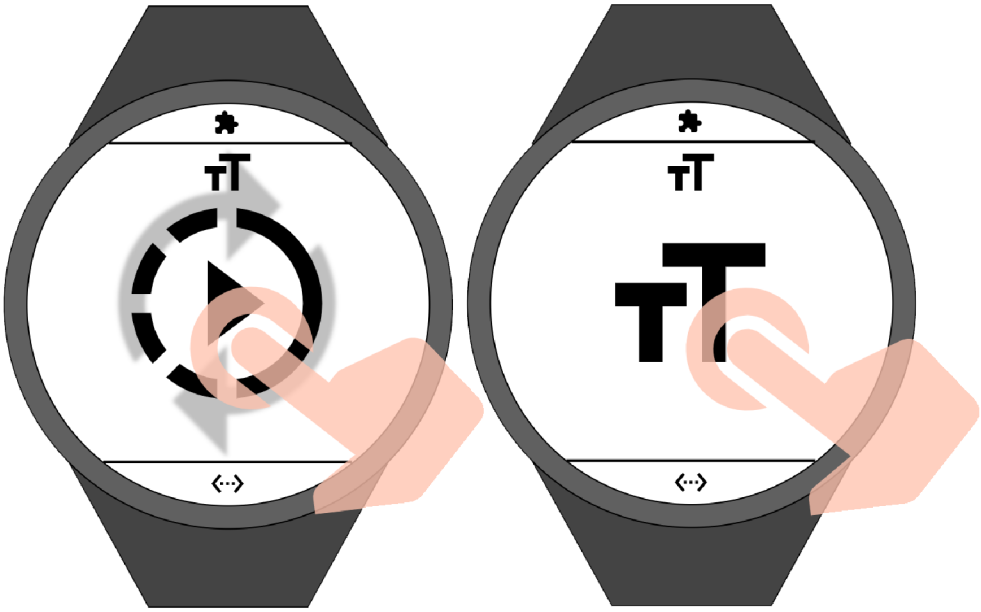
Obr. 5.2: Grafická demonštrácia interakcie s užívateľským prostredím aplikácie zložená z niekoľkých funkčných častí. Menu komponent aplikácie dostupné po potiahnutí prstu z vrchnej časti displeja nadol, umožňujúci zmenu požadovanej funkcionality/zobrazovaných informácií pomocou gesta potiahnutia prstu z jednej strany zariadenia do druhej. *Viewport* označujúci práve zobrazovaný obsah s možnosťou posunu, kde sa nachádzajú prvky interakcie priamo s obsahom. Spodné vyťahovacie menu umožňujúce aplikovať operácie súvisiace so zobrazovaným obsahom po potiahnutí prstu nahor. Prevzaté z [8].

Na základe predchádzajúceho rozdelenia celkovej zamýšľanej funkcionality na dve separátne aplikácie bolo nutné zadefinovať užívateľské prostredia pre každú komponentu zvlášť v kontexte použitých vývojových nástrojov pre jednotlivé moduly.

5.3.1 Android Wear

Aplikácia pre nositeľné zariadenia bežiaca na systéme Android Wear a návrh jej užívateľského prostredia vychádzal z doporučení a pokynov od spoločnosti Google [8]. Tie na základe obmedzení menšieho množstva zobraziteľných informácií definujú komponenty vyťahovacích okien, ktoré ponúkajú jednoduchý a rýchly prístup ku kontextovým menu a ovládaniu celej aplikácie.

V prípade vysúvacích menu sú predpísané spôsoby použitia, ktoré obmedzovali kustomizovanie vo väčšom rozsahu. Východiskový návrh bol obohatený iba o aplikačne špecifické ikony, ktoré poskytovali vizuálnu reprezentáciu jednotlivých položiek umožňujúcu jednoduchšiu orientáciu užívateľa.



Obr. 5.3: Grafický návrh užívateľského prostredia využívajúci 3 základné komponenty a to spodné a vrchné menu bez zmeny v porovnaní s pokynmi pre návrh a tvorbu UI [8] a komponentu zobrazenia informácií, ktorá obsahuje interaktívne tlačidlo s grafickou indikáciou v návrhu na ľavo a interaktívny text možný prepínania kontextu textových dát v návrhu na pravo.

Na základe týchto doporučení bolo možné vytvoriť špecifický návrh len v prípade obsahového okna označeného na obrázku 5.2 ako *Content View*. Keďže aplikácia má poskytovať jednoduché, ľahko dostupné a prehľadné prvky, na ovládanie bolo zvolené jednoduchého tlačidlo prípadne textový výstup zaberajúci väčšinu zobrazovanej plochy. Tento prístup umožňoval zvýšenú viditeľnosť a minimalizáciu problémov s interakciou počas aktivít, pri ktorých je aplikácia najviac využívaná a užívateľ nie je vo väčšine prípadov schopný extrémne presných dotykov.

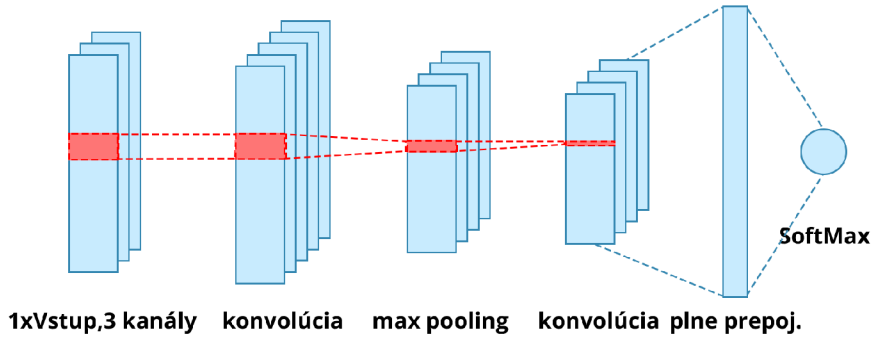
5.3.2 Desktop

Aplikačná komponenta umožňujúca parsovanie dát a následné trénovanie neurónovej siete bola zamýšľaná ako nástroj určený pre vývojárov a nie bežného užívateľa práve z dôvodu vyššej komplexnosti a požadovaných znalostí. Tento dôvod a taktiež neustále prebiehajúci vývoj a pridávanie funkcionality prispel k rozhodnutiu využitia textového užívateľského prostredia v prípade tohto nástroja.

Textové užívateľské prostredie minimalizuje zásahy do ovládania v prípade zaradenia novej funkcionality a rýchlejšie integrovanie novej verzie do procesu bez nutnosti komplexnej úpravy a možného premiestňovania jednotlivých komponent v kontexte grafického prostredia.

5.4 Model neurónovej siete

Na základe predchádzajúcej analýzy v sekcii 3.1 je použitý model založený na konvolučných neurónových sieťach. Návrh štruktúry a použitých prvkov vychádza z dostupných funkčných modelov, u ktorých sa osvedčila správnosť a efektívnosť v rozpoznávaní v podobných úlohách ako v prípade tejto práce.



Obr. 5.4: Diagram zobrazujúci štruktúru použitej neurónovej siete, skladajúcu sa z 1D vstupu o 3 kanáloch (pre každú zložku farby RGB), na ktorý je v prvom kroku aplikovaná konvolúcia, jej výsledok je následne spracovaný operáciou *max pooling* a následne znovu aplikovaná konvolúcia. Výsledok poslednej operácie je následne privedený na plne prepojenú vrstvu a celkový výsledok je spracovaný funkciou *SoftMax*.

Špecifikovanie a použitie presných detailov jednotlivých operácií je následne rozobrané v sekcii 6.

5.5 Algoritmy

Pre správnu analýzu dát nazbieraných z nositeľného zariadenia bolo vhodné tieto dáta upraviť a predpripraviť na samotné spracovanie neurónovou sieťou. Bolo identifikovaných niekoľko možných optimalizácií dát, ktoré mohli byť aplikované:

- Odfiltrovanie počiatočného a koncového šumu pri interakcii s nositeľným zariadením za účelom začatia a ukončenia záznamu aktivity/pohybu.

- Odbremenenie užívateľa od nutnosti zadávania počtu opakovaní cviku/pohybu, pre účely spríjemnenia interakcie s aplikáciou pre záznam, čo vyžaduje podpornú logiku umožňujúcu odhad dĺžky periódy na základe vstupných dát.
- Grafová vizualizácia nazbieraných dát pre účely analýzy supervízorom/človekom interagujúcim s nástrojom určeným pre samotný tréning neurónovej siete.
- Obrazová reprezentácia dát pre vyššiu úroveň porozumenia, objasnenia a využitia konvolučných neurónových sietí.

5.5.1 Filtrovanie vstupných dát

Prvotný návrh filtrovania šumu v dátach vychádzal z automatickej identifikácie a odstránenie, bez nutnosti interakcie s užívateľom obsluhujúcim nositeľné zariadenie alebo nástroj na spracovanie nazbieraných dát. Avšak autonómne filtrovanie by si vyžadovalo veľmi špecifickú implementáciu s veľkou pravdepodobnosťou využitia dodatočnej neurónovej siete, čo nakoniec viedlo k voľbe triviálnejšieho riešenia tohto problému.

Filtrovanie zachytených dát bolo z hľadiska efektivity rozdelené na 2 časti:

1. **Filtrovanie začiatku procesu záznamu** prebiehalo priamo v aplikácii určenej pre záznam dát na nositeľnom zariadení. Bolo vhodné nielen graficky ale i haptickou odozvou, ktorá predstavovala časový odpočet a následné upozornenie užívateľa pomocou vibrácie pre inicializáciu vykonávania daného pohybu. Týmto spôsobom bolo možné dosiahnuť veľmi nízkeho šumu na začiatku záznamu z dôvodu poskytnutia užívateľovi dostatku času na pripravenie sa do správne pozície a plynulému začiatku cviku.

   Po niekoľkých testoch a analýze nazbieraných dát bol identifikovaný krátky reakčný úsek, kedy bol užívateľ upozornený haptickou odozvou, no samotná reakcia a kontrakcia svalových skupín spôsobila dodatočný šum. Tento úsek vo väčšine prípadov predstavoval rozmedzie od 0,2-0,3 sekundy, ktorý musel byť dodatočne odfiltrovaný. Odstránenie tohto počiatočného úseku bolo premiestnené z nositeľného zariadenia kvôli výskytu nekonzistencií v zázname od rôznych užívateľov a ich odlišnej reakčnej doby.

2. **Filtrovanie koncovej fáze záznamu** prebiehalo v nástroji určenom na spracovanie dát a tréning neurónovej siete, ktorý poskytuje možnosť vizualizácie dát a teda lepší odhad počtu záznamov, ktoré sú identifikované ako koncový šum. Umiestnenie tejto logiky bolo zvolené i z kapacitných dôvodov, kedy v prípade že by bolo implementované v nositeľnom zariadení, funkcionalita by si vyžadovala cache-ovanie dát pred zápisom, čo by negatívne ovplyvňovalo i tak veľmi obmedzenú pamäť RAM.

   Vo väčšine testovacích subjektov a ich dát 7 bolo identifikované rozmedzie medzi 2,5 - 2,7 sekúnd ako čas odpovedajúci koncovému šumu, kedy užívateľ privádza nositeľné zariadenie do polohy kedy s ním môže interagovať a následne ukončí celý proces záznamu.

5.5.2 Odhad dĺžky periódy

Nadväzujúc na predikcie optimalizácií zo sekcie 5.5 bol, pre zvýšenie užívateľského komfortu, odstránený krok kedy užívateľ musel zadať počet opakovaní zaznamenaného cviku ihneď po jeho ukončení. Táto skutočnosť si však vyžadovala analýzu vstupných dát a určenie počtu opakovaní v priebehu procesu spracovávania.

Keďže typ dát, ktorý je zaznamenávaný má charakteristiky periodického diskrétneho signálu, je možné vo väčšine prípadov analyzovať túto radu záznamov a nájsť tak periódu [2]. Prerekvizitou spracovania a presného odhadu je vyfiltrovanie všetkého prebytočného šumu, ktorý bol identifikovaný v podsekcii 5.5.1.

Samotný odhad periódy prebieha pre každú os akcelerometra zvlášť, čo umožňuje presnejší odhad v prípade ôs, ktoré zaznamenali len minimálny pohyb alebo postrádajú identifikovateľnú periódu. Proces je zložený z niekoľkých operácií:

1. **Normalizácia dát**, ktorá umožní jednoduchšiu analýzu.
2. **Diskrétna Furierova transformácia**, pomocou ktorej prebehne odhad periódy.
3. **Odfiltrovanie nesprávnych odhadov**, v prípade ak neboli dáta „dostatočne“ periodické.
4. **Spriemerovanie valídnych výsledkov**, ktorých výsledkom je množstvo záznamov odpovedajúce jednému opakovaniu cviku/pohybu.

Druhou alternatívou kroku č.2, kde estimácia periódy prebieha pomocou diskrétnej Furierovej transformácie, bolo použitie *autokorelácie*[6]. Tento prístup sa však pri testovaní neosvedčil a výsledky boli často veľmi nepresné.

5.5.3 Metóda krížovej validácie

Pre účely trénovania a testovania efektivity a správnosti rozpoznávania neurónovej siete bola použitá metóda anglicky nazývaná *Holdout*, ktorá rozdeľuje dáta určené pre tréning neurónovej siete do 2 skupín [24].

1. Kategória určená na **trénovanie** neurónovej siete obsahujúca väčšinu dostupných dát, ktoré sú využívané priamo na proces učenia sa siete.
2. Menšia časť dát slúžiaca na **testovanie** a vyhodnocovanie presnosti a správnosti detekcie.

Týmto spôsobom je možné overiť presnosť rozpoznávania pomocou dát, ktoré pri tréningu siete neboli dostupné a jedná sa teda o novú vzorku.

