Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Arvutiteaduse instituut

Karl-Jonathan Lellep

iOS rakendus piljardi treeningute jälgimiseks ja toetamiseks

Bakalaureusetöö (9 EAP)

Informaatika

Juhendaja:

Artjom Lind

Tartu 2021

iOS rakendus piljardi treeningute jälgimiseks ja toetamiseks

Lühikokkuvõte:

Töös kirjeldatakse piljardi tausta ja käsitletakse probleemi, mis tõuseb iseseisval treenimisel. Probleemiks on tehnikavigade märkamise keerukus ja arengu piiratus teatud tasemele jõudes. Selle probleemi lahenduseks on olemas erinevaid meetodeid, nende hulgas ka abivahendid, mis kasutavad nutiseadmeid. Selle tööga pakutakse lahenduseks nutikella kasutusele võtmist, löögi soorituse jälgimiseks ja tagasiside koostamiseks. Töö eesmärgiks on luua rakenduse prototüüp (iOS ja WatchOS platvormile) ja masinõppe mudel, mis koostöös tunneksid inertsiaalsete liikumisandurite andmevoost ära piljardi löögi liigutust esindavad andmed. Töö käigus valminud mudelit ja rakenduse prototüüpi testiti sõltumatu mängija peal ja vahendid said positiivse tagasiside. Valminud prototüüpi kasutades saab usaldusväärselt koguda löögi andmeid ja kogutud andmeid kasutada projekti edasiarenduseks lööki hindava mudeli treenimisel.

Võtmesõnad: liigutuste klassifitseerimine, masinõpe, nutiseadmete arendus, iOS

CERCS: P170 Arvutiteadus, P175 Informaatika

iOS Application for Monitoring and Supporting Training in Billiards

Abstract:

This paper introduces billiards background and works on a problem that is present in individual training. The problem is difficulty noticing technical errors and limited progress upon reaching a certain skill level. There are various methods available to deal with this issue; one of them is using smart devices. This thesis proposes using a smartwatch to track shot motion and compile feedback based on the collected data. This thesis aims to create a prototype application (on iOS and WatchOS) and a machine learning model that together would recognise data representing shot motion from a continuous stream of motion sensor data. An independent player tested the tools created and gave positive feedback. Collecting motion data representing billiard shots was reliably possible using the tools developed in the scope of the thesis. The data collected will be used in the future development to create a model that will give feedback based on shot performance.

Keywords: gesture recognition, machine learning, mobile development, iOS

CERCS: P170 Computer science, P175 Informatics

Sisukord

1. Sissejuhatus

Nutiseadmed mängivad meie igapäevaelus üha suuremat rolli. Selle teadustööga soovitakse nutiseadmete pakutavaid võimalusi rakendada piljardispordis. Nii piljardi kui ka teiste kiisportide juures on löögitehnika kogu soorituse peamine alustala. Probleem, millele töös keskendutakse, tõuseb sellest, et jõudes teatud tasemele on mängijal endal raske tehnikat parandada. Kui algajal piisab arenemiseks lihtsalt mänguajast, siis kogenumad mängijad jõuavad teatud ajaga tasemele, kus lihtsalt mänguajast arenguks ei piisa. Sellise arengu platoo peamiseid põhjuseid on kaks. Esiteks, teadmiste hulk, mida on võimalik omandada lihtsalt mängides, on piiratud. Teine põhjus on löögitehnika stabiliseerumine.

Tehnika stabiliseerumine tähendab seda, et mängija on harjunud oma tehnikavigadega piisavalt hästi hakkama saama. Selleks, et kiispordis oleks löök tehniliselt hea, on eelkõige vajalik kii sirgjooneline ja ühtlane liikumine. See tähendab nii ühtlast kiirendust enne lööki kui ka ühtlast aeglustust pärast löögi sooritust. Kuna piljardi löögi liigutus ja löögiasend on inimese kehale anatoomiliselt ebaloomulikud, siis esineb peaaegu kõigil iseseisvalt alustanud mängijatel tehnikas vigu. Tehnikavigade enamlevinud näiteks võib tuua tahtmatud küljelt küljele liigutused löögi ajal ja kuuli „torkamine“. Sageli on mängijad harjunud selliste vigadega alateadlikult hakkama saama. Probleemi tekitavad alateadlikult korrigeeritavad vead pingeolukorras mängides, sellisel juhul on vigade alateadlik korrigeerimine häiritud ja vigade mõju võimendatud. Harjunud vigade parandamine, eriti löögitehnika juures on keeruline esiteks seetõttu, et selliseid vigu iseseisvalt märgata on väga keeruline. Kui vead on siiski suudetud tuvastada võib väga kergelt tekkida olukord, kus tähelepanu hajub ja mängija pöördub tagasi vanade vigade juurde.

Selliste raskuste ületamiseks tehnika treenimisel on erinevaid võimalusi ja vahendeid. Visuaalsel vaatlusel põhinevad lähenemised on treenimine koos treeningpartneriga või oma treeningute filmimine ja seejärel salvestuste analüüsimine. Mõõteseadmetel põhinevad lähenemised on treeningvahend *DigiCue* ja mobiilirakendus *Cue Measure*.

1.1 Eesmärk

Teadustöö esialgne eesmärk oli ehitada iOS ja WatchOS platvormile rakendus, mis kasutab nutikella liikumisandureid löögi soorituse mõõtmiseks ja hindamiseks. Töö idee tuleneb mängijate praktilisest vajadusest piljardis ennast iseseisvalt efektiivsemalt treenida. Kuigi on olemas vastavaid treeningvahendeid ja ka alternatiivseid lahendusi on neil kõigil omad

puudujäägid. Eelnevalt mainitud lahendustest kõige suurema potentsiaaliga iseseisvaks treenimiseks on mobiilirakendus *Cue Measure*. Et saada mööda selle vahendi suurimast probleemist, mida kirjeldatakse peatükis 2.2, otsustati toimida analoogselt ehk mõõta liigutust nutiseadmega, aga mõõtmiseks kasutada nutikella, mitte telefoni. Selline lähenemine koondab löögitehnika treenimiseks alternatiivsete võtete parimad küljed.

Nutikell, mis liigutust mõõdab, on asetatud käele millega kiid liigutatakse. Mängija saab käekella pidevalt kanda, seega harjub ta lisaraskusega ja see kõrvaldab kiile raskuse lisamise probleemi. Kuna nutikella ja selle andurite jaoks on kõik löögid üldjoontes analoogselt jälgitavad, on kõrvaldatud ka piiratud vaatenurkade probleem. Sõltuvalt sellest, kui edukas on masinõppe mudeli treenimine tagasiside koostamiseks, on lahendusel potentsiaali ka anda paremat tagasisidet kui lihtsalt binaarne „hea“ või „halb“ tagasiside.

Töö käigus osutus seatud eesmärk ehitada valmis rakendus eeldatust keerulisemaks ja teostatava töö eesmärk kohandati kättesaadavamaks. Probleemi, mille tõttu eesmärki muudeti, kirjeldatakse peatükis 3.4.4. Kohandatud eesmärgiks sai masinõppe mudeli ja prototüüp rakenduse loomine, mis võimaldaks automatiseerida löögi andmete kogumist jooksvast andmevoost. Sellist mudelit kasutades saaks reaalajas jooksvate andmete pealt koguda ja salvestada vaid löögi andmed. Sel viisil saaks koguda suuremal hulgal andmeid ja see lihtsustaks tulevikus projekti edasiarendamist, kõrvaldades sessioonide ükshaaval käsitsi sildistamise probleemi.

Töö on jaotatud viieks peatükiks. Teises peatükis tutvustatakse teema tausta ja alternatiivseid vahendeid. Kolmandas peatükis kirjeldatakse tehtud tööd. Neljandas peatükis kirjeldatakse ja analüüsitase töö tulemusi.

2. Valdkonna ülevaade

Peatükis tuleb juttu nii piljardi kui ka inertsiaalsete liikumisandurite kasutamise taustast. Tutvustatakse alternatiivseid vahendeid probleemi lahendamiseks ja tutvustatakse masinõppe mudeli treenimiseks valitud lähenemist.

2.1. Kiispordi eripärad

Kiisportide juures taanduvad sooritust mõjutavad tegurid kahte peamisesse valdkonda. Nendeks on löökide tehniline sooritus ja teoreetilised teadmised mängu kohta koos planeerimisoskusega. Tehniline sooritus mõjutab otseselt löögi edukust, kas soovitud löögi sooritamisega saadakse edukalt hakkama. Teoreetilistest teadmistest ja planeerimisoskusest sõltub löökide keerukus, mida on vaja sooritada, et mängu lõpetada. Kavalalt planeerides ja läbimõeldud otsuseid langetades on võimalik teha mäng endale nii lihtsaks, et mängu lõpetamiseks on vaja sooritada vaid tehniliselt lihtsaid lööke. Kuigi mõlemad valdkonnad on omavahel tihedalt seotud, on tehnilist sooritust lihtsam kvalitatiivselt mõõta ja seega ka lihtsam parandada. See teadustöö keskendubki just tehnilisele sooritusele.

