TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Infotehnoloogia kolledž

Märt Lõhmus 175011IDDR

MASINÕPPE MUDELITE REALISEERIMISE VÕIMALUSED ANDROID OPERATSIOONISÜSTEEMIL NÄGUDE HÄGUSTAMISE NÄITEL

Bakalaureusetöö

Juhendaja: Toomas Lepikult
PhD
Kaasjuhendaja: Silver Keskküla
MSc

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Märt Lõhmus

13.05.2020

Annotatsioon

Antud diplomitöö uurib ja analüüsib masinõppe mudelite rakendamise võimalusi Android operatsioonisüsteemil. Töö käigus leiab autor optimaalsema lahenduse masinõppe mudelite rakendamiseks Android operatsioonisüsteemil.

Autor annab ülevaate, mida kujutab endast masinõpe, mis on masinõppe treenimise kategooriad ja kuidas toimub selle rakendamine. Tutvustatakse masinõppe mudelite arhitektuure, mis sobivad antud probleemi lahendamiseks ning võrreldakse näopõhitunnuste ja inimese koordinaatide leidmist.

Kirjeldatakse riistvaralise kiirenduse võimalusi Android operatsioonisüsteemil ning analüüsitakse olemasolevaid masinõppe rakendamise tarkvarasid Androidile. Töö käigus väljaselgitatud parima masinõppe rakendamise tarkvaraga teostatakse prototüüplahendus, mis hägustab videos kaadri kaupa inimesed ja nende näod kasutades masinõppe mudeli poolt saadud inimeste koordinaate.

Autor kirjeldab leitud kitsaskohti ning toob välja edasised optimeerimise võimalused.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 5 peatükki, 13 joonist, 6 tabelit.

Abstract

Running machine learning models on Android OS through a practical example of face blurring

The goal of the thesis is to research and analyse possibilities of running machine learning models on Android operation system. The main challenge is to find the most optimal solution to inference machine learning models on Android operation system.

The author gives brief overview what machine learning is about, what are the categories of training machine learning models and what happens in machine learning model inference stage. Also, overview is given about machine learning model architectures which are relevant for finding optimal solution how to blur faces in videos in mobile phones. Author compares different approaches and lists pros and cons between finding face keypoints and finding entire person from single frame of video.

The author gives examples how machine learning is already being applied in Android operation system and explains how hardware accelaration is achieved by using different hardware calculation accelarations. Author describes in analysis section which machine learning frameworks exist currently and finds out the best one. The selected machine learning framework is used to conduct experiments on two different mobile phones and to find out how much time it takes to blur a person's face on video per frame.

Using the analysis and conducted experiment author explains further possibilities to optimise and improve the solution.

The thesis is in Estonian and contains 38 pages of text, 5 chapters, 13 figures, 6 tables.

Lühendite ja mõistete sõnastik

Android	Mobiili operatsioonisüsteem, mis baseerub kohandatud Linux kernelil ning vabavaralisel tarkvaral, loodud põhiliselt puutetundlikele mobiiliseadmetele
API	Rutiinide, protokollide ja tööriistade kogumik tarkvararakenduste arendamiseks
ATI	TTÜ Arvutitehnika instituut
Bitmap	Digitaalne pilt, mis koosneb punktide maatriksist
DPI	Dots per inch, punkti tolli kohta
iOS	Mobiili operatsioonisüsteem, mis on loodud ja kasutatav vaid Apple korporatsiooni riistvaral, näiteks iPhone ja iPod seadmetes
Kaadrisagedus	Kuvatavate kaadrite arv sekundis
Kaamera MP	Kaamera megapikslid
Klassifitseerimata andmed	Sisendandmed mudelile, millel ei ole olemas oodatavat tulemust
Loss funktsioon	Mudeli õigsuse arvutamise funktsioon
Mudeli kvantiseerimine	Analüütiline protsess, mille käigus hinnatakse mudeli väärtust
Neural Network API (NNAPI)	Arvutusintensiivsete operatsioonide API
Pilveteenus	Võrgupõhine ladustamise mudel, kus andmeid ladustatakse teenusepakkujate loogilistes basseinides
Raspberry Pi	Mikroarvuti, mis võimaldab kontrollida elektroonilisi komponente füüsiliseks arvutamiseks
RGB	Red green blue ehk punane roheline sinine värvimudel
Sildistatud andmed	Sisendandmed mudelile, millel on olemas oodatav tulemus
Tehisnärvivõrk	Lihtsustatud bioloogilise närvivõrgu mudel
Treeningandmed	Sisendandmed, mida kasutatakse mudeli treenimisel
Virtuaalne masin	Operatsioonisüsteemi jäljendamise tarkvara, mis emuleerib tõelist arvutit kõigi vajalike ressursside ja komponentidega
YOLO	You only look once ehk reaalaja objekti tuvastamise süsteem