Zväčša tento proces prebieha rozdelením dát na niekoľko skupín kde je jedna náhodná skupina odobraná z dát a označená ako vzorka pre následné testovanie. Tento proces je realizovaný vždy s iným náhodným rozložením, tak aby sa predišlo akejkoľvek favoritizácii segmentu dát, ako napríklad, výber vždy posledného vzorku dát, v ktorom užívateľ vykonal niekoľko predchádzajúcich opakovaní cviku a svalová únava či čiastočné vyčerpanie ovplyvňuje rýchlosť a presnosť pohybu oproti prvým opakovaniam.

[6] https://www.mathworks.com/help/signal/ug/find-periodicity-using-autocorrelation.html2

Kapitola 6

Implementácia aplikácie

Hlavným zámerom tejto práce je implementácia aplikácie umožňujúcej rozpoznávania pohybov, aktivít alebo cvikov vykonávaných užívateľom nositeľného zariadenia za pomoci strojového učenia a neurónových sietí. Nemenej dôležitou úlohou aplikácie je taktiež záznam a predpríprava dát určených pre tréning neurónovej siete za účelom zvyšovania celkovej presnosti pri detekcii spomínaných aktivít.

Operačný systém Android Wear, pre ktorý je aplikácia určená, má v dobe písania práce najväčší podiel na trhu[1] s nositeľnými zariadeniami a inteligentnými hodinkami. Samotná implementácia aplikácie a použité programovacie jazyky vychádzajú z návrhu v kapitole 5 a boli zvolené nasledovne:

1. Jazyk **Java for Android**, ktorý je základom aplikácie pre záznam dát a následnú detekciu s použitím natrénovanej neurónovej siete zo zachytených dát.

2. Jazyk **Python verzie 3** s knižnicou **TensorFlow** umožňujúci spracovanie zaznamenaných dát z nositeľného zariadenia a ich použitie za účelom tréningu neurónovej siete navrhnutej pomocou danej knižnice.

6.1 Android Wear

Aplikácia pre nositeľné zariadenia bola navrhnutá i implementovaná na základe doporučení z [15] za pomoci 2 komponent reprezentujúcich moduly poskytujúce špecifickú funkcionalitu:

1. Komponenta pre **záznam dát** určených pre tréning neurónovej siete.

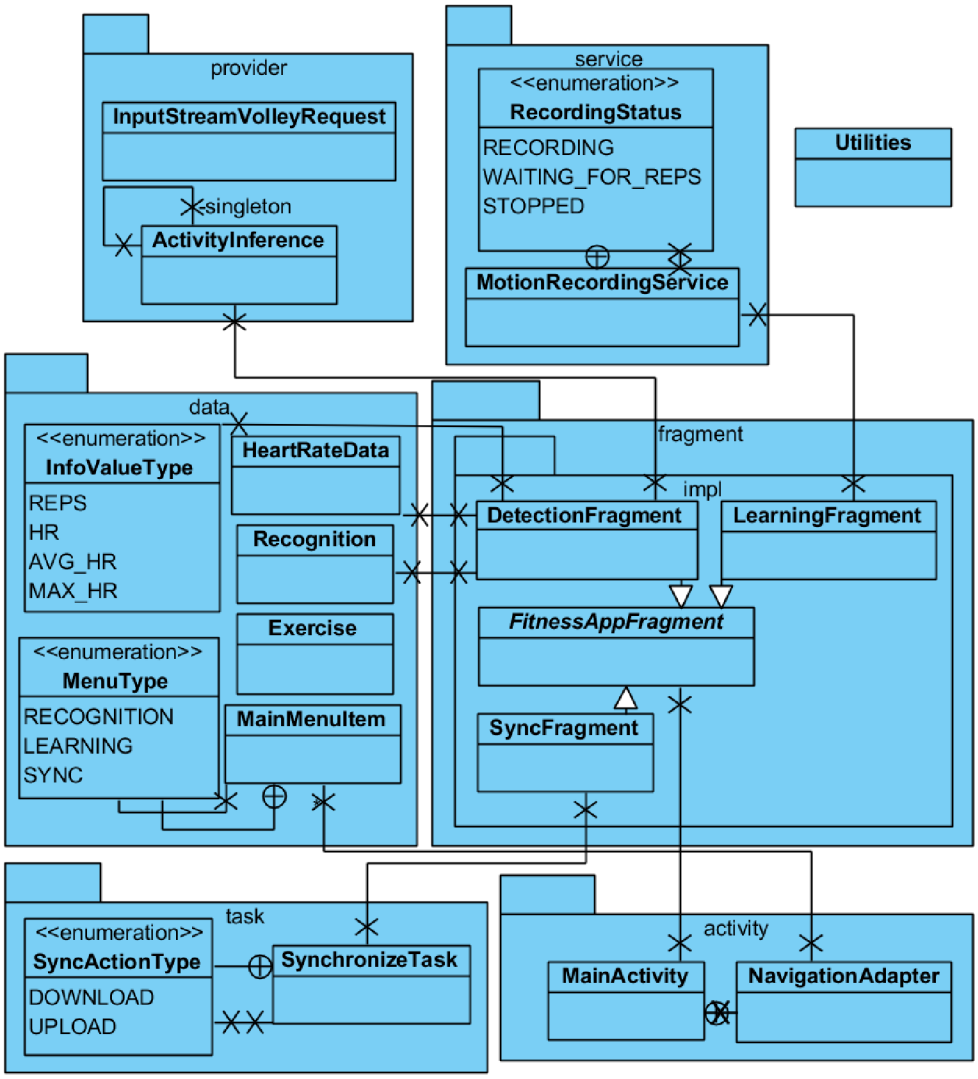
2. Komponenta na **detekciu pohybov** za pomoci natrénovanej neurónovej siete.

Samotné komponenty sú obsiahnuté v jednej samostatne bežiacej aplikácií, kde výber a voľba režimu/komponenty, ktorú si užívateľ zvolil prebieha za pomoci sťahovacej ponuky z vrchnej časti grafického prostredia *WearableNavigationDrawerView*[2]. Užívateľ tak má možnosť si zvoliť iný režim aplikácie v ktoromkoľvek kroku bez nutnosti návratu.

Obrázok 5.2 zobrazuje základnú ideu interakcie a zvolené komponenty. Samotná funkcionalita jednotlivých obrazoviek je následne dostupná v *Content View*, s ktorým užívateľ vo väčšine času interaguje.

[1] https://www.statista.com/statistics/466563/share-of-smart-wristwear-shipments-by-operating-system-worldwide/

[2] https://developer.android.com/reference/android/support/wearable/view/drawer/WearableNavigationDrawer



Obr. 6.1: Vizualizácia tried a ich závislostí v kompozícii balíčkov pre Android Wear aplikáciu reflektujúca implementovaný kód.

Súčasťou každej obrazovky v aplikácii je taktiež spodné vysúvacie menu *WearableActionDrawerView*[3], ktoré obsahuje funkcionalitu priamo viazanú na zobrazovaný obsah. Dvojica dostupných menu vychádza zo základných doporučení pre tvorbu grafického rozhrania aplikácií pre operačný systém Android Wear a je popísaná v [8]. Tento prístup umožňuje maximalizovať celkovú prehľadnosť informácií na displeji v kombinácii s rýchlou a intuitívnou interakciou priamo na zápästí.

6.1.1 Zbieranie dát

Proces i spôsob záznamu nových dát určených pre následné trénovanie neurónovej siete boli navrhnuté tak, aby túto funkcionalitu mohli využívať i bežný užívatelia so zámerom zlepšenia celkových výsledkov rozpoznávania ako priameho dôsledku zvýšenia variety zá-

[3] https://developer.android.com/reference/android/support/wear/widget/drawer/WearableActionDrawerView

Obr. 6.2: Ukážka reálnych snímkov z behu aplikácie. Prvý riadok zobrazuje funkcionalitu pre zber nových dát, zľava: upozornenie užívateľa na skutočnosť, že nebol vybraný typ aktivity, ktorý bude nahrávaný pomocou textu a zošednutého, neaktívneho, tlačidla nasledované obrazovkou indikujúcou možnosť započatia záznamu. Druhý riadok ukazuje prípravný odpočet pred samotným začatím zaznamenávania a indikáciu prebiehajúceho procesu. Posledný riadok obsahuje ukážku, zľava: hlavného menu dostupného potiahnutím z vrchu a špecifické menu dostupné po potiahnutí zo spod obrazovky.

znamov a ich objektívnejšej kategorizácie. Keďže záznam dát prebieha priamo na zariadení, bolo použité natívne rozhranie pre získavanie dát zo senzorov dostupných na nositeľnom zariadení [7].

Zaznamenávaný typ dát

V prvotnom štádiu vývoja boli ako zaznamenávané dáta použité informácie priamo z akcelerometer senzora, ku ktorému systém Android implementuje prístup s možnosťou špecifikácie vzorkovacej frekvencie[4]. V prípade tohto typu dát sú však hodnoty ovplyvňované gravitačným zrýchlením v miere úmernej k polohe danej osi záznamu. Tento šum bol v neskorších fázach a testovaní identifikovaný ako problém, ktorý pri trénovaní a následnom rozpoznávaní prinášal do prostredia nepresnosť a chyby. Z tohto dôvodu bola zvolená ako náhrada lineárna akcelerácia, ktorá je o tento aspekt filtrovaná priamo v jadre operačného systému a nie je potrebné ďalšie spracovanie záznamu, čo je v prípade obmedzeného výpočetného výkonu nositeľných zariadení veľkým prínosom.

[4] https://developer.android.com/reference/android/hardware/SensorManager

Vzorkovacia frekvencia

Voľba vzorkovacej frekvencie bola limitovaná nižšou kapacitou zariadení a ako frekvencia prinášajúca ideálne výsledky z pohľadu presnosti a náročnosti na záznam bolo zvolené vzorkovanie každých 20 ms. Táto rýchlosť umožňuje zaznamenať 50 záznamov za sekundu cviku, pričom jedno opakovanie cviku zväčša trvá 1,5-3 sekundy (viď. sekcia 7.1) v závislosti od rýchlosti, náročnosti a komplexnosti, teda 75-150 záznamov z akcelerometer senzora. V prípade vyšších rýchlostí boli identifikované problémy s nepresnosťou časového rozmedzia v záznamoch takisto ako i variáciou v množstve záznamov v jednej sekunde.

Proces interakcie

Grafické prostredie pre samotnú interakciu užívateľa z obrázka 6.2 (záznam dát 1 a 2) vizuálne indikuje možnosti práce s aplikáciou. Po štarte a nabehnutí aplikácie spolu so zvolením režimu *učenia* je užívateľ vyzvaný na výber typu cviku, ktorého opakovania má v záujme zaznamenávať. Výber prebieha pomocou vysúvacieho spodného menu, bližšie popísaného v obrázku 5.2. Menu poskytuje okrem niekoľkých predvolených cvikov možnosť hlasového zadania vlastného názvu cviku. Hlasové zadávanie vychádza z natívnej podpory operačným systémom, *RecognizerIntent*[5]. Vytvorený typ cviku je nastavený ako zvolený a jeho výber je udržovaný i počas minimalizovania aplikácie, avšak nie je vložený do existujúceho zoznamu predvolených cvikov ani menu samotného. Existuje teda len po dobu behu aplikácie (i na pozadí).

Po výbere cviku pre záznam je graficky i logicky povolená interakcia s tlačidlom pre začatie záznamu (obrázok 6.2 záznam dát 1). Po kliknutí na aktívne tlačidlo je zobrazený dialóg[6] obsahujúci informáciu o blížiacom sa začatí zaznamenávania. Informácia je delegovaná užívateľovi pomocou sekundového odpočtu, ktorý zaobstaráva implementácia *CountDownTimer*[7] upravujúca správu tela dialógu. Po skončení odpočtu je dialóg odstránený a celý proces záznamu je delegovaný príslušnej servise, bežiacej na pozadí.

Služba záznamu dát

Služba slúžiaca pre záznam dát využíva implementáciu *Background Service*[8] umožňujúcu operáciám beh na pozadí, bez zaťažovania vlákna užívateľského rozhrania. Pri inicializácii tejto servisy je vytvorený/otvorený súbor na ukladanie záznamov a zaregistrovanie na odposluch udalostí zo senzorového manažéra[4]. Samotný priebeh záznamu je realizovaný pomocou *FileOutputStream*[9] a zapisovania konkatenovaného reťazca znakov obsahujúceho informácie definované v kóde 5.1.

Služba beží na pozadí pokým je zvolený učiaci mód. Detekcia a prijímanie dát zo senzorov je aktívne len na požiadavku začatia záznamu, ktorá taktiež vyžaduje špecifikáciu názvu zaznamenávaného cviku. Na základe tejto žiadosti si následne služba zaregistruje získavanie dát z akcelerometra a povolí zapisovanie do súboru. Tento proces je aktívny pokým nepríde žiadosť o ukončenie. Tú môže vyvolať užívateľ alebo operačný systém samotný. V tej chvíle je všetko získavanie dát deaktivované/odregistrované a práca so súborom bezpečne dokončená.

[5] https://developer.android.com/reference/android/speech/RecognizerIntent
[6] https://developer.android.com/reference/android/app/AlertDialog.html
[7] https://developer.android.com/reference/android/os/CountDownTimer
[8] https://developer.android.com/guide/components/services
[9] https://developer.android.com/reference/java/io/FileOutputStream

Vizuálna a haptická odozva

Interakcia a spätná väzba užívateľovi je v aplikácii realizovaná niekoľkými spôsobmi. Grafická indikácia je smerovaná ku tlačidlu v rozhraní, ktoré má niekoľko stavov.

1. Zašednuté tlačidlo značí nemožnosť začatia záznamu, či už z dôvodu nešpecifikovaného názvy cviku alebo internej chybe aplikácie (aplikácii chýbajú povolenia, nepodarilo sa vytvoriť súbor pre zápis, či iné).