Tehnilises soorituses on lähemalt vaadates väga mitmeid samme ja võrdlemisi palju liikuvaid osasid. Nendest peamised on löögi joondamine, löögiasendi võtmine ja löögi sooritamine. Osa löögi sooritusest, millele keskendutakse on löögi sooritamine löögiasendisse kummardunult. Löögi joondamist ja löögiasendi võtmist on väga keeruline hinnata ühe mõõteseadmega. Löögi joondamisel on vaja võrdluspunkte kuulide asetusega ja löögiasendi võtmise hindamiseks on vaja jälgida kogu keha asendit. Löögi sooritamine löögiasendis jaotub liigutuste näol kolmeks osaks: soojendusliigutused, paus ja löögi lõplik sooritus. Soojendusliigutused on perioodilised edasi tagasi liigutused kiiga, mis jäljendavad löögi lõplikku sooritust. Paus on soojendusliigutuste ja löögi lõpliku sooritamise vahele jääv passiivne periood, mille jooksul mängija sooritab viimase kontrolli ja fikseerib pilgu. Löögi viimane osa on löögi lõplik sooritamine, ehk kii liigutamine tagasi kiirendamise võimaldamiseks ja seejärel kii viimine löögikuulini. Kõigi nende liigutuste vältel liigub kiid hoidev käsi koos kiiga ja seega on võimalik sellel käel asuva nutikella andureid kasutades teha järeldusi löögi sooritusе kohta.

Eelpool mainitu on põgus tutvustus piljardi ja kiispordi kohta. Detailselt on piljardit ja selle eripärasid tutvustatud näiteks Jack H. Koehleri teoses [1].

2.2 Alternatiivsed vahendid

Kõik varasemalt mainitud meetodid tehnika parandamiseks põhinevad kii liikumise jälgimisel ja jaotuvad kahte kategooriasse: visuaalne vaatlus ja mõõteseadmetega mõõtmine. Mõlemal lähenemisel on nii plusse kui miinuseid, järgnev selgitus on refereeritud Kinam Kim jt artiklist [2]. Visuaalne vaatlus on vaatlusaluse jaoks vähem invasiivne kui andurite kasutamine, kuna isiku külge ei ole vaja kinnitada mingeid seadmeid. Samas on visuaalne vaatlus kapriissem keskkonnategurite suhtes ja seega ka enamasti ebatäpsem kui mõõteseadmete kasutamine.

Enamlevinud meetod on koos treeningpartneriga treenimine. Selle lähenemise probleemiks on aga see, et alati pole võimalik treeningpartnerit leida ja seda eriti arvestades asjaolu, et tehnika parandamiseks peaks olema partner ka ise võrdlemisi tugeva tasemega mängija. Eeldusel, et on olemas sobiv treeningpartner on see lähenemine väga tõhus kuna tagasiside on vahetu ja teine mängija võib märgata probleeme, mis treenijal võivad sageli märkamata jääda.

Teine vaatlusel põhinev lähenemine on treeningu filmimine ja hiljem salvestuste analüüsimine. See annab mängijale võimaluse enda sooritust kõrvalt vaadata ja näha vigu, mida mängija lööki sooritades tähele ei pane. Probleem selle lähenemisega on aga see, et vaatenurk on kaamerat kasutades piiratud. Isegi mitut kaamerat kasutades on keeruline iga lööki näha optimaalse nurga alt. Teine probleem on sageli ka see, et kuigi mängija näeb enda sooritust kõrvalt, ei pruugi ta siiski teadmiste puudumise tõttu märgata enda tehnikas vigu.

Üks liigutuse mõõtmisel põhinev treeningvahend on *DigiCue*. See on inertsiaalsete liikumisanduritega seade, mis kinnitatakse kii külge, jooniselt 1 on näha seadme kiile kinnitamist. Seade mõõdab kii liikumist löökide ajal liikumisanduritega. Probleem selle seadmega seisneb selles, et kii kaalu muutmine on paljude mängijate jaoks problemaatiline. Treenimine raskema kiiga, kui võistlusolukorras mängitakse, võib teha mängule rohkem kahju kui kasu. Teine probleem selle seadmega on see, et tagasiside seadme poolt halvaks hinnatud löögile piirdub lihtsalt vibratsiooniga, seega ei saa selget tagasisidet, mis võiks paremini olla. Samas selle seadme juures on hea kasutusmugavus, jättes kõrvale kiile raskuse lisamise. Tagasiside löögile on vahetu ja seadmega toimetamine on lihtne ja kiire.

Joonis 1. *Treeningvahend DigiCue [3].*

Järgmine võimalus on kasutada *Cue Measure* mobiilirakendust. Rakendus kasutab nutitelefoni liikumisandureid löögi sooritusе mõõtmiseks, kuid annab täpsemat tagasisidet, kui lihtsalt binaarne „hea“ või „halb“ sooritus. Selle rakenduse negatiivseks pooleks on see, et rakendus eeldab nutitelefoni kinnitamist kii külge, telefoni kinnitamist kiile on näha jooniselt 2. Selliselt nutitelefoni mõõteseadmete kasutamine võimendab oluliselt eelmise seadme juures välja toodud probleemi kiile raskuse lisamisega.

Joonis 2. *Kiile kinnitatud telefon.*

2.3 Inertsiaalsete liikumisandurite kasutamine

Liigutuse jälgimiseks kasutatakse andmeid kiirendusandurist ja güroskoobist. Kiirendusandur mõõdab kolmele teljele mõjuvat lineaarkiirendust. Güroskoop mõõdab samale kolmele teljele mõjuvat pöördeimpulssi. Sooituse edukuse hindamine toimub liikumisanduritest kogutud andmete analüüsimisel masinõppe mudeliga.

Kiirendusandurite kasutamisel ja nendega andmete kogumisel on oluline andurite paigutus ja nende kalibreerimine. Victor A. Carmona-Ortiz jt [4] on kirjutanud, et kehal kantavate kiirendusandurite kalibreerimiseks on üks võimalus kasutada ettemääratud standardseid asendeid ja jälgida andurite edastatavat informatsiooni. Sellisteks asenditeks on näiteks N-poos, milles on käed külgedel rippumas, või T-poos, milles on käed külgedele sirutatud. Kogutud info põhjal saab viia andurite ja inimese kehaosade koordinaatide süsteemid omavahel vastavusse. Alternatiivse lahenduse on välja pakkunud Jochen Tautges jt [5]. Nimelt, kui mõelda andurite paigutus hästi läbi ja fikseerida need kehale nii, et anduri ja keha koordinaatsüsteemid kattuvad, võib teatud rakenduste jaoks kalibreerimist vältida. Selles töös

õnnestus toimida analoogselt Jochen Tautges-i protseduurile. Kell fikseeriti randmele tugevamini kui tavalise kandmise puhul, mis kõrvaldas kella üleliigse loksumise randmel ja anduritest saadavad andmed olid puhtamad.

2.4 Masinõppe mudeli treenimine

Lisaks anduri paigutusele on oluline läbi mõelda kogutud andmehulkade töötlemine. Oluline otsuse koht on, kas liigutuste ära tundmiseks korraldada kogutud andmete võrdlus standardiseeritud andmehulga vastu, mis kujuneb üle kõigi katsete ja katseisikute, või ühele katseisikule personaliseeritud andmehulga vastu. Chao Shen jt [6] on sellise võrdluse kohta kirjutanud, et personaliseeritud andmehulga vastu võrdlemine annab enamasti küll täpsemaid tulemusi liigutuste ära tundmise puhul. Kuid standardiseeritud andmehulga vastu võrdlemine ebanormaalsete andmete suhtes vähem tundlik.

Esialgse eesmärgi täitmiseks ehk lööki tuvastava ja seda hindava rakenduse täieliku ja optimaalse realiseerimise jaoks oleks vaja luua kaks masinõppe mudelit. Üks mudel löögi liigutuse ära tundmiseks ja teine löögi liigutusele tagasiside koostamiseks. Kui liigutuse ära tundmist ja vastava andmehulga lõikamist oleks võimalik teha jooksvast andmevoost, vähendaks see märkimisväärselt töödeldavat andmehulka. Analüüsi teise sammuna on vaja löögi sooritust hinnata ja anda tagasiside sooritatud löögi kohta.

Tulenevalt asjaolust, et löögi sooritusele antav tagasiside põhineb hinnatava liigutuse kõrvalekaldest keskmisest sooritusest, oleks personaliseeritud andmehulga kasutamine liigutuse hindamiseks ja tagasiside andmiseks võrdlemisi halb valik. Sellise lahendusega saadav tagasiside ei ole eriti väärtuslik löögitehnika parandamisel, kuna sel juhul oleks kasutaja personaalse andmehulga keskmise väärtuse sees ka kasutaja personaalsed löögitehnika vead. Selline mudel väärtustab stabiilsust, seega ei ole võimalik sellise mudeli abil tehnikat treenides teha reaalseid edasiminekuid parema löögitehnika suunas. Antud olukorras oleks rakenduse eesmärgi täitmiseks kasulikum kasutada standardiseeritud mudelit. Kui mudelit treenida võimalikult paljude erinevate mängijate sooritatud löökidega, siis võiksid mudelisse koonduda parimad levinud praktikad. Probleem selle lähenemise juures oleks aga see, et standardiseeritud mudeli treenimiseks vajalik andmehulk on väga suur, selle katseline kogumine oli töö tegemise ajal kättesaadavates tingimustes väga keeruline.

Lisaks andmehulgale, mille peal mudelit treenida, on vaja langetada valik, kuidas löögi soorituse kohta tagasisidet anda. Tagasiside võib olla binaarne või tagasiside hõlmata mitmeid

alamklasse. Binaarse väljundiga tagasisidet oleks lihtsam teostada, aga selle reaalne väärtus treeningul oleks piiratud, kuna mängijal on raske aru saada, mis täpselt oli valesti.