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva diplomitöö eesmärgiks on välja selgitada kõige optimaalsem viis kuidas rakendada masinõppe mudeleid Android operatsioonisüsteemil. Töö käigus uuritakse võimalusi, võimekusi ja puudusi masinõppe mudelite rakendamisel Android operatsioonisüsteemil ning luuakse prototüüplahendused kõige sobivamatele lahendustele. Neid analüüsides selgitatakse välja sobivaim meetod kuidas jooksutada masinõppemudeleid mobiilis – näohägustamise näitel.

Teemavalikul arvestas autor käesoleva teema aktuaalsust tänapäeval ning selle tulevikku vaatavat potentsiaali. Autor kavatseb diplomitöös selgunud tulemusi kasutada firmas Supervaisor, et hägustada liikluses filmitud videotes inimeste näod. Antud hetkel on peamine viis telefonis tehtud videotes ja piltides olevate nägude hägustamiseks saata need pilveteenustesse, kus toimub nende anonümiseerimine inimeste nägude hägustamise näol. Antud lahendus ei ole optimaalne, kuna anonümiseerimata videod talletatakse pilveteenustes mingi perioodi vältel, mis avab teoreetilise võimaluse kolmandatel osapooltel neile ligi pääseda. Kasutaja seadmes nägude udutamine garanteerib, et tundlik informatsioon ei jõua kasutaja seadmest edasi ning lisaboonusena muudab mobiilide arvutusvõimsuse kasutamine masinõppe mudelite jooksutamise oluliselt soodsamaks.

1.1 Aktuaalsus

Tänapäeva maailmas on kasutajate privaatsuse tagamine äärmiselt aktuaalne ja oluline teema. Mobiilseadmed on saanud primaarseks interneti kasutamise vahendiks ning mobiiltelefone kasutatakse üha enam muudel eesmärkidel kui ainult helistamiseks. Uued võimalused tekitavad omakorda uusi takistusi, näiteks kuidas tagada kasutajate andmete privaatsus ja turvalisus.

Järjest enamatesse telefonidesse on sisse integreeritud spetsiaalselt masinõppe mudelite jooksutamiseks mõeldud masinõppe kiibid, mis muudavad mudelite jooksutamise efektiivseks ja telefoni akut ning muid ressursse säästvaks.

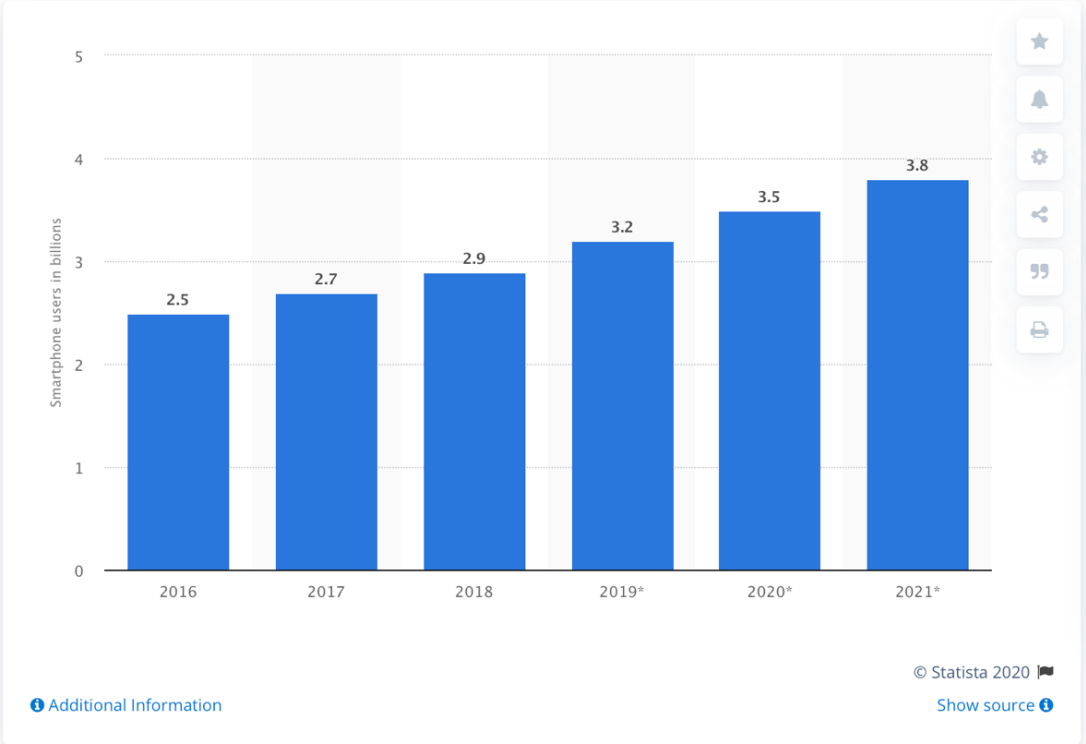
Masinõppe mudelite rakendamine mobiilis võimaldab suurendada kasutajate privaatsust, filtreerides välja ainult vajaliku informatsiooni. Näiteks rakendused, mis filmivad või pildistavad liiklust, saaksid ennem videote üleslaadimist udutada inimeste näod, mis kindlustaks, et nende tundlik informatsioon ei jõuaks vastavatest seadmetest edasi. Samuti vähendaks kasutajate seadmetes mudelite rakendamine oluliselt hägustamise hinda, kuna ei peaks rentima selleks virtuaalseid masinaid ning võimaldaks üles laadida vaid vajalikud osad.