2. Plne farebné tlačidlo slúži ako notifikácia o pripravenosti aplikácie na proces záznamu.

3. Kruhový nešpecifický (anglicky *indeterminate*) *Progressbar*[10] v tesnom okolí tlačidla, ktorý indikuje práve prebiehajúci záznam.

Okrem vizuálnej spätnej väzby je užívateľ notifikovaný o začatí záznamu aj haptickou odozvou (vibrovaním). Tento typ notifikácie bol zvolený tak, aby sa užívateľ mohol pripraviť a dostať do počiatočnej pozície bez nutnosti vizuálnej interakcie s displejom.

6.1.2 Prenos nazbieraných dát

Jedna z dostupných komponent aplikácie poskytuje funkcionalitu pre nahranie doposiaľ zaznamenaných dát na vzdialený server. Tento typ prenosu bol zvolený z dôvodu zníženia celkovej náročnosti prenosu dát, keďže operačný systém Android Wear neposkytuje aplikáciu súborového manažéra alebo zdieľania dát z vnútornej pamäte. Jediný štandardný spôsob je prenos dát pomocou vývojového nástroja *adb*[11], čo by pre bežného užívateľa mohlo predstavovať čiastočný problém.

Odoslanie z nositeľného zariadenia

Zasielanie dát na vzdialený server prebieha pomocou knižnice pre Android Wear s názvom *Volley*[12], ktorá umožňuje intuitívne vytváranie požiadaviek na vzdialený *REST*[13] server. Aplikácia využíva *StringRequest* server, kde je pomocou implementácie metódy rozhrania vložené telo požiadavky vo forme textového reťazca. Samotný dotaz i vytvorenie fronty si knižnica manažuje sama, bez nutnosti implementačnej špecifikácie.

Pre úspešné nahranie dát je počas procesu vyžiadaný od operačného systému tzv. *WakeLock*[14], ktorý nepovolí zariadeniu a systému prejsť do úsporného režimu, čo by mohlo spôsobiť automatické ukončenie aplikácie.

Po ukončení procesu je užívateľ notifikovaný pomocou *Toast*[15] notifikácie so správou o výsledku akcie alebo prípadnou špecifikáciou problému, ktorý nastal.

Vzdialený PHP server

Požadovanou funkcionalitou tohto serveru bolo prevzatie dát zasielaných z nositeľného zariadenia a ich následné uloženie na lokálne úložisko serveru. Server využíva základných funkcií jazyka PHP.

[10] https://developer.android.com/reference/android/widget/ProgressBar
[11] https://developer.android.com/studio/command-line/adb
[12] https://developer.android.com/training/volley/simple
[13] https://restfulapi.net/
[14] https://developer.android.com/training/scheduling/wakelock
[15] https://developer.android.com/guide/topics/ui/notifiers/toasts

Načítanie zasielaných dát prebieha pomocou funkcie *file-get-contents*[16] a ako názov súboru je špecifikovaná I/O premenná *php://input*[17], poskytujúca prístup do vyrovnávacej pamäte obsahujúcej telo dotazu. Následne je pomocou *file-put-contents*[18] zapísaný získaný obsah do súboru formátom typu *CSV*. Jedinečnosť záznamov a eliminácia prepisovania súborov bola zabezpečená špecifikáciou názvu súboru na strane servera pomocou reťazca znakov založeného na aktuálnom čase a dátume.

Z hľadiska bezpečnosti nebolo nutné verifikovanie alebo zabezpečovanie spojenia práve z dôvodu neprítomnosti interakcie servera s dátami.

6.1.3 Rozpoznávanie pohybov

Na rozpoznávanie pohybov a detekciu opakovaní je v aplikácii dostupná dedikovaná komponenta, ktorá využíva knižničné funkcie TensorFlow [13, 20]. Detekcia za pomoci natrénovaného modelu prebieha v niekoľkých krokoch:

1. Vytvorenie objektu *Interpreter* a predanie súboru natrénovanej neurónovej siete vo formáte *.tflite*.
2. Nazbieranie dát v špecifickom rozsahu, ktorý bol definovaný počas trénovania neurónovej siete ako veľkosť vzorky.
3. Úprava dát na požadovaný formát.
4. Vykonanie inferencie neurónovou sieťou a získanie výsledkov.
5. Parsovanie a mapovanie neoznačených výsledkov do čitateľného formátu za pomoci podporných dát získaných po natrénovaní neurónovej siete.
6. Analýza výsledkov a ich porovnanie s minimálnou hodnotou pre uznanie ako opakovanie.

Nižší výpočetný výkon nositeľného zariadenia si vyžadoval využitie skokového okna pre rozpoznávanie analyzovaných dát. To znamenalo nevykonávanie inferencie pri každom príchode nových dát, ale vždy po nazbieraní určitého počtu nových. Týmto spôsobom nebol kladený príliš veľký nárok na kapacitu zariadenia a to nemalo problém s behom aplikácie.

V užívateľskom prostredí sa rozpoznávanie javí ako meniaca sa hodnota počtu opakovaní a názvu cviku, ktorý bol identifikovaný. Užívateľ má však stále plnú kontrolu nad zobrazovanými dátami a môže vidieť rôzne štatistiky okrem počtu opakovaní dostupné pomocou kliknutia na zobrazovanú hodnotu. Medzi tieto štatistiky patria:

- počet opakovaní
- aktuálna hodnota srdečného tepu
- priemerná hodnota srdečného tepu
- maximálna hodnota srdečného tepu

Všetky tieto hodnoty sú uchovávané v objekte triedy *Exercise*, kde každá inštancia objektu zdržuje štatistiky pre jeden cvik.

[16] https://www.php.net/manual/en/function.file-get-contents.php
[17] https://www.php.net/manual/en/wrappers.php.php
[18] https://www.php.net/manual/en/function.file-put-contents.php

Obr. 6.3: Ukážka reálnych snímkov z behu aplikácie. Prvý riadok zobrazuje funkcionalitu detekciu opakovaní, zľava: upozornenie užívateľa na započatú detekciu a bežiacu inferenciu nasledované informáciou o úspešne rozpoznanom opakovaní s informáciou o stupni istoty z poslednej inferencie (nie posledného úspešného rozpoznania) . Druhý riadok ukazuje menu dostupné potiahnutím zo spod displeja umožňujúce začatie a ukončenie detekcie a tiež obrazovku informujúcu o priemernom srdečnom tepe od počiatku aktuálnej detekcie.

6.2 Aplikácia pre desktop

Základným stavebným prvkom aplikácie na parsovanie dát zaznamenaných na nositeľnom zariadení a ich následné použitie pre proces trénovania neurónovej siete bol programovací jazyk *Python*[19] verzie 3 a softwérovej knižnice TensorFlow [1] umožňujúcej vytvorenie a prácu s neurónovou sieťou.

Aplikácia sa skladá z niekoľkých logicky rozdelených modulov poskytujúcich špecifickú funkcionalitu:

- *main.py* zodpovedný za základnú logiku a komunikáciu s ostatnými modulmi
- *cnn.py* obsahujúci logiku zodpovednú za interakcie s neurónovou sieťou a jej vytvorenie
- *clickable_plot.py* poskytujúci funkcionalitu interaktívneho grafu používaného pre manuálne filtrovanie a identifikáciu periódy v záznamoch
- *globals.py* udržujúci zdielané informácie medzi modulmi

6.2.1 Parsovanie zaznamenaných dát

Po získaní dát z nositeľného zariadenia sú dáta prevedené pomocou knižničných funkcií *pandas*[20] umožňujúcich parsovanie *CSV* formátovaného súboru.

[19] https://www.python.org/
[20] https://pandas.pydata.org/

Kód 6.1: ukážka nápovedy z aplikácie popisujúca funkcionalitu

```
usage: convert.py [-h] -i FILE [-o FILE] [-m FILE] [--length LENGTH]
                  [-v] [--image] [-f] [--estimations] [--auto-filter]

Process input CSV file of recorded activities and trains CNN or visualize parsed
 data.

optional arguments:
  -h, --help            show this help message and exit
  -i FILE, --input FILE
                        CSV formatted file
  -o FILE, --output FILE
                        Trained neuron network prefix name
  -f FILE, --filter-input FILE
                        Information about repetitions in dataset
  --export-filter       Exports data from manual filtering and separating.
  -m FILE, --model FILE
                        Previously trained neuron network prefix name
  --length LENGTH       Length of single sample, allowed values : 'max','avg'
                        or ANY NUMBER
  -v, --visualize       Show plots of each recorded exercise
  --image               Show image of each recorded exercise chunk
  --fourier             Show plots of each DFT used for period estimation
  -g, --graph           Exports graph designed for TensorBoard
  --estimations         Show estimations(num. of records) of each activity
  --auto-filter         Filter start and end of the recording and estimate
                        period automatically
```

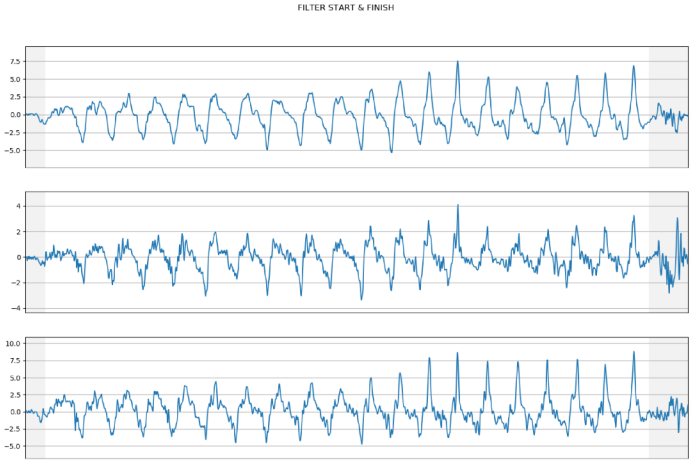
Po načítaní, sú dáta rozdelené na základe časovej značky, tak aby každý blok dát zastupujúci jeden cvik v jednom časovom úseku bol separovaný zvlášť. Týmto spôsobom je možné lepšie automaticky detekovať periódu a počet opakovaní v zázname. V prvotnej verzii bol na rozdelenie dát použitý len názov cviku, ktorý však prinášal problémy s filtrovaním počiatku a konca záznamov kvôli odlišnostiam v dĺžke periódy jednotlivých datasetov.

Program poskytuje automatické a manuálne filtrovanie a separovanie jednotlivých opakovaní v záznamoch.

Clickable plot

Tento modul poskytuje funkcionalitu na manuálnu interakciu so zaznamenanými dátami vo forme grafu. Podľa zvoleného typu grafu je možné filtrovať počiatok a koniec alebo určovať začiatky periódy/identifikácie jedinečných opakovaní v dátach.

Každý blok dát reprezentujúci jeden časový úsek jedného cviku je vyobrazený za pomoci 3 grafov (pre každú os akcelerometra), kde os x vyobrazuje čas a os y hodnoty lineárnej akcelerácie.



Obr. 6.4: Ukážka interakcie a manuálneho filtrovania počiatku a konca dát pre účely odstránenia šumu. Filtrovanie umožňuje odstrániť šum vzniknutý nepredvídateľným pohybom po začatí záznamu a pred ukončením, ako napríklad kliknutie na ukončenie záznamu alebo reakčný čas po vibračnej odozve zariadenia indikujúcej začiatok. Interakcia s grafom prebieha pomocou pc myši, kedy ľavé tlačidlo slúži na označenie bodu, po ktorý sa majú dáta filtrovať (klik v 1. polovici označuje nový začiatok, klik v 2. polovici označuje koniec, a pravé tlačidlo slúži na potvrdenie výberu.

Automatická detekcia a parsovanie

Prvotne navrhnutá funkcionalita aplikácie využívala plne automatizované filtrovanie šumu na počiatku a konci záznamov spolu detekciou periódy a rozdelením dát na individuálne opakovania. Tento prístup sa však počas testovania v prípade veľkej diverzity záznamov neosvedčil ako efektívny z dôvodov bližšie analyzovaných v kapitole 7. Funkcionalita bola vo finálnej verzii upravená na voliteľnú, viď. kód 6.1.

Filtrovanie počiatku a konca záznamov bolo realizované pomocou časovej konštanty, ktorá vychádzala z analýzy v 7. Hodnoty, ktoré sa javili ako najčastejší reakčný čas a čas pre interakciu so zariadením pre ukončenie boli 0,3 a 2,7 sekundy. Z toho dôvodu bolo realizované orezanie analýzou časového razítka jednotlivých záznamov.

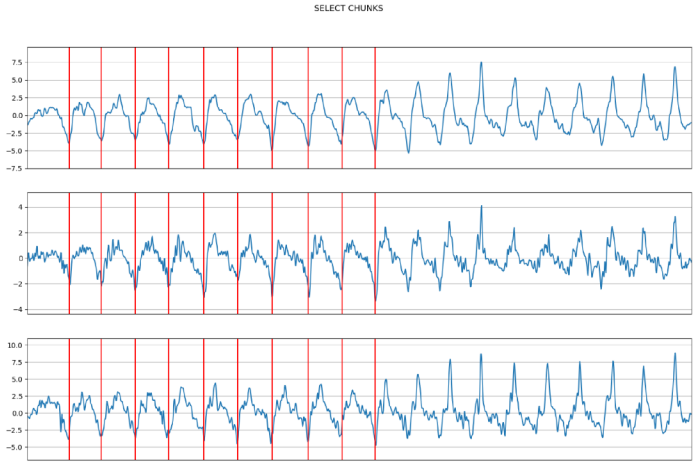
Implementácia logiky zodpovednej za estimáciu je založená na analýze diskrétnej Furierovej transformácie (DFT[21]) aplikovanej na normalizované dáta a následne porovnané so špecifickým prahom. Normalizácia dát prebieha pomocou aplikovania rovnice

$$(x - \bar{x})/\sigma_x \tag{6.1}$$

teda aritmetického priemeru $mean$[22] a štandardnej odchýlky std[23].