2.4.1 Turi Create ja selle abil mudeli koostamine

Tänapäeval on enamlevinud masinõppe tööriistadeks näiteks *TensorFlow* ja *Keras*. Selle töö raames valiti mudeli treenimiseks Apple'i vabavaraline masinõppe teek nimega *Turi Create*[1]. See teek on suunatud kasutajatele kes ei ole spetsialiseerunud masinõppele, mis tähendab, et selle kasutamiseks ei eeldata kasutajalt spetsiifilisi teadmisi masinõppe hingeelust. Jonathan Balaban [7] on öelnud, et teek on vägagi võimekas, hoolimata sellest, et see on suunatud tavakasutajatele. Lisaks kasutajasõbralikkusele on lihtne selle teegi abil treenitud mudeleid eksportida iOS arenduses kasutatavas formaadis.

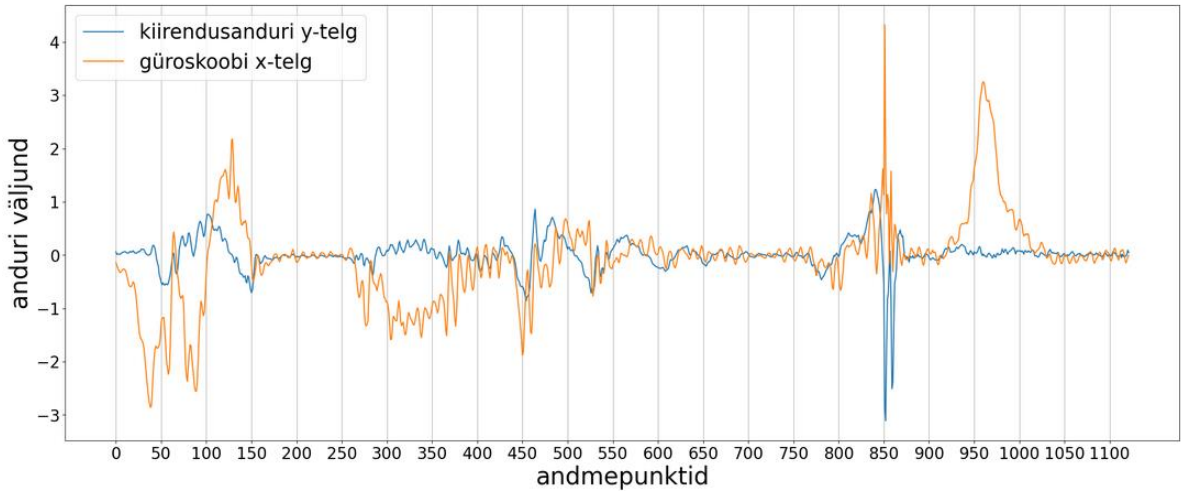
Teeki kasutades on võimalik luua väga kompaktseid masinõppe mudeleid, mida saab mugavalt teisendada kasutamiseks mobiilirakendustes. Mudelite treenimine on optimeeritud nii, et seda saaks teha personaalarvutis ilma suurte riistvaraliste nõudmisteta. Üheks suureks teguriks mudeli treenimise optimeerimise juures on masinõppe teegi kasutatav andmestruktuur *SFrame*. Selle põhiline eelis on see, et treenimiseks kasutatavaid andmeid ei hoita arvuti operatiivmälus[2]. Andmed laetakse operatiivmällu vastavalt vajadusele, mis võib küll protsessi mõnevõrra aeglustada, aga samas on tänu sellele võimalik treeningprotsess läbi viia tavalist kättesaadavat riistvara kasutades. Alternatiivne näide andmestruktuurist on tänapäeval laialdaselt kasutusel olev *Pandas DataFrame*, mille kasutamisel on vaja see tervikuna laadida operatiivmällu ja seega suurema koguse andmetega töötlemisel on see lahendus riistvara poolest halvasti skaleeruv.

Masinõppele läheneti selles töös musta kasti meetodit kasutades. See tähendab, et koguti andmeid, treeniti nende andmete pealt mudel ja testiti treenitud mudeli reaalset sooritust mudeli poolt seni nägemata ehk uute löökide ära tundmisel. *Turi Create* teegis on alamklass *activity classifier*, mis on spetsialiseerunud liigutuste klassifitseerimisele. Sagar Howal [8] näitas, kui lihtsalt on võimalik seda teeki kasutades luua töötav mudel liigutuste ära tundmiseks. Sellest näitest tuli välja, et andmetele pole vaja teha muud eeltöötlust kui andmepunktide sildistamine neile vastava liigutuse või klassiga. Kuigi *Turi Create*'i kasutades pole loodavaid mudeleid võimalik väga kergelt peenhäälestada, on tööriist piisavalt hästi optimeeritud, et enamasti ei

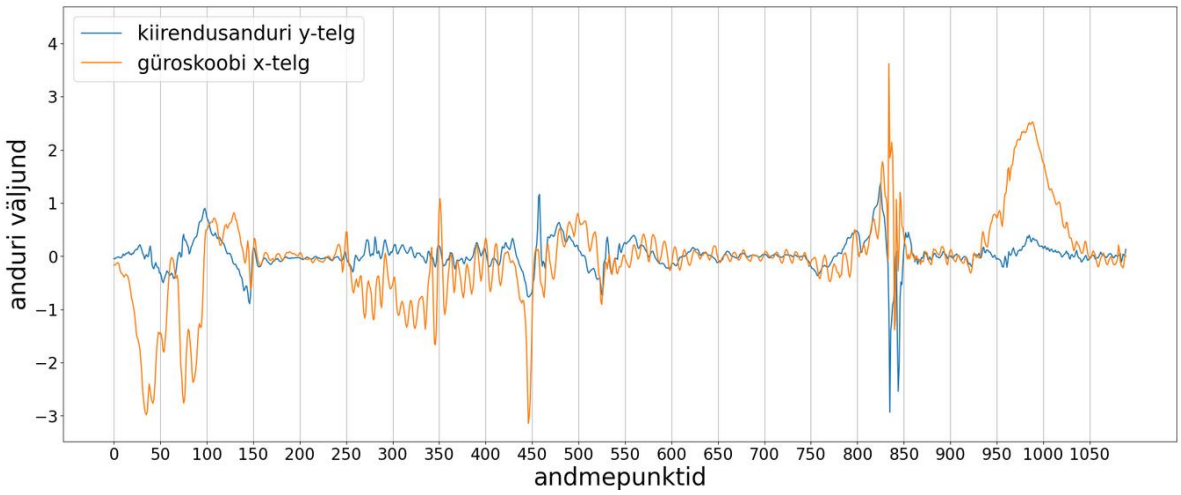
[1] Turi Create: https://github.com/apple/turicreate
[2] SFrame dokumentatsioon: https://apple.github.io/turicreate/docs/api/generated/turicreate.SFrame.html

olegi peenhäälestust vaja. Sellised omadused on heaks eelduseks käsitlemaks masinõpet musta kastina.

Liigutuste klassifitseerimisele suunatud alamklassis on kasutusel konvolutsiooniline ja rekurrentne lähenemine [9]. Konvolutsioonilised närvivõrgud on laialdaselt kasutusel näiteks piltide ära tundmisel, see lähenemine põhineb andmehulgast mustrite leidmisel ja leitud mustrite vahel seoste loomisel [10]. Seetõttu sobib selline lähenemine ka liigutuste klassifitseerimiseks, kuna ühte tüüpi liigutuste puhul kujunevad välja selged mustrid andmehulkades. Näiteks joonised 3 ja 4, millel on näidatud kiirendusanduri y-telje ja güroskoobi x-telje andmeid kahe erineva löögi sessiooni vältel. Mõlemal graafikul jääb löögi sooritus laias laastus x-telje punktide 250 ja 900 vahele. Nendes vahemikes on näha selgelt kattuvad mustrid, mis on hea eeldus nende andmete ära tundmiseks kasutades masinõpet.



Joonis 3. *Esimene näidislöök.*



Joonis 4. *Teine näidislöök.*

Turi Create'i liigutuste klassifikaator kasutab ka rekurrentset kihti. Rekurrentsust kasutavad mudelid sobivad ajas muutuvate andmete klassifitseerimiseks, kuna arvestavad ka möödunud andmetega. Töö raames käsitletavad andmed on samuti ajas muutuvad ja jälgivad sarnaseid mustreid.

Peatükis tutvustati kiispordi eripärasid, alternatiivseid lahendusi piljardi treeningute toetamiseks, inertsiaalsete liikumisandurite kasutamist, masinõppe mudeli treenimist ning töö raames valitud vahendit masinõppe mudeli treenimiseks. Järgmises peatükis tutvustatakse bakalaureusetöö raames tehtud praktilist tööd.

3. Töö käik

Peatükis tutvustatakse teadustöö raames tehtud praktilist tööd. Töö praktilise osa peamised sammud on nutikella anduritest andmete kätte saamine, andmete kogumine ja ettevalmistus masinõppe mudeli treenimiseks, masinõppe mudeli treenimine ja treenitud mudeli rakendamine mobiilirakenduse prototüübis löögi andmete kogumiseks.

3.1 Ettevalmistus

Töö esimene samm oli iOS arenduskeele Swift'iga tutvumine. Selleks kasutati Apple'i dokumentatsiooni[3] ja vabavaralisi mitteakadeemilisi õppematerjale (veebiblogid ja foorumid). Järgmine samm oli nutikella anduritest kätte saada liikumisandurite andmed. Kasutades sisseehitatud teeki *CoreMotion* oli andmete kättesaamine võrdlemisi kerge. Sisseehitatud funktsionaalsuses on olemas ka teek *CMDeviceMotion*, mis väljastab eeltöötluse läbinud liikumisandurite andmed. Selle eeltöötluse peamine kasutegur on gravitatsiooni mõju eemaldamine kiirendusanduri andmetest. See tähendab, et saadavate väärtuste näol on tegemist lineaarsete väärtustega, seega on need omavahel võrreldavad.