2 Analüüs

Telefonide kaamerate kvaliteet ja võimalused on arenenud tundmatuseni alates aastast 2000, mil müüki tuli esimene kaameraga telefon Sharp J-SH04. Kui esimesel telefonil oli kaamera resolutsioon 0.11MP [1], siis näiteks uuema iPhone 11 pro resolutsioon on juba 12MP [2]. Kasvav telefonide arv maailmas ning paranev telefonikaamera muudab olukorda maailmas.

2.1 Tehniline lähteülesanne

Kaamerate järjest kasvava kvaliteedi tõttu kasutab suur osa inimesi just telefone pildistamiseks ning samuti kasutab kaamerat kasvav hulk mobiilis olevaid rakendusi. See toob endaga kaasa olukorra, kus kaadrisse võib jääda suur hulk inimesi, kelle kohta käiv privaatne informatsioon (näod, asukoht, videos olevad tegevused jne) laetakse üles pilve, kus nende näod võidakse hägustada või mitte. Internetti üleslaetud videoid kasutab ära enda tehisintellekti mudelite treenimiseks näiteks ettevõte Clearview AI, kes müüb erinevatele firmadele ja valitsusasutustele näotuvastusteenust [4]. Inimeste privaatsuse tagamiseks on plaanis uurida võimalusi inimeste nägude hägustamiseks juba telefonis nii, et tundlik informatsioon ei jõuaks kasutaja seadmest edasi.



Joonis 1. Mobiiltelefonide arvu kasv ja prognoos 2016-2021 [3].

Selle saavutamiseks on otstarbekas analüüsida olemasolevaid lahendusi ja koguda võimalikult palju informatsiooni parima lahenduse väljaselgitamiseks.

Süsteeminõuded:

- Nägude hägustamine peab toimuma täielikult mobiilis.
- Nägude hägustamine telefonis peab töötama ilma internetiühenduseta.
- Rakendus peab suutma video kaadrikaupa lahti kodeerida.
- Programm peab suutma video kaadrid bitmap formaati konverteerida ning need mudeli poolt nõutavasse suurusesse ja formaati teisendama.
- Inimeste nägude hägustamine toimub mudelirakendamisel saadud inimeste koordinaatide abil.

Autor valib analüüsi tulemusena olemasolevate lahenduste seast välja sobivaimad optimaalsed lahendused ning praktilises osas valmivad neist prototüüplahendused. Prototüüplahenduste soorituse võrdlemise käigus selgitatakse välja kõige parem lahendus masinõppe mudelite jooksutamiseks Android operatsioonisüsteemil.

2.2 Masinõpe

Masinõpe on tehisintellekti rakendamine viisil, mis võimaldab süsteemil õppida ja oma võimekust parandada, defineerimata konkreetseid käske. Masinõpe keskendub arvutisüsteemidele, mis suudavad etteantud andmeid kasutada enda võimekuse kasvatamiseks.

Masinõppe õppimise protsess algab sisestatud andmete töötlemisega, et aru saada neis olevatest mustritest ning nende põhjal tulevikus teha intelligentseid otsuseid. Masinõppe peamine eesmärk on teha otsuseid inimeste sekkumiseta ning parandada oma otsustamisprotsessi vastavalt leitud seostele [6].