Ako alternatíva k DFT sa naskytovala $Autokorelácia$[24]. Pri testovaní jej presnosti sa však neosvedčila ako dostatočne účinná a presná z pohľadu komerčného použitia.

[21] https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html
[22] https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html
[23] https://docs.scipy.org/doc/numpy/reference/generated/numpy.std.html
[24] https://en.wikipedia.org/wiki/Autocorrelation



Obr. 6.5: Ukážka interakcie a manuálneho identifikovania jednotlivých opakovaní/periód v zázname. Hlavnou úlohou je rozdeliť vyobrazené dáta na jednotlivé úseky, ktoré reprezentujú jedno opakovanie v čase. Interakcia prebieha pomocou tlačidiel pc myši, kde ľavé tlačidlo slúži na vloženie rozdelenia. Užívateľ má voľnosť kliku na ktorýkoľvek z 3 grafov. Pravé tlačidlo má dve funkcie. Pri kliku na už vložené rozdelenie toto rozdelenie odstráni a pri kliku mimo graf potvrdí, uloží a ukončí prácu s grafom.

6.2.2 Tréning neurónovej siete

Výsledok operácií definovaných v podsekcii 6.2.1 je možné následne využiť pre samotný tréning neurónovej siete. Kombináciou filtrovaných dát a odhadu/definovania počtu periód je možné dáta rozdeliť na segmenty a následne poskytnúť spolu s označeniami typu cviku neurónovej sieti. Formát dát, ktoré vyžaduje neurónová sieť, je však veľmi špecifický a z tohto dôvodu musia byť pred samotným procesom upravené.

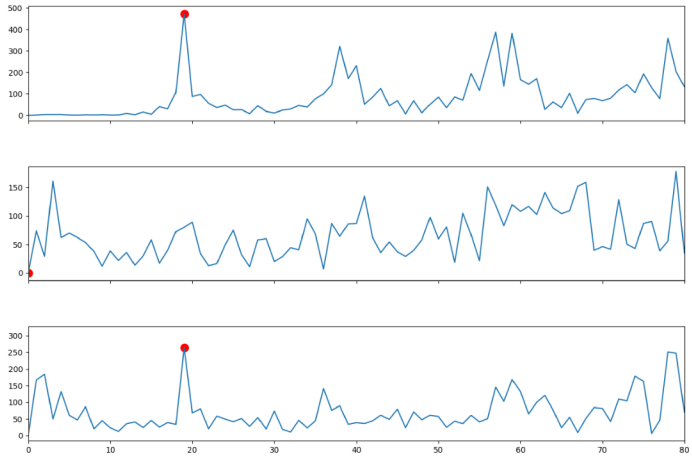
Formátovanie dát pre neurónovú sieť

Využitím detekovaného/definovaného počtu periód je možné dáta rozdeliť na jednotlivé segmenty/opakovania viď. obrázok 6.6. Neurónová sieť vyžaduje všetky exempláre dát v rovnakej veľkosti a formáte, čo si vyžaduje ich prevzorkovanie alebo doplnenie na špecifickú dĺžku. K tomuto procesu je využívaná funkcia *resample*[25] a *pad*[26].

Kód 6.2: Formát dát určený pre proces tréningu neurónovej siete definovaný ako pole N polí o veľkosti 3 (osi akcelerometra), kde každý záznam vnútorného poľa inkrementálne reprezentuje dáta v čase.

```
[
    [x1, y1, z1],
    [x2, y2, z2],
    [x3, y3, z3],
    [.., .., .],
    [xn, yn, zn]
                ]
```

[25] https://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.signal.resample.html
[26] https://docs.scipy.org/doc/numpy/reference/generated/numpy.pad.html



Obr. 6.6: Analýza parsovaných dát za pomoci diskrétnej Furierovej transformácie a následné porovnanie s prahovou hodnotou (hodnota na obr. 225) umožňuje detekovať a predpovedať pravdepodobnosť počtu období nachádzajúcich sa v zázname. Osa x vyobrazuje počet opakovaní a osa y absolútne hodnoty *DFT*. Červený bod zvýrazňuje úsek v zázname, kde bola nájdená hodnota vyššia ako definovaný prah. Po analýze grafu je možné identifikovať, že v analyzovanom zázname je podľa tejto metódy identifikovaných 20 periodicky sa opakujúcich segmentov, teda 20 opakovaní cviku. V prípade osi Y akcelerometra je možné vidieť, že sa algoritmu nepodarilo určiť periódu. Dáta s príliš nízkym alebo vysokým odhadom sú v takom prípade anulované a vyradené z predikcie.

V prípade štítkov označujúcich typ cviku pre špecifické dáta je nutné previesť textové reťazce na binárne pole. Tohto výsledku je dosiahnuté pomocou *get_dummies*[27]. Výsledkom je binárne číslo o minimálnej dĺžke schopné každou z hodnôt reprezentovať jeden z typov cvikov.

Vo výstupných formátoch definovaných vyššie je možné dáta priamo odovzdať neurónovej sieti, ktorá môže byť na nich následne trénovaná.

Vrstvy CNN

Implementácia jednotlivých vrstiev je realizovaná pomocou knižničných funkcií dostupných z [21]. Boli použité primárne funkcie pre *max_pool*, *depthwise_conv2d*, *Variable*, *tanh*, *softmax* a *reduce_mean*.

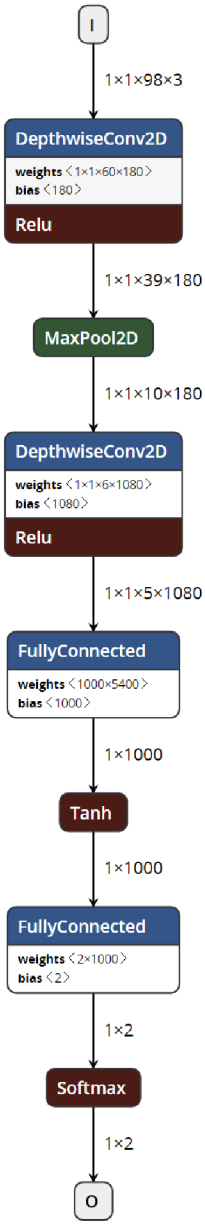
Priebeh trénovania

Trénovanie neurónovej siete prebieha v niekoľkých epochách, pri ktorých prejde celý dataset neurónovou sieťou tam aj späť. V navrhnutom tréningovom modeli boli použité 4 tréningové epochy, pri ktorých boli dosiahnuté najlepšie výsledky a minimalizovaný jav nazývaný *overfitting*[28].

V prípade veľkosti dávok počas procesu tréningu, bola zvolená veľkosť 10 záznamov pre každú várku.

[27] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html
[28] https://elitedatascience.com/overfitting-in-machine-learning

V prvotnom kroku je neurónová sieť trénovaná na časti dát určenej pre tento proces. Po úspešnom skončení tejto časti je využitá druhá časť dát na overenie účinnosti a správnosti rozpoznávania.



Obr. 6.7: Graf vizualizujúci vrstvy a logiku na pozadí použitej neurónovej siete. Využívaných je niekoľko matematických funkcií, ako *Add*, *Relu*, *MaxPool*, *Prevzorkovanie* a *Tanh*, viď. 3.1. Na vytvorenie grafu, ktorý vychádza z kompletne natrénovanej siete, vyexportovanej do formátu *.tflite* bol použitý open-source nástroj *Netron*[29].

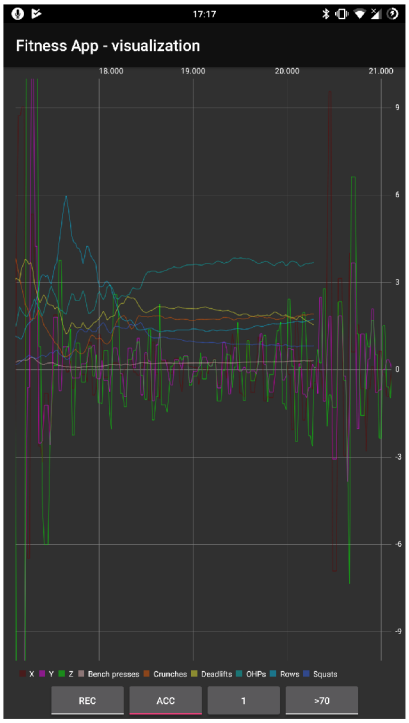
[29] https://github.com/lutzroeder/netron

Verifikácia CNN a účinnosť

Pred spustením samotného procesu trénovania neurónovej siete sú všetky dáta náhodne rozdelené pomerom 7 ku 3, kedy 7 častí je použitých na trénovanie/učenie sa a 3 časti sú po úspešne ukončenom tréningu použité dodatočne z dôvodu verifikácie/otestovania presnosti rozpoznávania na dátach, s ktorými sieť doposiaľ neinteragovala. [3]

Tento prístup nám umožňuje detekciu nepresností, pretrénovania a iných problémov spôsobujúcich nekorektnosť detekcie siete.

6.3 Testovacia aplikácia



Obr. 6.8: Modul aplikácie pre smartphone, ktorý okrem základnej funkcionality, ktorú ponúka modul pre nositeľné zariadenia, umožňuje vizualizovať dáta z akcelerometra a výsledky inferencie neurónovej siete vo forme časovo závislého grafu. Os y zobrazuje hodnoty akcelerometra a hodnoty výslednej inferencie v percentách namapované na interval $\langle 0{,}10 \rangle$. Užívateľ je schopný taktiež vypnúť zobrazovanie jednotlivých štatistík separátne alebo nastaviť koeficient modulácie dát.

Počas testovania účinnosti rozpoznávania dát a celkovej korektnosti aplikácie bol jednou z najväčších výziev obmedzený výkon nositeľných zariadení, ktorý sťažoval možnosť ladenia kódu a sledovania štatistík v ňom počas behu aplikácie. Ukázalo sa, že zariadenie nemá dostatočný výkon na simultánny beh záznamu, parsovania dát spolu s inferenciou neurónovej siete a dodatočné paralelné zasielanie štatistík do *adb*[30] rozhrania.

Z dôvodu tohto obmedzenia vznikla podporná aplikácia pre zariadenia s vyšším výkonom, smartphone, ktoré umožňovali takýto simultánny beh práve kvôli vyššiemu celko-

[30] https://developer.android.com/studio/command-line/adb

vému výkonu. Keďže bola aplikácia vyvíjaná pre systém Android Wear, ktorý je súčasťou ekosystému Android, portácia kódu zo základného modulu pre nositeľné zariadenia bola bezproblémová. Implementačné detaily sú teda medzi modulmi vo veľkej miere zdieľané.

Okrem základnej funkcionality, ktorú ponúka aplikácia pre Wear, bola pridaná i vizualizačná zložka umožňujúca lepšie pochopenie správania sa neurónovej siete a jej výsledkov na základe zaznamenávaných dát.

Implementácia vizualizačnej zložky bola uskutočnená pomocou open-source knižničných funkcií *MPAndroidChart* dostupných z [11], kde je komplexný vykresľovací proces funkčne závislý na tejto knižnici. Základná logika spracovania zaznamenaných dát a inferencie vychádza z implementačných detailov 6.1.

Kapitola 7

Testovanie

Pre overenie správnej funkcionality aplikácie ako celku bolo nutné overiť funkčnosť a účinnosť každej z častí procesu individuálne i ako celok:

- intuitívnosť rozhrania pre záznam dát
- overenie správnosti získaných dát
- správnosť parsovania dát
- korektnosť filtrovania a odhadu periódy
- presnosť neurónovej siete na základe testovacích dát
- presnosť neurónovej siete použitej vo výslednej aplikácií

7.1 Zbieranie dát

Na ponúknutie aplikácie pre využitie širokou verejnosťou je nutné pripravenie neurónovú siete v čo najobjektívnejšom smere. Toho je možné dosiahnuť tréningom na dátach o dostatočnom počte vzoriek od väčšej variety užívateľov.

Práve z nutnosti získania dát od čo najväčšieho počtu užívateľov si vyžadoval aplikačný modul pre záznam dát optimalizovanie pre čo najintuitívnejšie zaobchádzanie bez nutnosti dodatočných návodov a inštrukcií.

V prvotných fázach bola aplikácia a jej zámer predstavený užšej skupine užívateľov a následne im bolo zariadenie s aplikáciou bez predchádzajúcej inštruktáže zapožičané. Následne bol subjekt požiadaný o vytvorenie niekoľkých záznamov, ktoré by bolo možné použiť pre trénovanie neurónovej siete. Po ukončení testu bola skolektivizovaná spätná väzba od užívateľov zameraná na intuitívnosť rozhrania, problémy a návrhy na zlepšenie používania. Nižšie je možné vidieť návrhy, ktoré boli akceptované ako relevantné (viazali sa k dizajnu aplikácie a nie k samotnému systému Android Wear) spolu s ich riešením:

1. užívateľ musí sledovať displej pre informáciu o začatí záznamu - pridaná vibračná odozva
2. čas medzi stlačení tlačidla v UI a samotným záznamom príliš krátky (3s) - časové okno zvýšené na 6s

3. dodatočné zadávanie počtu vykonaných opakovaní - vyradenie zobrazovania výberu počtu opakovaní (implementácia automatickej detekcie periód v 6.2.1)

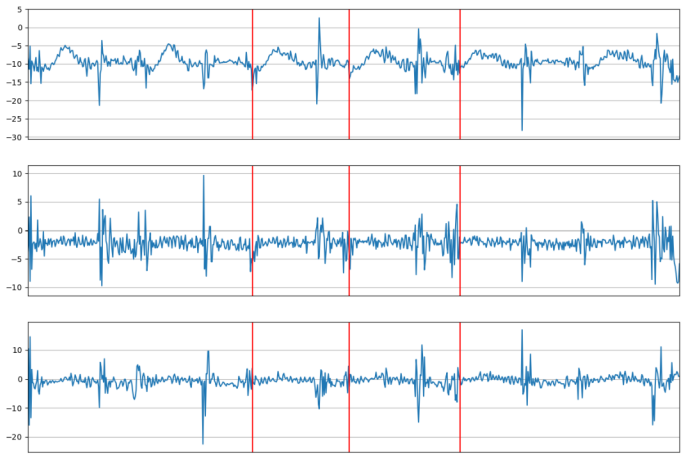
4. pop-up notifikácie resetujúce zvolený typ cviku - persistencia zvoleného cviku i pri minimalizovaní aplikácie

Po zapracovaní spätnej väzby od užívateľov sa celková intuitívnosť a dojem aplikácie podstatne zlepšili na úroveň umožňujúcu nasadenie do produkcie.