Liikumisanduritest andmete kogumisel osutus probleemiks see, et nutikella liikumisandurid lõpetavad andmete edastamise kui kasutaja seadmega otseselt ei interakteeru, ehk olukorras kus seade on puhkerežiimis. Näiteks kui kasutaja langetab käe ja seadme ekraan kustub. Lahendus sellele probleemile on jooksutada rakendust treeningrežiimis, mis on operatsioonisüsteemi sisseehitatud funktsionaalsus. Treeningrežiimis rakendust kasutades ei peatata liikumisandurite andmevoogu, kui seadme ekraan kustub. Selle kontseptsiooni tõestuseks võeti Apple'i loodud treeningute jälgimiseks mõeldud avatud lähtekoodiga näidisrakendus *SpeedySloth* [11]. Olemasolevat koodi muudeti ja sellesse lisati liikumisanduritest andmete väljastamine konsooli. Sellisel moel saadi kella liikumisanduritest pidev andmevoog, sõltumata sellest kas kasutaja on seadmega otseses interaktsioonis. Olles kontseptsiooni tõestusena saanud kätte katkematu andmevoo liikumisanduritest, loodi uus projekt, milles oli olemas ka rakenduse nutitelefoni komponent. Uude projekti lisati näidisrakenduse eeskujul võimekus jooksutada nutikella komponenti treeningrežiimis.

Ettevalmistuse juurde kuulus ka Jupyter Notebooki[4] üles seadmine, see keskkond võimaldab koostada skripte mitmete programmeerimiskeeltega. Masinõppe mudeli treenimine kasutades

[3] Apple'i arendaja dokumentatsioon: https://developer.apple.com/
[4] Jupyter Notebook: https://jupyter.org/

Turi Create teeki viidi läbi eelmainitud keskkonnas, kasutades programmeerimiskeelt Python[5]. Lisaks mudeli treenimisele teostati selles keskkonnas ka skriptidega andmetöötlus.

3.2 Andmete salvestamine

Peale andmevoole ligipääsemist oli vaja andmeid koguda masinõppe mudeli treenimiseks. Liikumisanduritest andmete kogumise sageduseks valiti 100 Hz. See valik põhines asjaolul, et uuritava liigutuse jaoks on täpsem mõõtmine parem, kuna liigutuse juures taheti uurida detaile. Samas pidi ka arvestama nutikella riistvaraliste piirangutega. Vastavalt dokumentatsioonis öeldule, on valitud sagedus toetatud enamike seadmete poolt [12]. Esimeses rakenduse katsetuses väljastati andurite andmed arenduseks kasutatavas arvutis konsooli. Kuna sellisel moel suurema koguse andmete kogumine oleks küllaltki ebaefektiivne, otsustati kogutud andmed kirjutada nutikella salvestusruumi. Salvestusruumi kirjutati andmed sõne faili kirjutamise meetodit kasutades. Arenduseks kasutatavast arvutist polnud aga võimalik otse kätte saada kellale salvestatud faile. Küll aga olid lihtsasti kättesaadavad telefoni salvestatud failid, seega otsustati kogutud andmed saata telefoni ja need hilisemaks töötluseks sinna salvestada.

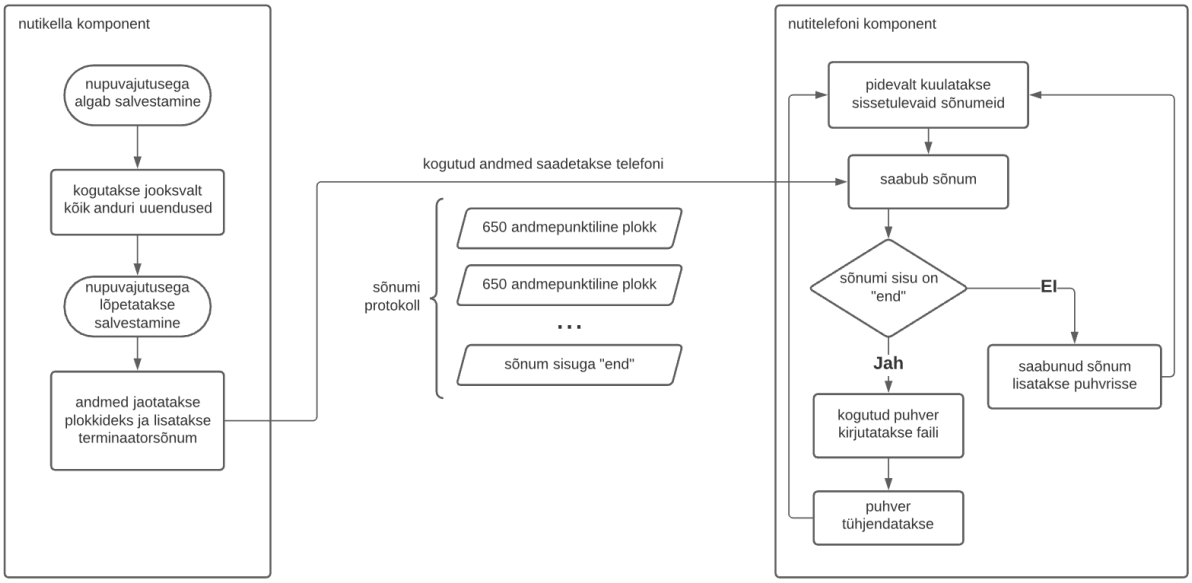
3.2.1 Andmete saatmine kellast telefoni

Telefoni salvestamiseks oli vaja saata andmed nutikellast telefoni. Nutikellas koguti andmed ja vormistati need .csv faili kirjutamiseks valmis oleva sõnena, mis saadeti seejärel kellast nutitelefoni. Kasutades *Watch Connectivity* raamistikku oli see tehniliselt üsna lihtne lahendus [13]. Esialgu saadeti ühe sõnumina terve kogutud sessioon ehk umbes 700 kuni 800 andmepunkti.

Pikemate sessioonide andmete salvestamisel osutus probleemiks ühe saadetava sõnumi piiratud maht. Sellise sõne kujul sai 100 Hz sagedusega salvestatud andmeid saata umbes kaheksa sekundi jagu. Selleks, et salvestatavad sessioonid saaksid olla pikemad, jaotati sessiooni jooksul kogutud andmehulk telefoni saatmise jaoks piisavalt väikesteks pakettideks. Paketi suuruseks valiti konservatiivselt 650 andmepunkti ehk .csv faili rida vormistatud sõnest. Need paketid saadeti järjest telefonile. Telefoni osapool rakendusest kuulas pidevalt sissetulevaid sõnumeid, esimese sõnumi saabudes hakati sõnumeid sõnesse kokku koguma. Kui telefon sai kätte andmevoo lõppu tähistava sõnumi, siis kirjutati kogutud andmed kõik ühte

[5] *Python*: https://www.python.org/

faili. Andmevoo lõppu tähistas sõnum *„end“*, sellist sõnumit ei sisaldunud mingil juhul andmevoo sisus. Andmete salvestamiseks valminud prototüübi loogikat on näha joonisel 5.

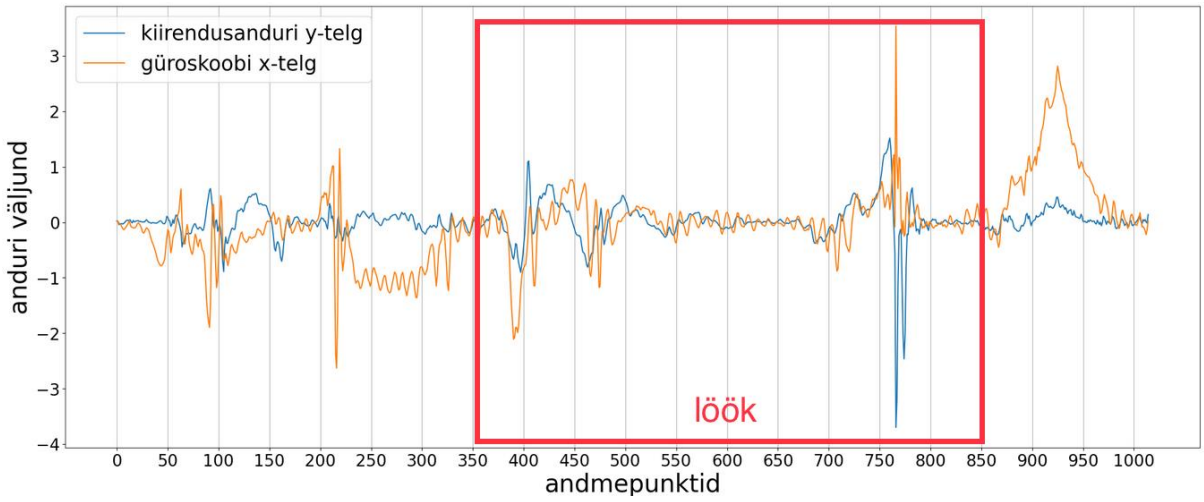


Joonis 5. *Andmete salvestamiseks valminud rakenduse loogika.*

3.3 Andmete kogumine

Treenitav mudel tegeleb andmevoost löögi ära tundmisega ehk mudeli väljund on binaarne, hinnatava andmeploki näol on tegemist kas löögi või millegi muuga ehk müraga. Kuna tegemist on binaarse mudeliga, siis jaotuvad andmed kahte osasse, positiivsed ehk lööki sisaldavad lõigud ja negatiivid ehk lööki mitte sisaldavad lõigud. Andmete kogumiseks oli valminud kaheosaline rakenduse prototüüp, mille moodustasid nutitelefoni ja nutikella komponendid. Nutikellas nupuvajutusega alustati nutikellas andmete kogumist ja peale teist nupuvajutust salvestati sessiooni jooksul kogutud andmed nutitelefoni salvestusruumi.