2.2.1 Tehisintellekti lühiajalugu

Masinõppe pioneeriks võib pidada Arthur Samuel-i , kes 1952. aastal töötades IBM-is arendas välja esimese tehisintellektil põhineva kabe mängimise programmi. Antud programm suutis saavutada taseme, mis pakkus konkurentsi isegi maailmatasemel mängijatele [5].

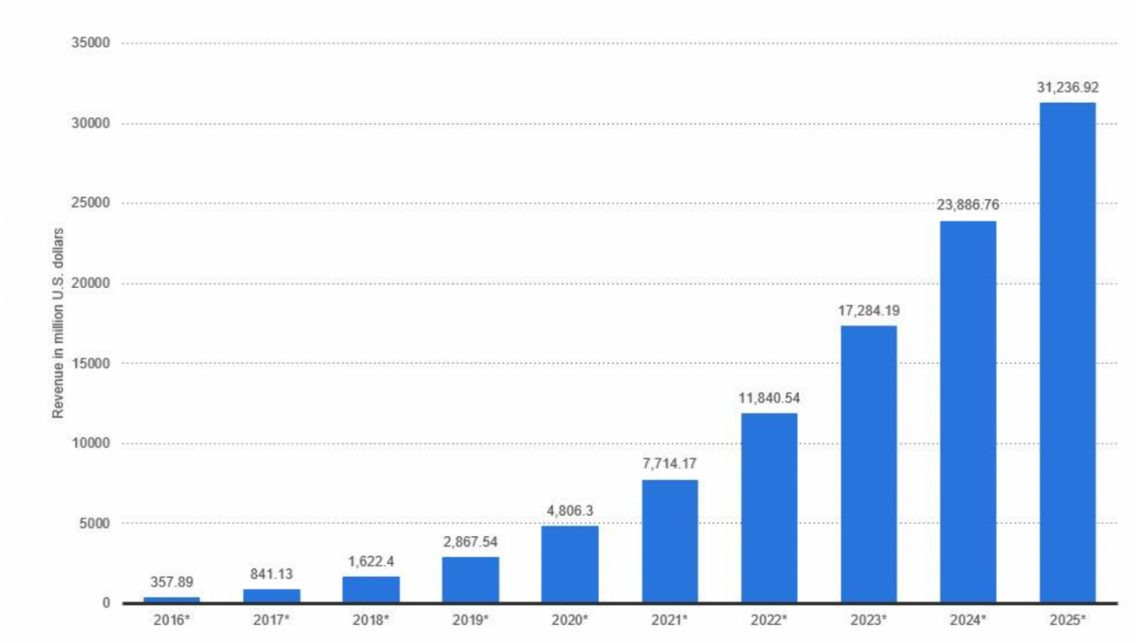
1965. aastal arendas Joseph Weizenbaum programmi ELIZA, mille eesmärk oli simuleerida psühholoogi rolli ning vastata temale esitatud küsimustele [5].

1981. aastal pärast pikka stagnatsiooni tehisintellekti maailmas lõi ettevõte Digital Equipment Corporation esimese kommertslikult eduka tehisintellekti süsteemi RI. RI aitas hallata arvuti süsteemide tellimusi [7].

1997. aastaks valmis IBM arvuti Deep Blue, mida treeniti male mängimiseks. Samal aastal mängis male suurkuju Garry Kasparov Deep Blue vastu ning esimest korda ajaloos suutis arvuti võita inimest males [7].

2008. aastal ilmus iPhone-il Google rakendus, mis sisaldas häältuvastust, mille veasagedus oli oluliselt väiksem kui varasematel sarnastel süsteemidel [7].

2012. aastal publitseeris Alex Krizhevsky artikli enda loodud sügavõppe tehnika Alexnet kohta. See suutis märkimisväärselt vähendada pildituvastuse veasagedust. Tema tehnikat kasutades on tehtud suuri edasiminekuid pildituvastuse osas [8].



Joonis 2. Tehisintellektil põhinevate teenuste käibe kasv aastast 2016 ja kasvu prognoos 2025 aastani [9].

Aastaks 2020 on tehisintellekt jõudnud peaaegu igasse eluvaldkonda ning kasv jätkub.