7.2 Parsovanie dát

Základným predpokladom parsovania dát bol predom určený počet opakovaní v spracovávaných záznamov. Po testovaní zberu dát v 7.1 bolo nutné vyvinúť funkcionalitu pre automatické detekovanie počtu opakovaní, viď. 6.2.1. Dáta boli v ďalších fázach rozdeľované za pomoci tohto nástroja.

Po zozbieraní väčšieho počtu rozličných dát od viacerých užívateľov boli pomocou vizualizačných nástrojov odhalené problémy automatickej detekcie.



Obr. 7.1: Graf zobrazujúci rozdielnu dĺžku jednotlivých opakovaní v jednom zázname. Dôvodom bola rozdielna rýchlosť vykonávania jednotlivých opakovaní, vo väčšine prípadov spôsobená únavou. Táto inkonzisencia spôsobila problémy pri rozdeľovaní záznamu na jednotlivé opakovania, ktoré priamo súviseli so zníženou presnosťou rozpoznávania natrénovanej neurónovej siete.

Zistenie z 7.1 ovplyvnilo vývoj aplikácie a umožnilo vzniku pridanej funkcionality pre manuálne rozdelenie a filtrovanie dát pomocou interaktívneho grafu 6.2.1.

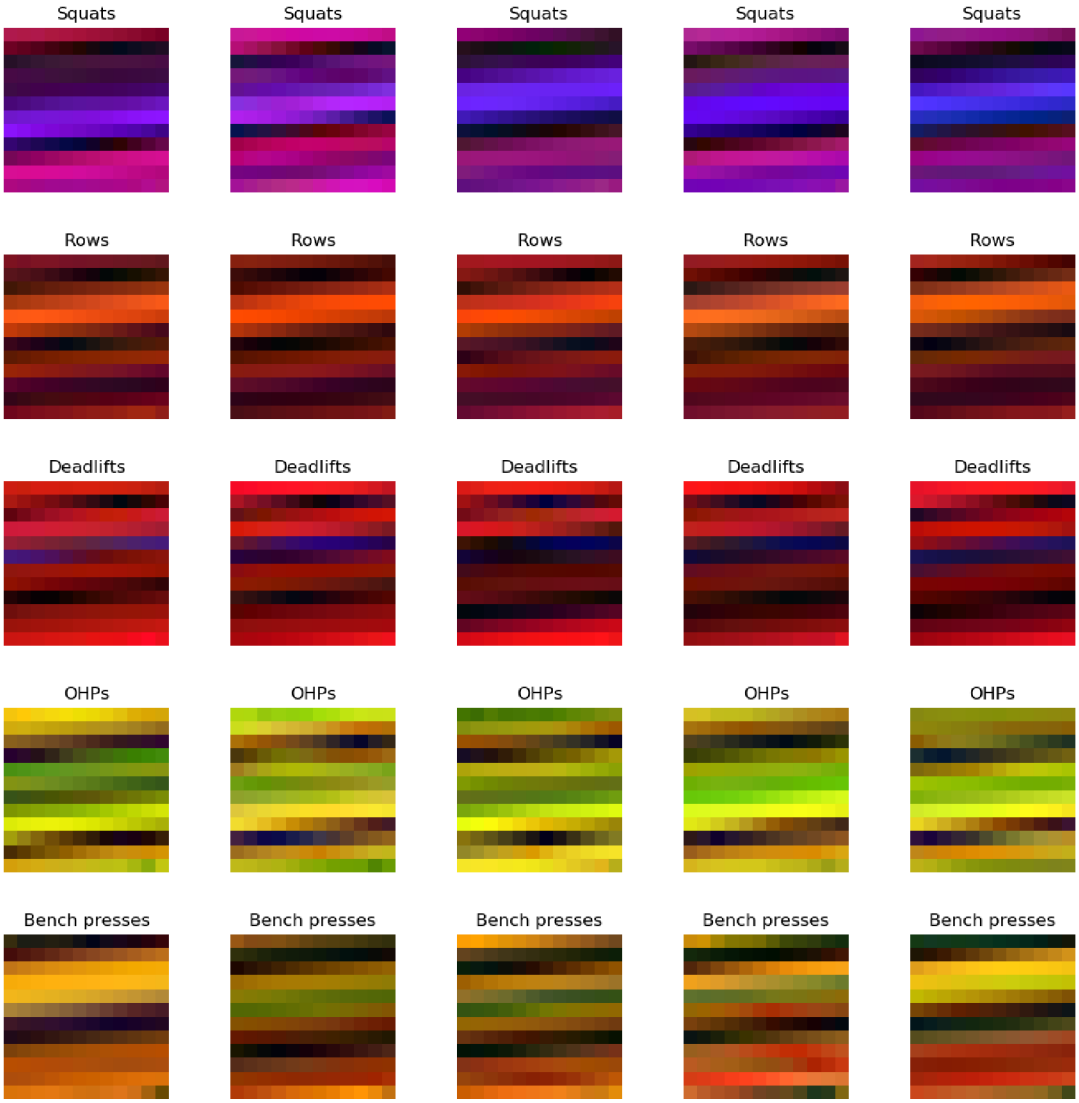
7.2.1 Vizualizačné nástroje

Z dôvodu verifikácie správnosti úpravy zaznamenaných dát a automatických procesov parsovania, aplikácia ponúka niekoľko vizualizačných nástrojov umožňujúcich prehľadné zobrazenie dát v rôznych fázach vykonávania logiky.

Medzi prvotné časti na využitie týchto nástrojov boli v prvom rade parsovanie a filtrovanie dát 6.2.1. Tu aplikácia ponúka vyobrazenie dát vo forme grafov pre každú z osí

akcelerometra separátne pričom je funkcionalita využívaná hlavne ako validačný aspekt zachytených dát. Nadväzujúc na predchádzajúcu časť 7.2 bola pridaná možnosť manuálnej úpravy dát, ktorá priniesla vizualizačným nástrojom i aspekt interaktivity.

Okrem fázy úpravy dát sa táto funkcionalita nachádza i v procese automatickej estimácie pomocou *DFT* 6.6, kde slúži pre možnosť rešpecifikácie ideálneho prahu pre tento postup.



Obr. 7.2: Vizuálna reprezentácia časti nazbieraných dát z aplikácie pre záznam bežiacej na nositeľnom zariadení, kde každá z osí akcelerometra a jeho hodnôt identifikuje jeden z kanálov RGB, pričom hodnoty boli prevedené na interval <0,255> v závislosti na maximu danej osi každého záznamu zvlášť. Táto demonštrácia primárne slúži na poukázanie vhodnosti využitia konvolučných sietí a očividné rozdiely medzi jednotlivými pohybmi. Titulok nad každým mini obrázkom značí typ aktivity a jednotlivé obrázky zaznamenávajú vždy jedno opakovanie daného pohybu.

Celkové použitie tejto logiky bolo v aplikácii postupne rozširované s pribúdajúcimi požiadavkami. Či už sa jednalo o nové moduly pre zvýšenie efektivity spracovania alebo kon-

trolné body pre overenie správnosti implementovanej logiky. Veľkým prínosom bola generickosť implementácie grafového vyobrazenia, ktorá umožňovala použitie s rôznymi typmi dát v odlišných scenároch.

7.2.2 Opätovné použitie natrénovanej siete

Po nasadení aplikácie do fáze testovania zberu dát bolo od užívateľov získaných niekoľko tisíc záznamov rôznych cvikov, kedy na jeden cvik pripadalo približne 500 až 800 jedinečných opakovaní. To umožnilo tréning neurónovej siete na markantne vyššom počte dát ako v predchádzajúcich fázach.

Testované subjekty však so záznamom dát pokračovali, čo znamenalo, v prípade pribúdajúcich dát, trénovanie siete odznova na nových dátach spolu s už použitými na trénovanie, teda sieť bola trénovaná odznova. Pri zvyšujúcom sa počte dát sa úmerne zvyšoval i čas potrebný na samotný proces tréningu. Tento trend vyžadoval vytvorenie funkcionality, ktorá umožní použitie už natrénovanej neurónovej siete a aplikovanie nových dát vo forme dotrénovania.

Implementácia tejto funkcionality umožnila zrýchliť a zefektívniť celý proces a rýchlosť nasadenia zmien do testovacieho prostredia, čím mohla byť zvýšené a zobjektívnené overovanie účinnosti v reálnych podmienkach.

7.2.3 Nepresnosti v dátach

Prvotné iterácie aplikácie a testovania používali ako vstup dáta priamo z akcelerometra. Pri spracovávaní a následnom testovaní natrénovanej neurónovej siete pomocou testovacej aplikácie pre smartphone 6.3 bolo zistené nadbytočné množstvo šumu v dátach spôsobené gravitačným zrýchlením, ktoré je v tomto tyoe dát zahrnuté. Z tohto dôvodu bol nahradený lineárnou akceleráciou, ktorá zložku gravitačného zrýchlenia odstraňuje a celkové výstupné dáta obsahujú menšie množstvo šumu.

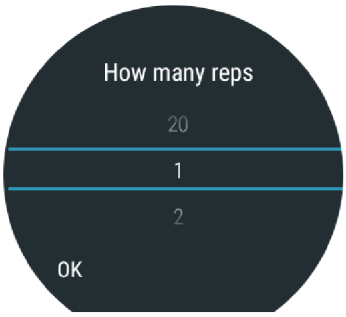
Počas zbierania dát užívateľmi v testovacej fáze prichádzalo v niekoľkých prípadoch k záznamu nechcených dát. Reakciou na spätnú odozvu tohto problému bolo pridanie funkcionality umožňujúcej vymazanie doposiaľ zaznamenaných dát viď. obrázok snímiek obrazovky 6.2, 3 riadok napravo, „reset all“. Táto implementácia umožnila znížiť celkovú náročnosť na manuálne predspracovanie dát v predtréningovom procese a zároveň zvýšiť kontrolu nad dátami samotným testovacím subjektom.

7.2.4 Vyradená funkcionalita

Vývoj aplikácie od začiatku prebiehal formou pridávania požadovanej funkcionality, ktorá bola identifikovaná ako potrebná/chýbajúca, na základe spätnej väzby. Týmto spôsobom bol minimalizovaný vývoj logiky, ktorá by bola neskôr vyradená. Prvotný návrh funkcionality záznamu dát však počítal s užívateľsky špecifikovaným počtom opakovaní pre daný dátový set. Táto možnosť sa v priebehu testovacích fáz ukázala ako prebytočná a z pohľadu užívateľa rušivá, čo viedlo k jej odstráneniu a nahradeniu modulom v desktop aplikácií umožňujúci určenie periódy automaticky/manuálne, viď. 6.2.1.

7.2.5 Analýza nazbieraných dát

Počas testovacej fázy subjekty zaznamenali vyše 5000 jedinečných opakovaní celkovo 7 cvikov, ktoré bolo možné následne použiť v procese trénovania neurónovej siete. Na jeden cvik



Obr. 7.3: Výber počtu opakovaní vykonaných v poslednom zaznamenávanom úseku pomocou aplikácie pre nositeľné zariadenia. Táto funkcionalita bola v produkčnej verzii vyradená na základe spätnej väzby od testovacích subjektov a bola nahradená v module pre parsovanie dát v desktop aplikácii.

pripadalo v priemere 700 záznamov, čo poskytlo dostatočnú pestrosť umožňujúcu vytvorenia objektívnejšej neurónovej siete.

Pri analýze získaných dát bolo pomocou vizualizačných nástrojov 7.2 zistených niekoľko skutočností, ktoré následne ovplyvnili samotné parsovanie, implementačné detaily aplikácie pre spracovanie dát a spôsob manipulácie s nimi.

Rozdiel v trvaní opakovania

Vizualizácia dát pred samotným procesom trénovania ukázala nesúrodosť dĺžky jednotlivých opakovaní v rámci jedného dátového setu. Tieto rozdiely mali v niektorých prípadoch dopad na samotné automatické parsovanie v rozsahu ovplyvňujúcom korektnosť rozpoznávania. Problémy spôsobovali dlhšie pauzy medzi jednotlivými opakovaniami, ktoré pri nekonzistentnej rýchlosti vykonávania spôsobili nekorektné rozdelenie datasetu, kedy niektoré časti zachytili len „prázdne“/statické dáta.

Výskyt takýchto problematických záznamov bol následne riešený pomocou implementácie manuálneho parsovania dát v 6.2.1, ktorý umožňoval korektné rozdelenie datasetov.