Negatiivsete lõikude esindamiseks koguti andmeid võimalikult suvaliste liigutuste kohta. Pärast nutikellast telefoni sõnumi saatmise piirangu lahendamist koguti negatiive umbes kümne minuti pikkuste sessioonidena. Positiive koguti umbes seitsme kuni kümne sekundi pikkuste sessioonidena. Selle aja sees oli üks löök, aga ka löögi alla mitte liigituvaid liigutusi. Need olid näiteks käe langetamine peale salvestamise alustamist, löögi ettevalmistus, pärast lööki püsti tõusmine ja salvestamise lõpetamiseks käe tõstmine. Joonisel 6 on näha üks positiivi sessioon. Punasega piiratud osa sees on lööki esindavad andmed, sellest alast välja poole jääb eelmainitud müra.



Joonis 6. *Positiivi sessioon, milles on märgitud lööki esindav osa.*

3.4 Mudeli treenimine

Mudeli treenimiseks rakendati juhendatud õppe meetodit ehk mudelile anti treenimiseks andmed siltidega, mis kirjeldavad tegevust, mida andmed sisaldavad. Mudeli treenimise juures oli kaks peamist parameetrit, mis muutsid treenimise ja seeläbi treenitud mudeli omadusi. Nendeks olid mudeli ennustusakna suurus ja andmehulga töötluste arv treeningsessiooni jooksul. Mudeli ennustusakna pikkus määrab mitut üksikut andmepunkti mudel ühe hinnangu väljastamiseks kasutab. Selleks suuruseks valiti 100 andmepunkti, kuna andmed salvestati sagedusega 100 Hz. Andmehulga töötluste arv määrab mitu kordust tehakse üle andmehulga mudeli treenimisel. Selleks suuruseks valiti 40, kuna töödeldavate andmehulkadega testides andis selline suurus valdavalt suurimaid täpsuseid. Nii vähemate kui ka rohkemate kordustega hakkas mudeli täpsus treenimisel vähenema.

Mudeli treenimisele läheneti iteratsioonidena. Koguti hulk andmeid, treeniti mudel ja testiti selle sooritust. Esialgu läheneti testimisele masinõppe traditsioonilise töövõttega, milles jaotatakse kogutud andmed enne mudeli treenimist treeninghulgaks ja testhulgaks. Treeninghulgaga treenitakse mudel ja testhulgaga kontrollitakse mudeli hinnangute täpsust. Lisaks testhulgaga treenimisele testiti ka mudeli reaalelulist sooritust.

Mudeli reaaleluliseks testimiseks modifitseeriti andmete kogumiseks kasutatud rakenduse prototüüpi. Andurite jooksvast andmevoost koguti mudeli ennustusaknale vastava suurusega andmeplokid ja lasti mudelil hinnata, kas andmeplokis sisalduvad andmed esindavad lööki või müra. Iga andmeploki kohta saadud hinnang saadeti nutitelefoni, mis kuvas saabunud sõnumi ekraanile. Testimise käigus oli telefon mängija nägemisväljas ja mängija sai seeläbi hinnata

mudeli klassifitseerimise täpsust. Esimese iteratsiooni mudeli reaalajas testimisel märgati märkimisväärset ebakõla võrreldes testhulgaga skriptis saadud tulemustega. Skriptis näitas mudel oluliselt suuremat täpsust kui reaalses katses. Kuna mudeli reaalne sooritus on olulisem kui teoreetiline täpsus skriptis testides, otsustati skriptis testimine kõrvale jätta ja esialgset andmehulka mitte osadeks jagada. See võimaldas mudeli treenimiseks kasutatavat andmehulka suurendada, kuna enam ei jäetud osa kogutud andmetest testimiseks kõrvale.

Keskendudes reaalelulisele testimisele nähti selgemini mudeli käitumist erinevate sisendite korral. Mudeli treenimiseks kasutatavat andmehulka korrigeeriti vastavalt testimise jooksul tehtud tähelepanekutele ja protsessi korrati kuni mudeli sooritusega rahule jäädi.

3.4.1 Esimene iteratsioon

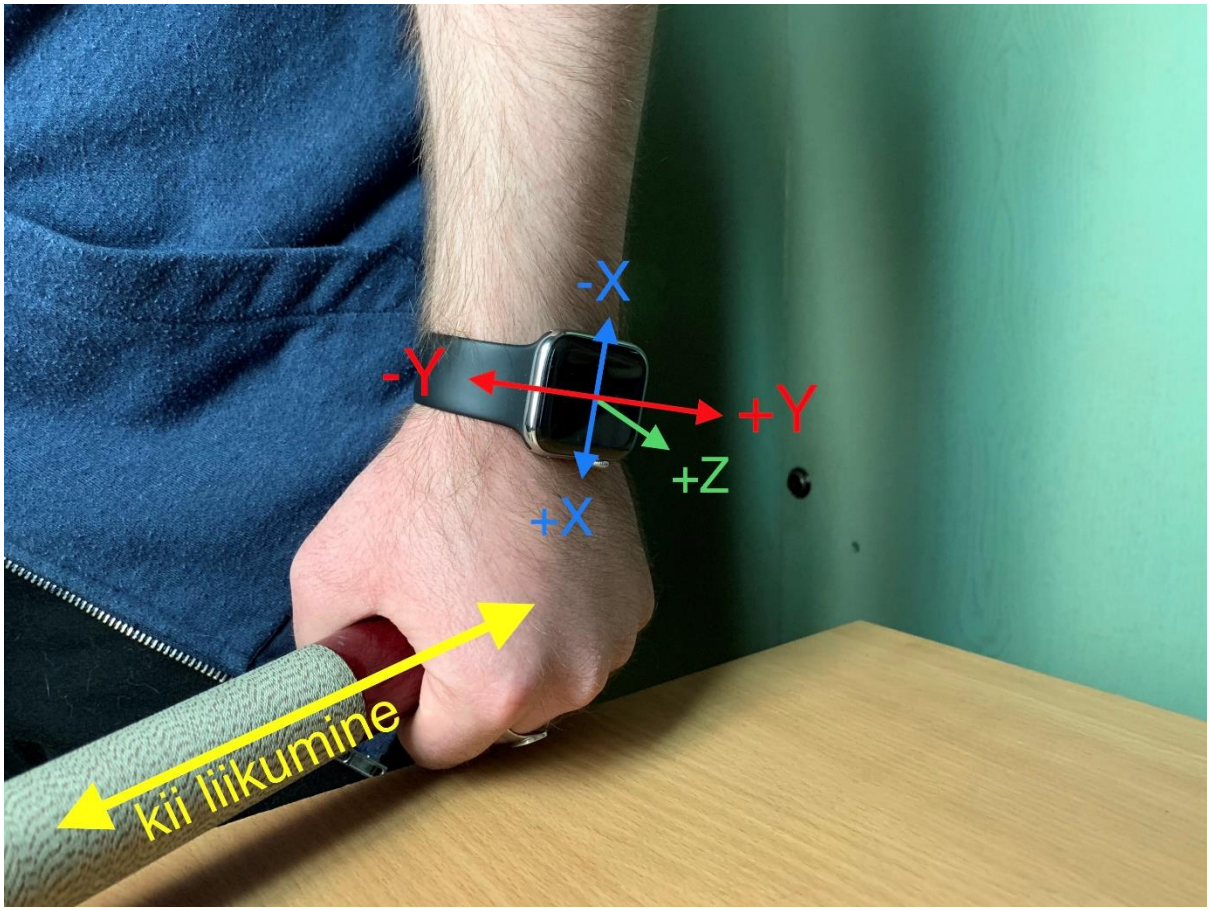
Esimese iteratsiooni mudeli treenimisel liigitati terve positiivi sessioon löögina. Positiivide sessioone oli 33 ja negatiivide sessioone umbes 100. Enne negatiivide kasutusele võtmist jaotati kogutud negatiivide pikad sessioonid väiksemateks, et nende pikkus oleks analoogne positiivide pikkusega. Jagamise tulemusena oli nii positiivi kui ka negatiivi ühes sessioonis umbes 800 andmepunkti. Selliste andmetega treenitud mudel näitas treenimise käigus viimaste kordustega treenimise täpsuseks 1 ja valideerimise täpsuseks 0.97. See-eest oli mudeli reaaleluline sooritus halb. Testimise käigus tundus mudeli ennustus juhuslik, ei olnud aru saada et mudeli väljund sõltuks sisendist. Ebaõnnestunud katse pealt püstitati hüpotees, et peamine probleem tuleneb liiga väikesest andmehulgast.

3.4.2 Teine iteratsioon

Teise iteratsiooni mudeli jaoks koguti juurde nii positiivide kui ka negatiivide sessioone. Juurde koguti 50 sessiooni positiive ja umbes sama palju sessioone negatiive, seega kokku oli 83 sessiooni positiive ja 150 sessiooni negatiive. Kogutud sessioonid sildistati analoogselt esimese iteratsiooniga ehk terve positiivi sessioon loeti löögiks ja terve negatiivi sessioon loeti müraks. Kuigi positiivide sessiooni alguses ja lõpus oli ka müra, oli lootus, et seda on piisavalt vähe ja mudel suudab siiski leida üles löögi mustrid. Suurema andmemahuga treenitud teine mudel näitas samuti treenimise viimaste korduste käigus treenimise täpsuseks 1 ja valideerimise täpsuseks 0.98. Mudelit testides oli tulemus taaskord sama, mis eelmisel korral. Hinnangud liigutuse klassifitseerimisel tundusid juhuslikud ja sisendist mitte sõltuvad. Sellest järeldati, et peamine probleem ei olnud andmete hulgas.

3.4.3 Kolmas iteratsioon

Järgmise mudeli iteratsiooni jaoks otsustati parandada andmete sildistamise täpsust, selleks lõigati positiivide sessioonidest välja löögile eelnev ja järgnev müra. Seda tehti käsitsi iga positiivi sessiooni jaoks. Andmete lõikamiseks vaadati graafiliselt läbi iga sessiooni kohta kogutud andmed. Otsus andmete lõikamise kohta tehti peamiselt vaadates kiirendusanduri y-telje andmeid. Selline otsus tehti seetõttu, et kella andurite teljestiku y-telg on löögiasendis kii liikumisega kõige paremini joondatud. Joonisel 7 on näha nutikella paiknemine kii ja liigutuse suuna suhtes. Joonisele on märgitud kollase noolega kii liikumise suund ja punase, rohelise ning sinise noolega kella liikumisandurite teljestik. Peamiselt vaid ühe telje andmete vaatlemine võimaldas märkimisväärselt vähendada iga sessiooni töötlemiseks kuluvat aega. Kõigepealt vaadati läbi kümmekond lööki, et näha milline muster peaks löögi ajal andmetes välja kujunema ning seejärel käidi läbi kõik positiivide sessioonid ja lõigati nendest välja vaid lööki esindav osa.



Joonis 7. *Kella andurite teljestiku asetsemine kii liikumise suhtes.*

Andmete kogus sessioonide näol jäi mudeli kolmanda iteratsiooni treenimisel samaks nagu eelmisel iteratsioonil. Mudel näitas treenimise viimaste korduste käigus taaskord väga kõrgeid täpsuseid nii treenimise kui ka valideerimise täpsus oli 1. Selle iteratsiooniga oli testimise käigus nähtud tulemus märkimisväärselt parem. Mudel tundis võrdlemisi usaldusväärselt ära löögi alguse, kuigi jäi mõni kord löögi ära tundmisega hiljaks. Põhiline probleem oli löögi lõpu ära tundmisega. Peale löögi sooritamist jäi mudel sageli kinni positiivse hinnangu peale, seega andes valepositiivse väljundi. Sellegipoolest oli selgelt näha, et positiivide eristamine mürast sessioonide lõikamise abil oli avaldanud soovitud mõju mudeli täpsusele uute andmete klassifitseerimisel. Kuna peamine probleem oli löögi lõpu ära tundmisega, püstitati uus hüpotees, et mudeli täpsust võiks parandada positiivide sessioonides löögi andmete ümber müra konteksti andmine.

3.4.4 Probleem andmete sildistamisega

Mudeli kolmanda iteratsiooni jaoks andmeid modifitseerides kerkis esile sagedane probleem juhendatud õppe kasutamisel. Andmete käsitsi sildistamine oli väga ajakulukas, ühe positiivi sessiooni käsitsi üle käimine võttis aega minimaalselt 30 sekundit. Kolmanda iteratsiooni jaoks kogutud 80 sessiooni töötlemiseks kulus ligi tund. Kuna oli teada, et valmis rakenduse loomiseks oli vaja koguda märkimisväärselt rohkem andmeid, otsustati töö eesmärk ümber sõnastada.

Korrigeeritud eesmärgiks sai luua masinõppe mudel ja rakenduse prototüüp, mille koostöös saaks automatiseeritult koguda ainult lööki sisaldavaid andmeid. See võimaldaks ilma ajakuluka käsitsi töötluseta koguda suuremal hulgal positiive, millega treenida löögi hindamiseks ja tagasiside koostamiseks vajalikku mudelit.

3.4.5 Neljas iteratsioon

Neljanda iteratsiooni mudeli jaoks käidi esialgsed positiivide sessioonid taas ühekaupa üle. Müra sessioonide alguses ja lõpus jäeti välja lõikamata, kuid sildistati mürana. Seega saadi positiivid mis sisaldasid lisaks löögile ka korrektselt sildistatud müra. Mudeli treenimiseks kasutatud andmete hulka ei muudetud ja ka negatiivide sessioone ei muudetud. Selle iteratsiooni mudel sai treenimise viimasel kordusel treenimise täpsuseks 0.969 ja validatsiooni täpsuseks 0.967. Kuigi treenimisel näidatud täpsused kahanesid mõnevõrra, paranes testimisel nähtud mudeli sooritus taaskord märkimisväärselt. Eelmise iteratsiooni mudeli peamine probleem sai täpsema sildistamisega parandatud. Mudel ei jäänud peale löögi sooritamist kinni

valepositiivse väljundi peale. Ka löögi alguse ära tundmine muutus usaldusväärsemaks - kui eelmine mudeli iteratsioon jäi löögi alguses positiivse väljundiga märgatavalt hiljaks, siis neljas iteratsioon andis positiivse väljundi enamasti kas juba löögiasendisse laskudes või esimese soojendusliigutuse sooritamise peale.

3.4.6 Viies iteratsioon

Kuigi üldiselt oli neljanda iteratsiooni mudeli sooritus juba rahuldav, märgati testimise käigus ühte uut probleemi. Mudel andis pika passiivse sisendi korral valepositiivse väljundi. See tulenes passiivse sisendi sarnasusest soojendusliigutuste ja löögi lõpliku sooritamise vahel oleva pausiga. Selle probleemi lahendamiseks prooviti suurendada mudeli treenimiseks kasutatavat andmehulka. Viienda iteratsiooni mudeli jaoks koguti seega andmeid juurde ja toimiti positiivide sildistamisega analoogselt eelmisele iteratsioonile. Positiive saadi kokku 230 sessiooni ja negatiive kokku 660 sessiooni. Suurendatud andmehulgaga treenitud mudel näitas treenimise viimasel kordusel treenimise täpsuseks 0.98 ja valideerimise täpsuseks 0.972. Testimise käigus nähti, et passiivse sisendi korral valepositiivsete väljundite osakaal küll vähenes, kuid probleem ei lahenenud täielikult.

Kuigi viienda iteratsiooni mudel ei parandanud neljandas iteratsioonis märgatud probleemi täielikult, oli siiski näha probleemi leevenemist. Selleks, et mudel annaks passiivsete sisendite korral pideva valepositiivse väljundi, pidi passiivne sisend kestma kauem, kui eelmise mudeliga. Üldiselt oli viimase mudeli sooritus piisavalt hea, et selle töö raames lugeda mudeli arendamine lõpetatuks. See otsus põhines asjaolul, et mudel tundis usaldusväärselt ära löögi alguse ja löögi lõpu ning ei andnud vahepeal valenegatiivseid väljundeid.

3.5 Treenitud mudeli kasutamine löögi andmete kogumiseks

Viimase iteratsiooni mudeli sooritus oli testimise käigus piisavalt usaldusväärne, seega integreeriti see mudel löögi andmete kogumiseks testimisrakendusse. Selleks arendati andmete kogumiseks kasutatud rakendust edasi. Varasemalt koguti treeningsessiooni ajal kõik andurite uuendused kokku ja saadeti sessiooni lõpus salvestamiseks telefoni. Edasiarendusega filtreeriti treenitud mudelit kasutades andmete kogumist. Aktiivse sessiooni ajal koguti anduritest sama palju andmepunkte kui on mudeli ennustusakna suurus, treenitud mudeli puhul oli selleks suuruseks 100 andmepunkti. Täitunud plokki hinnati mudeliga, kui mudel klassifitseeris ploki löögina, siis salvestati see plokk ajutiselt hilisemaks edasi saatmiseks. Peale ploki hindamist see tühjendati ja protsessi korrati, kuni sessioon lõpetati kellal nupuvajutusega. Selliselt koguti

sessiooni jooksul kokku ainult löögina klassifitseeritud andmeplokid. Sessiooni lõppedes saadeti kogutud andmed telefoni püsivaks salvestamiseks kasutades peatükis 3.2.1 kirjeldatud metoodikat.

Peatükis kirjeldati praktilise osa ettevalmistust Swift programmeerimiskeelega tutvumisel ja Jupyter Notebook arenduskeskkonna üles seadmist. Andmete kogumist, selle käigus esile kerkinud probleeme ja nende lahendamist. Kirjeldati ka mudeli treenimise metoodikat ja treenimisel läbitud iteratsioone. Järgmises peatükis tuleb juttu töö tulemustest.

4. Tulemused ja analüüs

Peatükis kirjeldatakse töö käigus saavutatud tulemusi ja analüüsitakse neid. Töö tulemusena valmis masinõppe mudel ja kaheosaline rakenduse prototüüp (nutitelefoni ja nutikella rakendus). Masinõppe mudel tunneb liikumisandurite andmetest ära piljardi lööki sisaldavad andmeplokid. Rakenduses mudelit kasutades filtreeritakse andurite jooksvast andmevoost välja lööki sisaldavad lõigud. Kirjeldatakse ka iga mudeli iteratsiooni sooritust samade testandmete klassifitseerimisel.