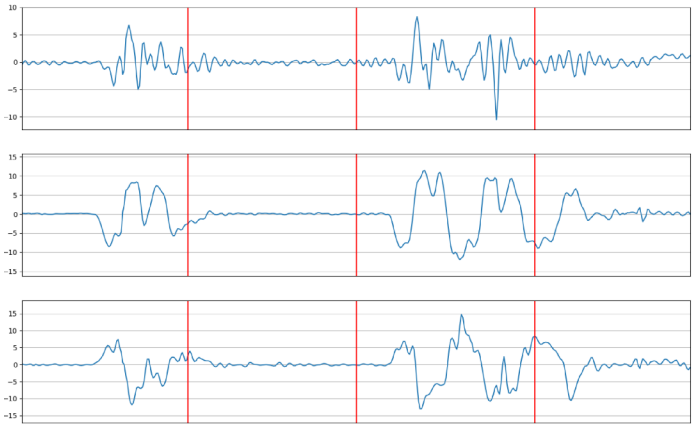
Počiatočný a koncový šum

Pri spracovávaní väčšieho počtu datasetov bola na základe vizuálnej analýzy zistená nadmerná variácia počiatočného a koncového šumu. Počiatočný šum bol zväčša spôsobený reakčným časom a reálnym začatím vykonávania cviku a koncový šum vo väčšine prípadov vychádzal z nutnosti interakcie s nositeľným zariadením a ukončením zaznamenávania.

Automatické filtrovanie založené na pevne stanovenom časovom úseku na počiatku a konci záznamu sa ukázalo ako nedostatočne efektívne. Práve z daného dôvodu bolo nahradené pomocou manuálneho filtrovania skrz interaktívny graf 6.2.1.

Export grafov

Pri spracovávaní väčšieho počtu datasetov sa analýza záznamov počas behu aplikácie ukázala ako neefektívna a časovo veľmi náročná. Prídavná funkcionalita umožňujúca export grafov jednotlivých záznamov umožnila bližšiu analýzu a verifikovanie správnosti dát vo forme súborov obrázkov reprezentujúcich tieto dáta, označených časovým razítkom reprezentujú-



Obr. 7.4: Graf zobrazujúci nekonzistentné dáta, aplikáciu automatického odhadu počtu opakovaní a následne rozdelenie, ktoré vyprodukovalo „prázdne“ statické dáta spôsobujúce problémy pri procese inferencie a detekciu falošných opakovaní. V tomto konkrétnom prípade išlo o 2. opakovanie v zázname, ktoré neobsahovalo žiadne identifikovateľné zmeny v pohybe.

cim prvý záznam v sade. Takto bolo možné dáta spracovať a vykonať ich rozbor efektívnejšie prípadne identifikovať nevhodné, poškodené alebo nekompletné záznamy.

7.3 Rozpoznávanie pohybov

Po nazbieraní dostatočného množstva dát umožňujúceho natrénovanie objektívne vyhodnocujúcej neurónovej siete bola táto sieť nasadená do aplikácie a následne podrobená testovaniu účinnosti.

Prvotné testy kvality a správnosti rozpoznávania siete boli úspešné a ich presnosť bola na úrovni 9 z 10 správne identifikovaných opakovaní cviku. Objektívnosť siete umožňovala väčšiu variáciu vo forme a spôsobe vykonávania cviku, či jeho rýchlosti.

Pri určitých testovaniach bol však zistený problém s rozpoznávaním tzv. falošných opakovaní, kedy stačilo, aby sa užívateľ dostal do počiatočnej polohy pri vykonávaní cviku a sieť túto statickú polohu identifikovala s veľkou istotou ako validné opakovanie. Tento problém viedol k vytvoreniu aplikácie pre smartphone, ktorá umožňovala simultánnu analýzu dát zo senzorov spolu s inferenčným procesom neurónovej siete 6.3.

Testovacia aplikácia následne vylúčila možnosť chybných vstupných dát, no zároveň identifikovala problematický šum spôsobený gravitačným zrýchlením, viď. 6.1.1 vedúce k zmene typu vstupných dát pre zvýšenie presnosti.

Analýza vstupných dát pre trénovaciu fázu odhalila niekoľko exemplárov 7.1, ktoré reprezentovali práve tento typ falošne pozitívnych opakovaní. Celý proces analýzy dát veľmi zjednodušila dostupnosť doplnených vizualizačných nástrojov 7.2.1 spolu s možnosťou exportovania týchto dát do grafickej súborovej reprezentácie 7.2.5.

Tento poznatok a skutočnosť zmeny typu dát na lineárnu akceleráciu znevalídnila všetky doposiaľ nazbierané dáta, ktoré nemohli byť v novej verzii aplikácie korektne použité. Naskytovala sa možnosť ich konverzie, no tento proces nedosahoval dostatočnej presnosti práve z dôvodu, že systém Android pre výpočet lineárnej akcelerácie používa i ostatné senzory pre presnejší proces filtrovania.

7.4 Výsledky

Testovanie efektívnosti aplikácie a jej modulov bolo realizované niekoľkými spôsobmi, kde cieľom v prípade funkčných testov bolo verifikovať správnosť zaznamenávaných dát, ich konzistenciu, presnosť či správnosť rozpoznávania a iné faktory, za ktoré priamo zodpovedá implementácia logiky aplikácie zabezpečujúca manipuláciu s dátami zo senzorov a ich podporné procesy na pozadí, či už v prípade záznamu dát pre tréningové účely alebo pre proces detekcie opakovaní. Mimo tohto typu testovania prebiehala validácia správnosti a konzistentnosti chovania užívateľského prostredia počas interakcií.

Celý proces bol realizovaný v niekoľkých iteráciách, počas ktorých sa menilo prostredie a podmienky testovania. V prvých fázach išlo o veľmi kontrolované prostredie so zameraním na pozitívne výsledky, vo väčšine prípadov ako práca jednej osoby, bez veľkého množstva premenných. Postupne boli menené závislosti s dôsledkom na iné typy výsledkov, pričom do testovania bolo postupne zaraďované väčšie množstvo subjektov prinášajúcich do systému väčšiu variáciu spolu s komplexnejšími a generalizovanejšími dátami.

7.4.1 Užívateľské prostredie

Pre celkový dojem z aplikácie bola správna funkčnosť a bezproblémovosť ovládania aplikácie skrz užívateľské rozhranie. Z tohto dôvodu bolo nutné odladiť jeho fungovanie i v prípade kritických stavov, ktoré mohli počas fungovania nastať, či už chybou systému alebo užívateľskej interakcie.

Rozhranie bolo testované na niekoľkých virtuálnych zariadeniach i na fyzickom modely Motorola *Moto 360 sport*[1], ktorý poskytoval dostatočnú senzorovú výbavu i výpočetný výkon na plynulý chod aplikácie. Celý vývoj aplikácie bol realizovaný na základe doporučení a pokynov pre Android a Android Wear, čo umožnilo minimalizovať celkové množstvo objavených problematických stavov.

Počas testovania konzistentnosti a správnosti fungovania užívateľského prostredia boli jediné objavené chyby viažúce sa na okrajové stavy odstránené a testovacie subjekty následne označili aplikáciu a interakciu s ňou za bezproblémovú.

7.4.2 Zber dát

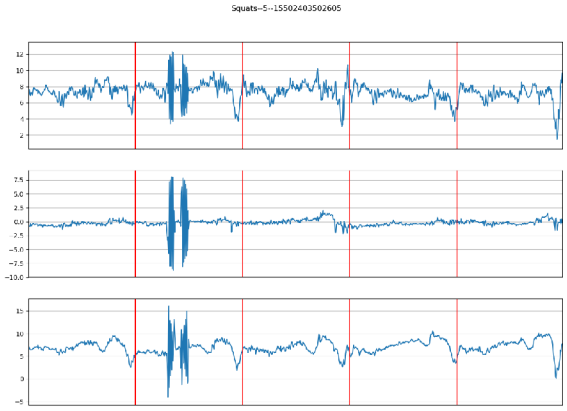
Overenie správnosti a funkčnosti celého procesu, zahrňujúceho zber dát a ich následné parsovanie, bolo uskutočnené pre jednotlivé logické celky zvlášť, čo umožnilo testovanie paralelizovať a zároveň zvýšiť kontrolu nad samotnými výsledkami spolu s určením detailnejšieho pôvodu problému, ktorý by mohol byť objavený.

Nepresné dáta, šum

Manuálna analýza dát od užívateľov a vyhodnotenie ich použiteľnosti z hľadiska procesu trénovania neurónovej siete odhalila 28 datasetov z celkových 575 nazbieraných, ktoré obsahovali zvýšené množstvo šumu alebo špecifické narušenia záznamu veľkou odchýlkou.

Dáta podobné obrázku 7.5 alebo obsahujúce veľké množstvá šumu vo väčšine prípadov spôsobené nadmernou aktivitou zápästia subjektu, na ktorom bolo nositeľné zariadenie upevnené, museli byť z procesu vyradené a nemohli byť použité pre dosiahnutie správnych výsledkov inferencie neurónovej siete.

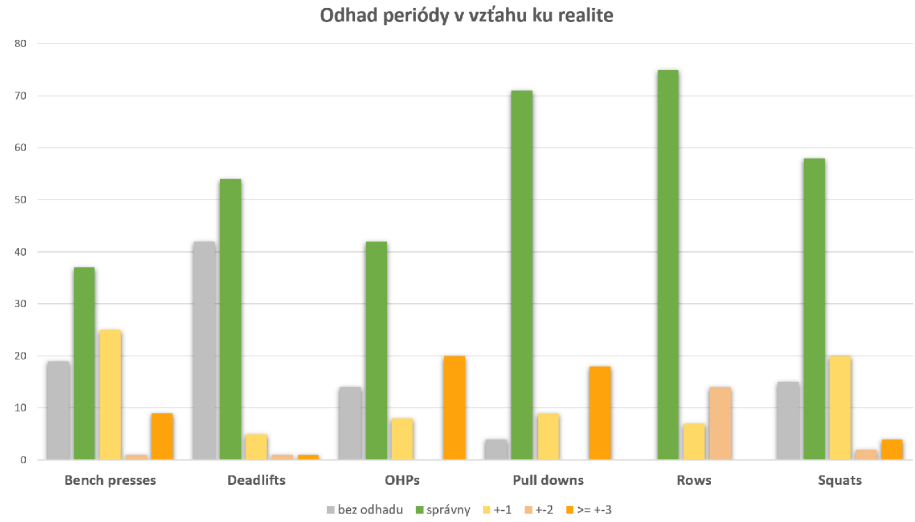
[1] https://www.motorola.com.au/products/moto-360-sport



Obr. 7.5: Ukážka datasetu s nekonzistentným záznamom, ktorý bol spôsobený vibračnou notifikáciou (2. úsek) systému Android Wear počas tvorby nových dát pre účely tréningu neurónovej siete. Tento typ šumu mal priamy dopad na nemožnosť použitia dát korektne. Keďže desktop aplikácia nepodporuje možnosť odstránenia len časti záznamu, dáta boli úplne vyradené.

Periódickosť dát

Aspekt periodickosti zaznamenaných dát bol v prvej iterácii aplikácie využívaný na schopnosť automatického rozdelovania datasetu na jednotlivé opakovania, ktoré mohli byť následne priamo využívané pre proces tréningu neurónovej siete.



Obr. 7.6: Graf zobrazujúci výsledky automatickej detekcie počtu opakovaní v desktop aplikácii. Priemerný počet analyzovaných dátových setov a ich následnej estimácie bol na úrovni 100 jedinečných záznamov pre každý zo 6 vyobrazených cvikov. Pre zjednodušenie sú zaznamenané výsledky tejto operácie pre prípady celkovej neúspešnosti odhadu, odhadu zhodnému realite, s odchýlkou v rozpätí jedného alebo dvoch opakovaní a výsledky, ktoré boli odlišné o tri a viac opakovaní voči realite.

Ako možno vidieť z grafu 7.6, presnosť aplikácie automatického parsovania dát nebola na dostačujúcej úrovni a preto bola z výslednej aplikácie táto funkcionalita čiastočne odstránená a je ju možné použiť len na vyžiadanie. Okrem nepresnosti odhadov nahradenie tejto funkcionality vychádzalo i z problému s tzv. falošnými opakovaniami, viď. obrázok 7.4, a nekonzistentným intervalom vykonávania opakovaní viď. obrázok 7.1.

Znížený výkon zariadenia

Znížený výpočetný výkon nositeľných zariadení v kombinácii s komplexnými aplikáciami a nárokmi operačného systému Android Wear predstavoval v určitých scenároch výzvu z hľadiska optimalizovanosti aplikácie a jej behu.

Jedným z problémov, ktorému bolo nutné venovať pozornosť počas vývoja, bolo automatické ukončovanie aplikácií na pozadí. To sa viazalo i na aplikácie, ktoré boli pozastavené z dôvodu vyskakujúcich notifikácií. Počas testovania nastalo niekoľko prípadov, kedy zobrazená notifikácia obsahujúca komplexnejšie informácie alebo následná interakcia s ňou ukončila práve bežiaci aplikáciu a tým i záznam dát alebo prebiehajúce štatistiky o detekcii.

Riešením bolo v prípade zníženia nárokov počas detekcie bolo minimalizovanie počtu požiadavkov na inferenciu neurónovej siete. Proces kedy boli analyzované každé príchodzie dáta bol nahradený prahom, ktorý určoval najmenší počet novo získaných dát na ich inferenciu neurónovej siete. Keďže v prípade implementácie aplikácie hodnota tohto prahu dosahovala v priemere 50 záznamov, výsledná náročnosť bola veľmi znížená. To znamenalo zároveň veľké zníženie pravdepodobnosti, že systém ukončí aplikáciu ak by bola na krátky čas presunutá do pozadia a teda uchovanie prebiehajúceho procesu.

V prípade zberu dát bol problém výpočetného výkonu vyriešený pomocou vypisovania do súboru počas celého behu, pokým je aplikácia na žive tzn. i v prípade, že je na pozadí. Okrem toho bola všetka práca so súborom vždy bezpečne započatá i ukončená pomocou dostupných nástrojov.

Zabezpečenie úspešného odoslania zaznamenaných dát na server bolo zrealizované pomocou *wake lock*[2] umožňujúci držať zariadenie mimo režimu spánku a tým limitovať možnosť predčasného ukončenia spojenia so vzdialeným serverom.