4.1 Mudeli iteratsioonide võrdlus

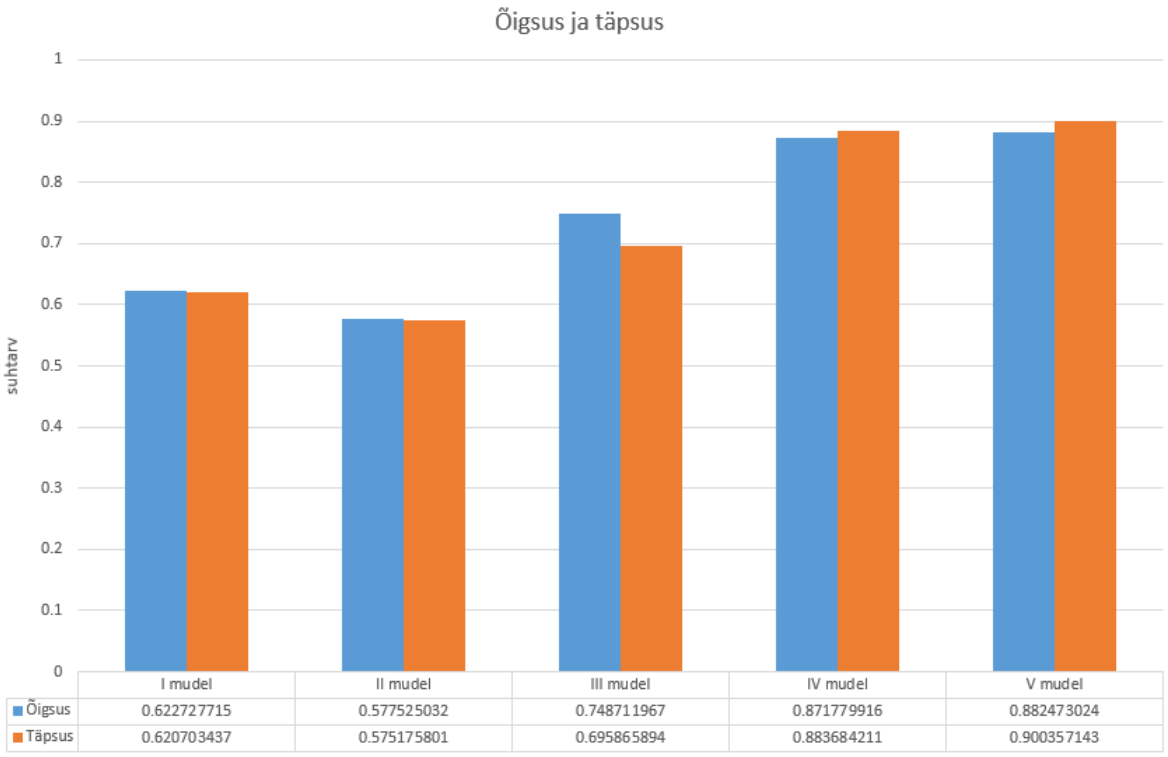
Mudeli arendamise ja testimise käigus nähti iga iteratsiooni sooritust päris andmete peal ja hinnati subjektiivselt mis on hästi ja mis on halvasti. Selleks, et saada mudeli staadiumite sooritusest objektiivne ülevaade kasutatakse kvantitatiivset analüüsi. Selleks et mudelite valideerimistulemuste võrdlus oleks võimalikult ühtlane, kõrvaldati võimalikult palju muutujaid. Kõigi iteratsioonide mudelid treeniti sama algpunktiga mudeli esmasel loomisel ja valideerimiseks kasutati kõigi mudelite puhul sama andmehulka.

Valideerimises kasutatavateks andmeteks koguti uued positiivide sessioonid. Selles kogumis on 18 positiivi sessiooni, mis on analoogsed treenimiseks kasutatud positiivide sessioonidele, kuid ei sisaldunud treeningandmete hulgas ja on seega mudelite jaoks uued. Iga sessioon sisaldab ühte lööki ja sellele eelnevat ning järgnevat müra. Sessioonid sildistati käsitsi, vastavalt neljanda mudeli treenimise iteratsiooni jooksul kujunenud rutiinile. Terviklikke negatiivide sessioone ei kaasatud valideerimishulka, kuna mudeli sooritust löökide ära tundmisel sooviti hinnata võimalikult täpselt. Mudelite hulgas oli probleem pika passiivse perioodi jooksul valepositiivse väljundiga, kuid selle probleemi saab lahendada mudeli kogutud andmete automatiseeritud järeltöötlusega. Lahendusest räägitakse peatükis 5.1. Kuna see on spetsiifiline erandjuhtum andmete hulgas ja sellele on olemas ka potentsiaalne lahendus, ei kaasatud seda olukorda kirjeldavaid andmeid valideerimishulka.

Valideerimise tulemusena saadud mudeli täpsust kirjeldatavate karakteristikute hulgast kasutatakse mudeli kirjeldamiseks kolme karakteristikut: *accuracy*, *precision* ja *recall*. Koo Ping Shung [14] on neid karakteristikuid järgnevalt kirjeldanud. *Accuracy* ehk õigsus on õigesti klassifitseeritud andmepunktide osakaal kõigi klassifitseeritud andmepunktide hulgas. See karakteristik väljendab üldiselt mudeli hinnangute korrektsust. *Precision* ehk täpsus on õigesti klassifitseeritud positiivide osakaal kõigi positiivide hulgas. See karakteristik kirjeldab, kui

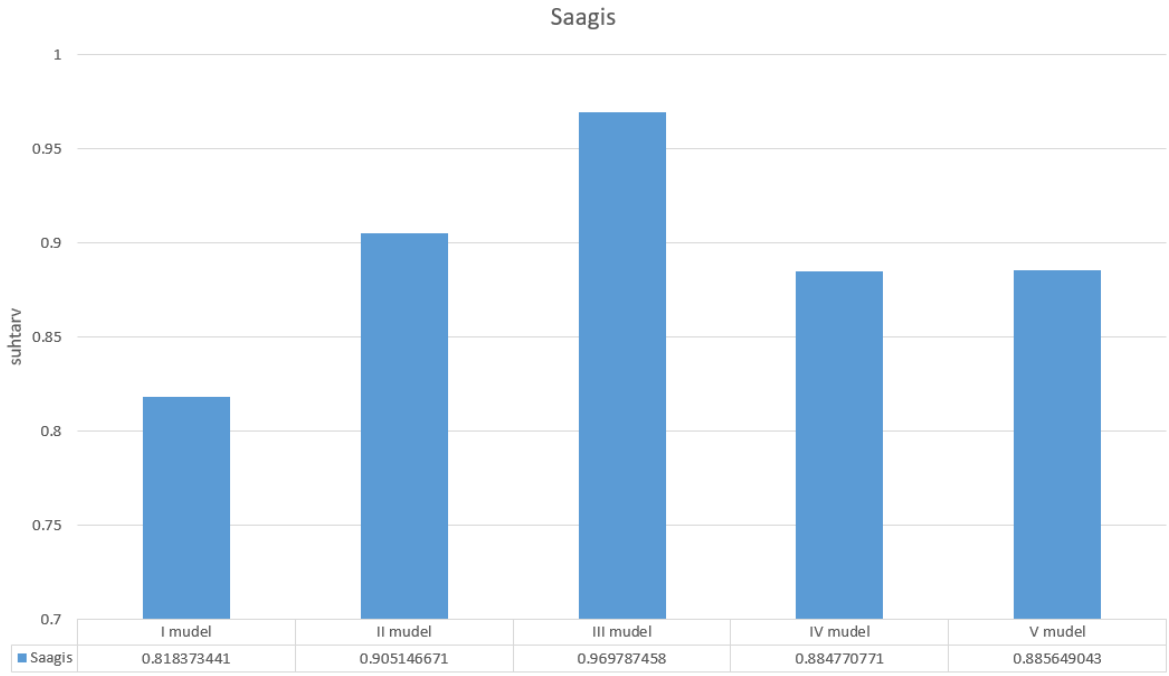
palju positiividest on korrektselt klassifitseeritud ja seda on hea jälgida juhul, kui valepositiivsete tulemuste kulu on suur. *Recall* ehk saagis on õigesti klassifitseeritud positiivide osakaal valepositiivide ja tõeste positiivide hulgas. See karakteristik kirjeldab, kui palju positiive mudel korrektselt klassifitseeris ja seda on hea jälgida juhul, kui valenegatiivsete kulu on suur.

Joonisel 8 on näha iga mudeli iteratsiooni jaoks valideerimisest saadud õigsuse ja täpsuse väärtus. Üldjoontes peegeldavad need karakteristikud mudeli testimise käigus nähtud käitumist. Esimese kahe iteratsiooni vahel ei olnud suurt erinevust, kuid kolmanda ja neljanda iteratsiooniga paranes täpsus hüppeliselt. Neljanda ja viienda iteratsiooni vahel ei olnud taaskord näha suurt erinevust. Hüppelisi muutusi seletab treenimise andmehulga fundamentaalsem muutus. Esimese hüppe korral lõigati positiividest välja müra ja teise hüppe korral võeti kasutusele lõikamata positiivid, kuid sildistati need täpsemalt. Mõlemad andmete muutmised andsid mudeli õigsuse ja täpsuse paranemise näol soovitud tulemuse. Väiksemad muutused on tingitud andmehulga mahu muutumisest. Kuna andmehulga mahu suurendamine ei muutnud märgatavalt mudeli käitumist, saab järeldada, et mudeli käitumises erinevuste nägemises peab andmehulka rohkem suurendama.