Optimalizácie výkonu a zníženie požiadaviek aplikácie na systém priniesli pozitívnu spätnú väzbu. Úprava zároveň ovplyvňovala celkový beh aplikácie i interakcie s ňou eliminujúc nečakané stavy, ktoré by mohli nastať pri nadmerne vyťaženom zariadení.

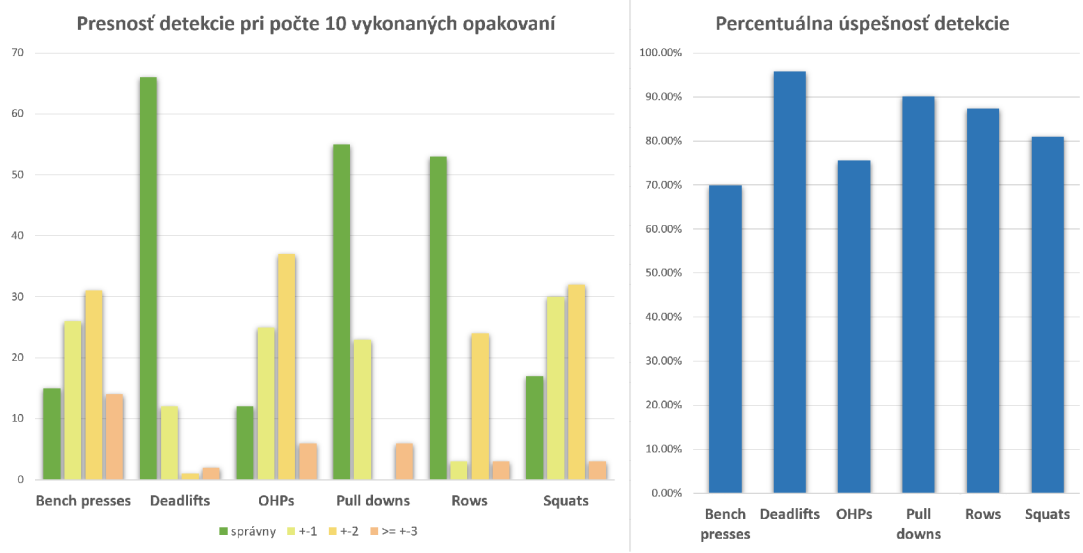
Výpadky siete počas nahrávania

Akýkoľvek typ prerušenia počas nahrávania dát na vzdialený server je spracovávaný v prvom rade pomocou knižničných funkcií *Volley*(viď. podsekcia 6.1.2) odkiaľ je následne delegovaná informácia užívateľovi pomocou jednoduchého dialógu. Ak však užívateľ pristupuje už do modulu synchronizácie bez internetového pripojenia, je nemožnosť nahrávania dát delegovaná pomocou vizualizačných techník, ako napríklad zošednutie a zakázanie tlačidla v užívateľskom prostredí.

Práve z dôvodu použitia už existujúcich knižníc s robustným spracovávaním akýchkoľvek chýb, ktoré sa môžu vyskytnúť počas procesu nahrávania, boli výsledky práce s týmto modulom veľmi pozitívne a neboli odhalené žiadne problémy.

[2] https://developer.android.com/training/scheduling/wakelock

7.4.3 Detekcia pohybov



Obr. 7.7: Súbor grafov vizuálne reprezentujúci schopnosť aplikácie korektne detekovať a správne určiť počet vykonaných opakovaní. Proces testovania prebiehal vždy vykonaním 10 opakovaní daného cviku (os x grafu) a následným porovnaním z celkovým počtom rozpoznaných opakovaní v aplikácii. Graf naľavo vyobrazuje počet detekovaných opakovaní, teda presnosť neurónovej siete, prípadne veľkosť odchýlky od reality. Graf napravo zastupuje percentuálnu presnosť inferencie neurónovej siete vychádzajúcej z grafu naľavo.

Ohodnotenie úspešnosti aplikácie vychádzajúce z grafu 7.7 bolo realizované pomocou logiky závislej na odchýlke, zvolené na základe spätnej väzby od užívateľov a ich vnímanej efektívnosti aplikácie v tvare:

- presný počet = 100%
- +-1 = 90%
- +-2 = 70%
- >= +-3 = 0% úspešnosť

Okrem správnej funkčnosti bola aplikácia testovaná i v niekoľkých špecifických prípadoch, ktoré sa vymykali štandardu a boli zamerané na odhalenie správania v týchto situáciach.

Nepresné dáta, šum

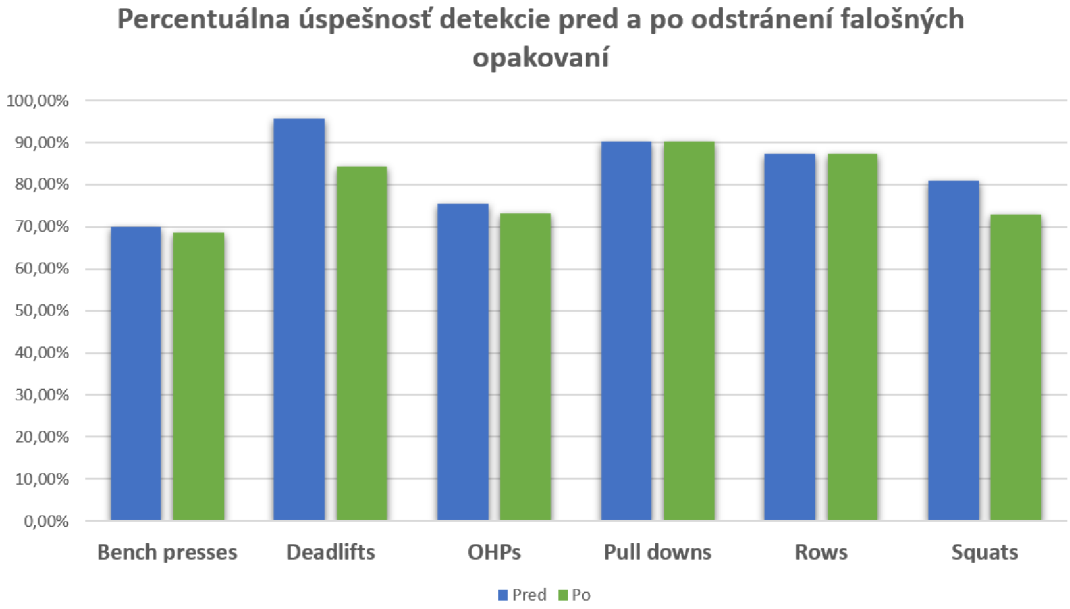
Po zanesení určitého typu šumu alebo nepresnosti dát, ktoré bolo vo väčšine prípadov spôsobené zvýšenou aktivitou pohybu zápästia, na ktorom bolo zariadenie upevnené, sa celková efektivita a správnosť rozpoznávania znížila v priemere o 30 - 40 %. Tento jav bol z veľkej časti spôsobený chýbajúcim prvkom periodicity dát a taktiež nedostatku vzoriek tohto typu počas procesu trénovania siete.

Neštandardné pozície

V prípade neštandardných pozícií pri vykonávaní opakovaní dosiahnutých pomocou zmeny orientácie zápästia v priestore, bola celková úspešnosť v prípade cvikov, ktoré počas tréningového procesu nemali vzorky rôznych typov naklonení zariadenia, znížená pod akceptovateľnú úroveň (<50%). V kontexte drepu, ktorý bol natrénovaný na niekoľko rôznych polôh (predné drepy, *lowbar*, *highbar* a iné)[3], nebola presnosť ovplyvnená v rozmedzí vyššom ako 5 %. Táto informácia jasne značí na nepresnosť spôsobenú chýbajúcimi dátami.

Variabilná rýchlosť rozpoznávania

Keďže samotné dáta využívané na proces trénovania neurónovej siete boli v aspekte rýchlosti dostatočne rozmanité, zmena rýchlosti vykonávania opakovaní cvikov mala takmer nulový dopad na presnosť detekcie. Jediný zaznamenateľný dopad malo zvýšenie rýchlosti v prípade tlaku v ležme (*bench press*), ktorý v procese trénovania nemal dostupné rýchle opakovania z dôvodu bezpečnosti cvičenca počas ich záznamu.



Obr. 7.8: Graf porovnávajúci účinnosť neurónovej siete po odstránení falošných opakovaní z datasetu určeného na trénovanie rozpoznávania, bližšie rozoberaný v analýze dát 7.2.5. V prípade *Deadlifts* a *Squats* nastala situácia, pred odstránením týchto nekorektných dát, kedy reálne opakovanie, ktoré malo byť rozpoznané, bolo rozpoznané na základe statických dát polohy ruky a nie zmenami hodnôt akcelerometra. To prinieslo do výsledkov korektnosti rozpoznávania nepresnosti a zvýšené pozitívne hodnotenie.

Neklasifikovaný subjekt

Presnosť detekcie počtu opakovaní v kontexte neklasifikovaného subjektu bola v priemere znížená len v rozmedzí 3-10% z dôvodu konzistentnosti spôsobu vykonávania jednotlivých cvičení, ktoré majú základné predpoklady na formu a spôsob vykonávania. Najväčšie ne-

[3] https://barbend.com/ultimate-squat-guide

presnosti vznikali len v prípade ak mal užívateľ veľmi špecifický spôsob vykonávania pohybu, ktorý sa odlišoval od bežnej formy. Tento problém sa ale v takom prípade viazal na predchádzajúcu analýzu neštandardných pozícii pri vykonávaní.

Nevalídne dáta

Nadväzujúc na informácie z analýzy dát 7.2.5, kde boli odhalené problémy s nekonzistentnými pauzami v opakovaní počas záznamu dát určených pre proces tréningu neurónovej siete, spôsobovali tieto falošné opakovania chyby v detekcii správnych cvikov a pri rozpoznávaní následne sieť identifikovala statické pohyby v správnej polohe na základe týchto falošných dát ako valídne opakovanie. Filtrovaním tohto typu bolo možné eliminovať výskyty nesprávnej detekcie.

Tento krok teda čiastočne znížil celkovú úspešnosť detekcie neurónovej siete v priemere o 4%. Pokles je len orientačný z dôvodu nemožnosti duplikovania identických podmienok testovania ako v prvej fáze testovania.

7.5 Návrhy na zlepšenia

Na základe spätnej väzby v testovacej fáze bolo možné väčšinu kritiky pretransformovať na vylepšenú funkcionalitu, ktorá tieto požiadavky alebo doporučenia implementovala, upravovala alebo odstraňovala, viď. 7. Takýchto zlepšení bolo niekoľko:

- vibračná odozva pri štarte procesu záznamu
- odstránenie nutnosti zadávania počtu opakovaní
- možnosť nahratia dát na server priamo zo zariadenia
- vizualizácia podpornej logiky
- odstránenie aspektu gravitačného zrýchlenia
- manuálne filtrovanie
- manuálne rozdelenie datasetov
- testovacia aplikácia pre výkonnejšie zariadenia

Okrem kritiky k existujúcej funkcionalite subjekty taktiež poskytovali návrhy na možné dodatočné prvky, ktoré by z ich pohľadu spravili aplikáciu zaujímavejšiu a poskytli jej výhodu nad konkurenciou.

- zostavenie sady cvikov a sledovanie procesu vykonávania
- spätná väzba pri cvičení
- možnosť trénovania neurónovej siete priamo na zariadení
- nezávislosť na tréningovom prvku - sieť sa učí pri používaní
- vizualizácia na nositeľnom zariadení
- rozdeľovanie opakovaní na série počas celého sledovacieho cyklu

- motivačná funkcionalita
- súťaženie s ostatnými užívateľmi

Tieto doporučenia patrili medzi tie, ktoré sa v ďalších verziách aplikácie budú s veľkou pravdepodobnosťou postupne pridávať, keďže celý vývoj aplikácie bude pokračovať i za medze tejto práce. Najväčším lákadlom pre užívateľov je určite funkcionalita umožňujúca sledovanie prebiehajúceho tréningu a štatistiky spolu s možnosťou porovnávania sa s ostatnými užívateľmi. V prípade *on-board* učenia neurónovej siete je táto funkcionalita obmedzená vývojom knižníc *TensorFlow-Lite*, kde je v dobe tvorby práce vývojármi ohlásená nová verzia, ktorá by mohla prísť na trh v najbližšom roku a povoliť tréning siete i priamo na nositeľnom zariadení s obmedzeným výpočetným výkonom.

Kapitola 8

Záver

Cieľom tejto práce bolo identifikovať, analyzovať, zhodnotiť a následne využiť čo najvhodnejším spôsobom strojové učenie na nositeľných zariadeniach s cieľom automatického rozpoznávania pohybov, ktoré predtým užívateľ definoval.

Samotná analýza dostupných neurónových sietí pre takýto účel funkcionality bola schopná identifikovať a zhodnotiť dostupné implementácie a knižnice, kde sa knižnica *TensorFlow-Lite* ukázala ako najvhodnejší kandidát na použitie.

Návrh aplikácie bral do úvahy predošlé zistenia a na základe nich navrhol štruktúru konvolučnej neurónovej siete, podobu a funkcionalitu aplikácie ponúkajúcej možnosť záznamu dát pre tréning i následnú intepretáciu z *inferenčného* procesu už natrénovanej siete.

Implementačná časť práce priniesla plne funkčnú, vyladenú aplikáciu, ktorá pomocou jednoduchého užívateľského prostredia vhodného pre použitie v zariadení umiestnenom na zápästí, s ktorým užívateľ interagoval, umožňuje funkcionalitu či už záznamu dát pre zlepšovanie procesu detekcie, jednoduché odoslanie týchto zaznamenaných dát i samotnú detekciu vopred natrénovaných cvikov. Podľa spätnej väzby užívateľov je aplikácia taktiež plne konkurenčne–schopná.

Tieto úlohy sa podarilo veľmi vhodne spracovať čo malo za následok vytvorenie detekcie schopnej aplikácie pre Android Wear využívajúcej natrénovanú neurónovú sieť za pomoci zozbieraných dát od testovacích subjektov.