Joonis 8. *Valideerimise õigsus ja täpsus igal mudeli iteratsioonil.*

Joonisel 9 on näha iga mudeli iteratsiooni jaoks saadud saagise väärtus. Esimese kolme iteratsiooni puhul on näha selle karakteristiku väärtuse tõusu. Selle põhjuseks on asjaolu et iga iteratsiooniga tundis mudel järjest paremini ära positiive, aga seejuures andis ka järjest rohkem valepositiivseid hinnanguid. Neljanda iteratsiooniga karakteristiku väärtus kahanes ja viienda iteratsiooniga jäi stabiilseks.



Joonis 9. *Valideerimisel saadud saagise väärtus.*

Mudelit soovitakse rakendada löögi andmete ehk positiivide kogumiseks. Löögina klassifitseeritud andmeplokid kogutakse faili hilisemaks töötluseks. Kui kogutud andmete hulgas on valepositiivse sildiga müra, segab see oluliselt andmete edasist kasutust ja nõuab ajakulukat töötlust müra eemaldamiseks. Seega on olulisem karakteristik täpsus, kuna see väljendab mudeli võimet tagastada vaid tõeselt positiivseid väljundeid.

4.2 Mudeli rakendamine teiste mängijate peal

Aastal 2021 COVID-19 tõttu kevadel kehtestatud eriolukorrast tingituna ei olnud töö tegemise ajal võimalik teiste kogenud mängijate löökide pealt andmeid koguda. Seega olid kõik treenimiseks kasutatud positiivid kogutud ühe mängija löökide pealt. Kuna mängija sooritab lööke oma dominantse käega on ühe mängija pealt kogutud andmete hulgas esindatud vaid ühe käega sooritatud löögid. Paratamatult saadakse selliste andmetega reeglina ülesobitatud mudel.

Hindamaks, kui suur probleem on ülesobitamine treenitud mudeli juures, testiti mudelit mängija mittedominantse käega. Katses, kus nutikell oli mängija mittedominantsel käel ei tundnud mudel ära mitte ühtegi lööki tervikuna ja väljastas harva üksikud positiivsed väljundid. Katses teise isikuga, aga sama käega, millega koguti treeningandmed, tundis mudel löögi ära. Märkimisväärne on ka asjaolu et katseisik ei olnud kunagi piljardit mänginud ja seega oli löögitehnika tugevalt varieeruv ja ebastabiilne. Kuigi teise isiku löökide ära tundmisel ei olnud täpsus nii suur kui treeningandmeid kogunud mängija peal, kinnitas see siiski mudeli võimet tunda ära teiste mängijate sooritatud lööke.

Lisaks teise käega löökide sooritamisele ja teise isiku peal katsetamisele, testiti ka esialgse mängija poolt löögi sooritust muutmist. Näiteks liialdades pausi pikkust või tehes vaid ühe soojendusliigutuse. Ka sellisel moel löögi sooritust muutes klassifitseeris mudel löögi edukalt. Sooritatud katsete pealt saab järeldada, et mudelit saab rakendada teiste mängijate löökide ära tundmiseks ja seega ka teiste mängijate löökide kohta andmete kogumiseks.

Peatükis võrreldi mudeli treenimise iteratsioonide käigus valminud mudelite omadusi ja kirjeldati viimase mudeli katselise testimise tulemusi. Järgmine peatükk on töö kokkuvõte.

5. Kokkuvõte

Tööga lahendatav probleem on piljardi treenimise keerukus teatud tasemele jõudes. Lahendusena arendati edasi nutiseadmete kasutamist treeningute toetamiseks. Töö esialgseks eesmärgiks oli ehitada rakendus iOS ja WatchOS platvormile, mis tunneks ära sooritatud piljardilöögi kasutades kella liikumisandureid ja annaks löögi sooritusе kohta tagasisidet. Peatükis 3.4.4 kirjeldatud probleemi tõttu seati töö eesmärgiks luua masinõppe mudel ja rakenduse prototüüp, millega saaks koguda vaid löögi andmeid sisaldavaid positiive. Kohendatud eesmärk saavutati töö käigus. Valmis usaldusväärne mudel löögi andmete ära tundmiseks ning rakenduse prototüüp, mis kasutab seda mudelit löögi andmete salvestamiseks. Katsetega kinnitati, et valminud mudel on rakendatav ka teiste mängijate peal suuremal hulgal löögi andmete kogumiseks.

5.1 Potentsiaalsed edasiarendused

Ülesobitamise parandamiseks sama käega sooritatud löökide puhul oleks vaja koguda rohkemate mängijate sooritatud löökide andmed ja kaasata need mudeli treeningandmete hulka. Käelisuse suhtes ülesobitamise jaoks oleks vaja koguda andmeid teise käega sooritatud löökide kohta. Sõltuvalt kuidas mudeli täpsus muutub, kui treeningandmete hulgas on mõlema käega sooritatud lööke, on võimalik toimetada kahte moodi. Kui treeningandmetesse mõlema käe kaasamine ei kahjusta mudeli täpsust, saab treenida käelisusest sõltumatu mudeli. Kui aga mudeli täpsus langeb mõlema käega sooritatud löökide kaasamisel, on võimalik treenida kummagi käe löökide tuvastamiseks eraldi mudelid. Sellisel juhul valiks mängija rakenduse avamisel kumma käega ta lööke sooritab ja rakendus kasutaks käele vastavat mudelit.

Pika passiivse sisendi korral tagastatava valepositiivse väljundi probleemi saaks lahendada kogutud andmete automatiseeritud järeltöötlusega. See töötlus toetuks asjaolule, et löögi sooritusе sees on kii viimine löögikuulini ja kontakt kuuliga. Need sündmused väljenduvad kiirendusanduri y-telje andmetes suurte piikidena. Kogutud löögi andmeid saaks valideerida kontrollides, kas selles andmehulgas on olemas tugevad piigid.

Kasutatud allikad

[1] Jack H. Koehler. *The Science of Pocket Billiards*. *Second edition 1995*. Marinette: Sportology Publications. 1989.

[2] Kinam Kim, Yong K. Cho, Effective inertial sensor quantity and locations on a body for deep learning-based worker's motion recognition, *Automation in Construction*, 2020, Vol 113

https://www-sciencedirect-com.ezproxy.utlib.ut.ee/science/article/pii/S0926580519311185?via%3Dihub (09.12.2020)

[3] Cue Sport Lv. *Billiard Accessories*.

http://www.cuesport.lv/eng/index.php/Billiard_accessories (04.05.2021)

[4] Victor A. Carmona-Ortiz, Joan Lobo-Prat, Jeremy Van Ruysevelt, Carme Torras, Josep M. Font-Llagunes, Development and Pilot Evaluation of the ArmTracker: A WearableSystem to Monitor Arm Kinematics During Daily Life, *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob) Biomedical Robotics and Biomechatronics (BioRob), 2020 8th IEEE RAS/EMBS International Conference for*. *:759-764 Nov, 2020*, 760

https://ieeexplore-ieee-org.ezproxy.utlib.ut.ee/document/9224302 (09.12.2020)

[5] Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Et al, Motion Reconstruction Using Sparse Accelerometer Data., *ACM Transactions on Graphics*, Vol. 30, Article 18, 2011, 18:3

https://dl-acm-org.ezproxy.utlib.ut.ee/doi/10.1145/1966394.1966397 (09.12.2020)

[6] Chao Shen, Yufei Chen, Gengshan Yang, On motion-sensor behavior analysis for human-activity recognition via smartphones, *2016 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)*, 2016, pp. 1-6

https://ieeexplore-ieee-org.ezproxy.utlib.ut.ee/document/7477231 (09.12.2020)

[7] Jonathan Balaban, *How Turi Create is Disrupting the Machine Learning Landscape*

https://towardsdatascience.com/how-turi-create-is-disrupting-the-machine-learning-landscape-37b562f01eab (04.05.2021)

[8] Sagar Howal, *Activity Monitoring with Apple's Turi Create | Machine Learning*, 2018.

https://medium.com/@howal/activity-monitoring-with-apples-turi-create-machine-learning-1043ce5b9203 (17.03.2021)

[9] Apple inc. *Activity classifier how it works, Turi Create User Guide*

https://apple.github.io/turicreate/docs/userguide/activity_classifier/how-it-works.html (04.05.2021)

[10] Song-Mi Lee, Sang Min Yoon and Heeryon Cho, Human activity recognition from accelerometer data using Convolutional Neural Network, *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2017, pp. 131-134, doi: 10.1109/BIGCOMP.2017.7881728.

https://ieeexplore.ieee.org/document/7881728 (07.05.2021)

[11] Apple inc. *SpeedySloth: Creating a Workout.*

https://developer.apple.com/documentation/healthkit/workouts_and_activity_rings/speedysloth_creating_a_workout (03.01.2021)

[12] Apple inc. *Developer documentation, Getting raw accelerometer events.*

https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events (05.05.2021)

[13] Apple inc, *Developer documentation*, W*atch connectivity.*

https://developer.apple.com/documentation/watchconnectivity/ (15.02.2021)

[14] Koo Ping Shung, *Accuracy, Precision, Recall or F1?*

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (02.05.2021)

Lisad

I. Lähtekood

Töö raames loodud iOS ja WatchOS rakenduse lähtekood ning mudeli treenimiseks ja andmetöötluseks kasutatud Jupyter Notebook'i skriptid on kättesaadavad järgmistelt linkidelt:

https://github.com/kjlellep/shotMotion

https://github.com/kjlellep/jupNotebooks

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Karl-Jonathan Lellep,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose, **iOS rakendus piljardi treeningute jälgimiseks ja toetamiseks**, mille juhendaja on Artjom Lind, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Karl-Jonathan Lellep

07.05.2021