Testovanie účinnosti a presnosti procesu tréningu i detekcie prinieslo niekoľko zaujímavých výsledkov. Prvotne zvolená funkcionalita automatického parsovania bola z dôvodu nekonzistentnosti datasetov nahradená predvoleným manuálnym parsovaním. Samotný proces detekcie sa stretol s niekoľkými problémami spôsobenými vo väčšine prípadov práve nepresnými dátami počas tréningového procesu. Celková úspešnosť aplikácie a schopnosti detekcie teda vychádzala z presnosti predprípravného procesu tréningu neurónovej siete a jej celkovej štruktúry.

Na základe výsledkov testovaní je možné vyvodiť niekoľko možných aspektov, ktoré boli zodpovedné za nižšiu schopnosť detekcie v prípade niektorých cvikov. Medzi prvotný, ktorý bol v neskorších iteráciách zmenený, bol typ dát lineárnej akcelerácie. Tento typ prinášal vhodnejšie výsledky práve z dôvodu nižšieho celkového šumu. Okrem typu dát sa tiež naskytuje možnosť upravenia štruktúry neurónovej siete, ktorá vychádzala zo štruktúr určených pre detekciu gest, no v prípade komplexnejších pohybov by mohli byť dodatočné vrstvy prospešné.

V prípade možnosti pridania funkcionality do samotnej aplikácie sa ponúka komplexnejšia možnosť sledovania údajov počas procesu detekcie. Tá by mohla priniesť možnosti predom zvolenej štruktúry celého tréningu, kedy by aplikácia postupne informovala uží-

vateľa o type cviku a zaznamenávala počet opakovaní spolu s celkovým progresom. Tieto doporučenia vychádzali i zo spätnej väzby od užívateľov, ktorý by takúto schopnosť aplikácie brali ako veľkú konkurenčnú výhodu. Z toho by taktiež vyplývala zvýšená previazanosť funkcionality so spárovaným smartphone, na ktorom by užívateľ mohol tieto štatistiky v prehľadnom formáte sledovať v čase.

Aplikácia v súčasnom stave ponúka základný dôkaz a demonštráciu možností použitia neurónových sietí na výpočetne obmedzenejších nositeľných zariadeniach. Pokračujúci vývoj by mohol priniesť ekosystém aplikácií ponúkajúcich komplexnú funkcionalitu určenú pre sledovanie užívateľských fitnes cieľov i asistenciu, ktorej stredobodom by boli práve nositeľné zariadenia, ktorých potenciál neustále rastie.

Literatúra

[1] Abadi, M.; Agarwal, A.; Barham, P.; aj.: *TensorFlow*. tensorflow.org, 2015, [Online; navštívené 11.1.2019].

[2] Anand, D. A.: *Foundations - 2: Periodicity Detection, Time-series Correlation, Burst Detection*. Leibniz Universität Hannover, L3S Research Center, 2014.

[3] Bonad, E.: *Cross-Validation Strategies*. Jan 2017, [Online; navštívené 6.11.2018].

[4] Cornelisse, D.: *An intuitive guide to Convolutional Neural Networks*. *medium.freecodecamp.org*, Duben 2018, [Online; navštívené 10.12.2018].

[5] Ebesu, T. A.: *Training a TensorFlow graph in C++ API*. github.io, Jully 2016, [Online; navštívené 25.10.2018].

[6] Gibson, A.; Nicholson, C.; Patterson, J.: *Deeplearning4j*. en.wikipedia.org, Prosinec 2018, [Online; navštívené 29.12.2018].

[7] Google LLC: *Motion sensors*. 2018, [Online; navštívené 11.1.2019].

[8] Google LLC: *Wear OS by Google*. withgoogle.com, 2018, [Online; navštívené 10.1.2019].

[9] Hao, Z.: *Loss Functions in Neural Networks*. github.io, Červen 2017, [Online; navštívené 15.10.2018].

[10] Hlavačka, M.: *Studie možností využití nositelných zařízení*. Vysoké učení technické v Brně, 2016, [Online; navštívené 02.09.2018].

[11] Jahoda, P.: *All symbols in TensorFlow | TensorFlow*. github, 2019, [Online; navštívené 6.12.2018].

[12] Karn, U.: *An Intuitive Explanation of Convolutional Neural Networks*. *ujjwalkarn.me*, Srpen 2016, [Online; navštívené 12.10.2018].

[13] Martinez, J. M. V.: *Hands-on TensorFlow Lite for Intelligent Mobile Apps*. Packt Publishing, Mar 2018, ISBN 9781788990677.

[14] dos Santos, L. A.: *Deep Learning · Artificial Inteligence*. *gitbook.io*, 2018, [Online; navštívené 02.12.2018].

[15] Siddique Hameed, J. C.: *Mastering Android Wear Application Development*. Packt Publishing Ltd., 2016, ISBN 978-1-78588-172-5.

[16] Skymind: *Guide | Deeplearning4j*. deeplearning4j.org, 2018, [Online; navštívené 12.10.2018].

[17] Stanford University School of Engineering: *Introduction to Convolutional Neural Networks for Visual Recognition*. youtube.com, Srpen 2017, [Online; navštívené 12.11.2018].

[18] Stanford University School of Engineering: *CS231n: Convolutional Neural Networks for Visual Recognition*. github.io, 2018, [Online; navštívené 12.10.2018].

[19] Tang, J.: *Intelligent Mobile Projects with TensorFlow*. Packt Publishing, 2018, ISBN 9781788834544.

[20] TensorFlow: *TensorFlow Lite*. tensorflow.org, Leden 2018, [Online; navštívené 15.8.2018].

[21] Tensorflow: *All symbols in TensorFlow | TensorFlow*. 2019, [Online; navštívené 6.8.2018].

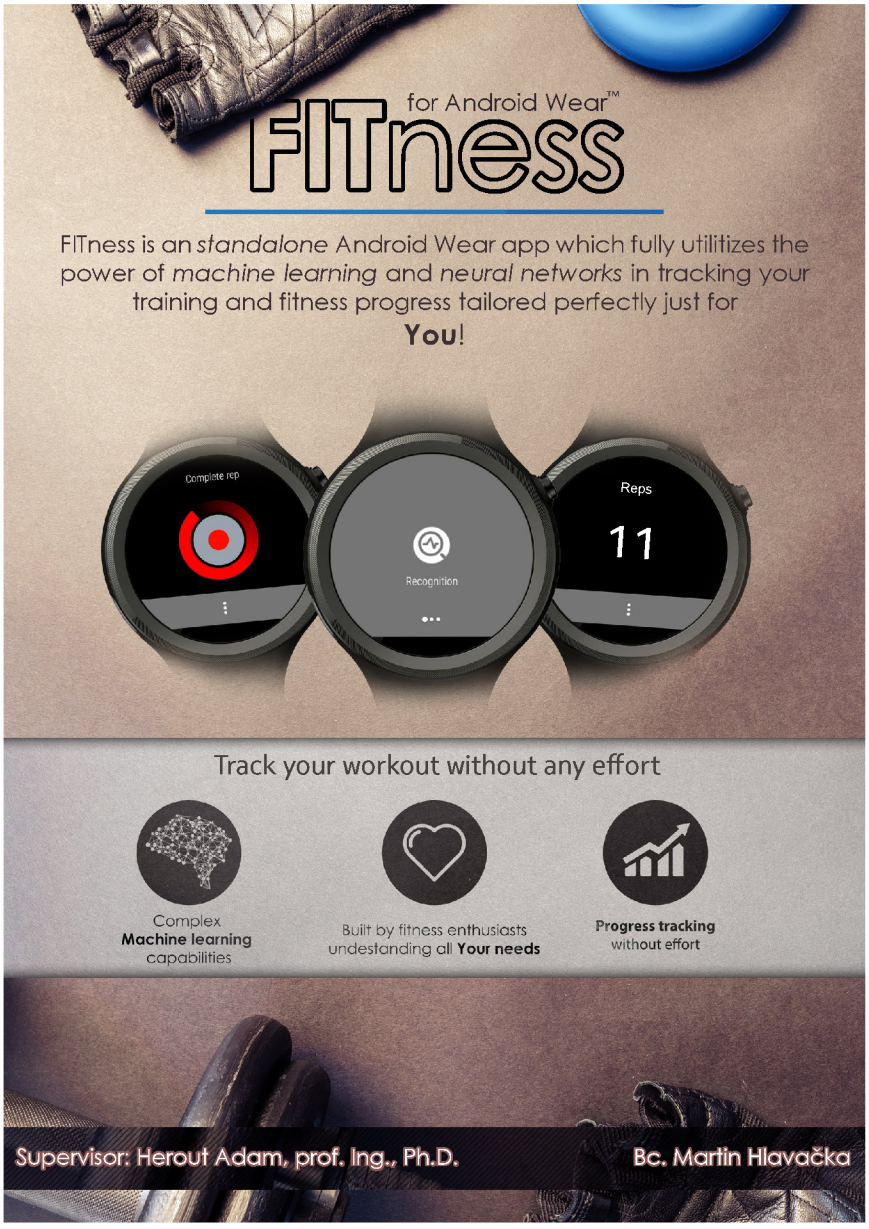
[22] V., K.; A., K.; S., K.: *Deep learning with theano, torch, caffe, tensorflow, and deeplearning4J: which one is the best in speed and accuracy? elib.bsu.by*, 2016, [Online; navštívené 03.01.2019].

[23] Wang, J.; Chen, Y.; Hao, S.; aj.: *Deep learning for sensor-based activity recognition: A Survey*. Elsevier, 2018.

[24] Štěpánková, O.; Sieger, T.; Vysloužilová, L.; aj.: *Testování modelů a jejich výsledků - Jak moc můžeme věřit tomu, co jsme se naučili?* FAKULTA ELEKTROTECHNICKÁ, České vysoké učení technické v Praze, 2018.

Príloha A

Plagát



Príloha B

Obsah CD

- `apk/` - inštalačné balíčky Android aplikácií
- `dataset/` - nazbierané ukážkové dáta
- `doc/` - programová dokumentácia Android aplikácie
- `model/` - základný natrénovaný model neurónovej siete a metadata súbor pre aplikáciu
- `src/desktop/` - zdrojové súbory aplikácie pre desktop
- `src/server/` - zdrojové súbory vzdialeného serveru pre prímanie dát z nositeľného zariadenia
- `src/smartphone/` - zdrojové súbory Android aplikácie pre smartphone vrátane súborov build.gradle potrebných pre správny preklad
- `src/wear/` - zdrojové súbory Android aplikácie pre nositeľné zariadenie vrátane súborov build.gradle potrebných pre správny preklad
- `tex/` - zdrojové súbory k písomnej správe v LaTeX-u
- `poster.pdf` - prezentačný plagát práce
- `projekt.pdf` - písomná správa
- `video-thesis.mp4` - prezentačné video Android Wear aplikácie

Príloha C

Manuál k aplikáciám

C.0.1 Minimálne požiadavky na systém

V prípade aplikácie pre smartphone, teda mobilný telefón alebo tablet je požadovaná minimálna verzia operačného systému Android `7.1`. Pre samotné nositeľné zariadenie je taktiež požadovaná verzia Android Wear `7.1`.

Aplikácia pre desktop využívaja jazyka Python 3. Doporučenou verziou pre korektný beh a bezproblémové fungovanie je `3.7.3`.

C.0.2 Potrebné povolenia Android aplikácií

Počas inštalácie je nutné povoliť aplikácii prístup k srdečnému senzoru, prístupu na internet, vibračnej odozve a možnosť udržovať zariadenie v chode bez prechodu do spánku.

C.0.3 Samotná inštálácia

Pre nainštalovanie Android aplikácií existujú 2 metódy.

Prvou možnosťou je inštalácia už vygenerovaných inštalačných balíčkov. V tomto prípade treba nahrať balíček do vnútornej pamäte zariadenia a následne nainštalovať cez správcu súborov. Pred inštaláciou je nutné v nastaveniach zariadenia povoliť inštalovanie aplikácií z neznámych zdrojov. Samotný proces inštalácie sa však nemení. V prípade aplikácie pre nositeľné zariadenie je vhodné použiť nástroj *adb*[1], ktorý umožňuje inštaláciu na zariadenia so systémom Android Wear keďže systém nemá integrovaného správcu súborov.

Druhou alternatívou je preklad poskytnutých zdrojových súborov pomocou niektorého z vývojových prostredí pre Android. Zložka `src/wear` a `src/smartphone` obsahuje okrem samotnej štruktúry aplikačných tried i súbory *build.gradle*, ktoré poskytujú informácie o tom, aké knižnice a spôsob prekladu je treba použiť. Po vlastnom preklade aplikácie je možné ju nainštalovať pomocou *adb*, alebo vygenerovaním inštalačného balíčka. Následne je už postup rovnaký ako v prvom prípade.

Aplikácie pre smartphone a nositeľné zariadenie na sebe nie sú závislé a nie je potrebné inštalovať obidve.

K behu aplikácie určenej pre desktop nie je nutné zdrojové súbory prekladať, sú interpretované.

[1] https://developer.android.com/studio/command-line/adb

C.0.4 Použitie neurónovej siete

Pre správnu funkcionalitu Android aplikácie je nutné poskytnúť súbor obsahujúci neurónovú sieť a metadáta. Tohto je možné docieliť pomocou skopírovania súborov do externého úložiska zariadenia a aplikačnej zložky špecifickej pre aplikáciu. Táto zložka je vo väčšine prípadov v lokácii:

- `/storage/emulated/0/Android/data/com.hlavackamartin.fitnessapp.smartphone/files/` pre smartphone
- `/storage/emulated/0/Android/data/com.hlavackamartin.fitnessapp.recognition/files/` pre nositeľné zariadenie

Požadovaný názov *tflite* súboru natrénovanej neurónovej siete *detect.tflite* a názov metadát *labels.txt*.