2.2.2 Masinõppe treenimine

Masinõppe mudeli treenimise eesmärk on mudelit õpetada mingit konkreetset ülesannet lahendama. Tüüpilisteks masinõppe poolt lahendatavateks probleemideks on näiteks objektide tuvastus piltidelt, näotuvastamine, objektide klassifitseerimine, kaardirakendustes lühema raja leidmine kahe või enama punkti vahel ning teksti ja hääle tuvastamine.

2.2.3 Masinõppe treenimise kategooriad

Masinõppe treenimist saab nende otstarve järgi kategoriseerida neljaks: juhendatud, juhendamata, pooljuhendatud ning stiimulõpe[10][6].

Juhendatud õppe korral sisestatakse algoritmile struktureeritud ja sildistatud andmed, mida kasutatakse mudeli treenimiseks ning mille põhjal oskab süsteem anda tulevikus õige väljundi. Näiteks süsteemile ette andes pilte, mis on sildistatud kui autod, õpib süsteem aru saama ja tulevikus ära tundma autosid ka sildistamata andmete korral [6][10].

Juhendamata õppe korral kasutatakse klassifitseerimata ning sildistamata treeningandmeid. Juhendamata õppimise puhul üritab masinõppe algoritm üles leida etteantud andmete seast peidetud seosed. Näiteks leiab mudel seosed andmete vahel, kui sisestada poeklientide andmed juhendamata algoritmidele [6][10].

Pooljuhendatud õppe korral kategoriseeritakse ning sildistatakse ainult peamised treeningandmete tunnused. Antud õpet kasutatakse, sest treeningandmete sildistamine ja kategoriseerimine on aeganõudev protsess ning tihti ei ole otstarbekas kogu andmestikku sildistada [6][10].

Stiimulõppe korral õpib masinõppe algoritm keskkonnaga suhtlemise kaudu maksimaalselt preemiat saama. Treeningandmetelt õppimise asemel keskendub stiimulõppe agent selliste tegevuste leidmisele, mille tulemusena saab ta maksimaalselt preemiat. Näiteks on see kasutuses AlphaGo-s – arvutiprogramm, mis mängib lauamängu Go õppides pidevalt oma tehtud vigadest ega korda neid, et maksimaalselt võita, mille tulemusena on programm võitnud isegi Go maailmameistrit [6][10].

Treenimine algab kvaliteetsete treeningandmete kogumise ja selle korrastamisega. Levinud praktika kohaselt jagatakse andmed kaheks: suurem osa mudeli treenimiseks ning väiksem osa mudeli õigsuse valideerimiseks.

Treeningandmeid analüüsides võib tekkida olukord, kus neid tuleb töödelda ennem kui neid saab kasutada sisendandmetena [14].

Kui sisendandmete hulgas leidub puudulikke andmeid, siis tuleb need eemaldada, sest masinõppe algoritmid ei oska neid tõlgendada. Selleks kasutatakse nende eemaldamist, täiendamist alternatiivsest allikast või täiendatakse algandmete keskmiste tulemustega [14].

Kategoriseeritavate andmete korral kasutatakse mudeli treeningu efektiivsuse tõstmiseks andmete numbrilisteks väärtusteks konverteerimiseks. Näiteks esitatakse värvid sõnede asemel RGB kujul numbritena [14].

Masinõppe mudelid töötavad palju efektiivsemalt ning paljude mudelite puhul on lausa nõutud, et mudeleid treenitakse andmetega, mis on skaleeritud ühele skaalale. Põhiliselt kasutatakse selleks kahte meetodit.

Normaliseerimise käigus skaleeritakse sisendandmete tunnused skaalale nullist üheni, mille saavutamiseks rakendatakse sisendandmete tunnustele min-max funktsiooni. Piltide puhul on iga piksel esindatud kolme täisarvuna vahemikus 0-255. Suured täisarvud muudavad treenimisprotsessi aeglasemaks ning masinõppe algoritmile keerulisemaks, mistõttu jagatakse iga pikslivääртus läbi suurima võimaliku väärtuse ehk 255-ga. [14][15].

Standardiseerimise käigus arvutatakse kõigepealt välja väärtuste keskmine väärtus ja hälve kogu treeningandmete põhjal. Seejärel rakendatakse neid treeningandmete peal, kus iga andme puhul lahutatakse keskmine ja jagatakse standardhälbega. Standardiseerimise käigus treeningpiltide ettevalmistamisel jagatakse see RGB kanaliteks. Iga RGB kanali puhul arvutatakse keskmine tulemus ja standardhälve ning rakendatakse iga treeningpildi peal [16].

2.2.4 Masinõppe mudeli rakendamine

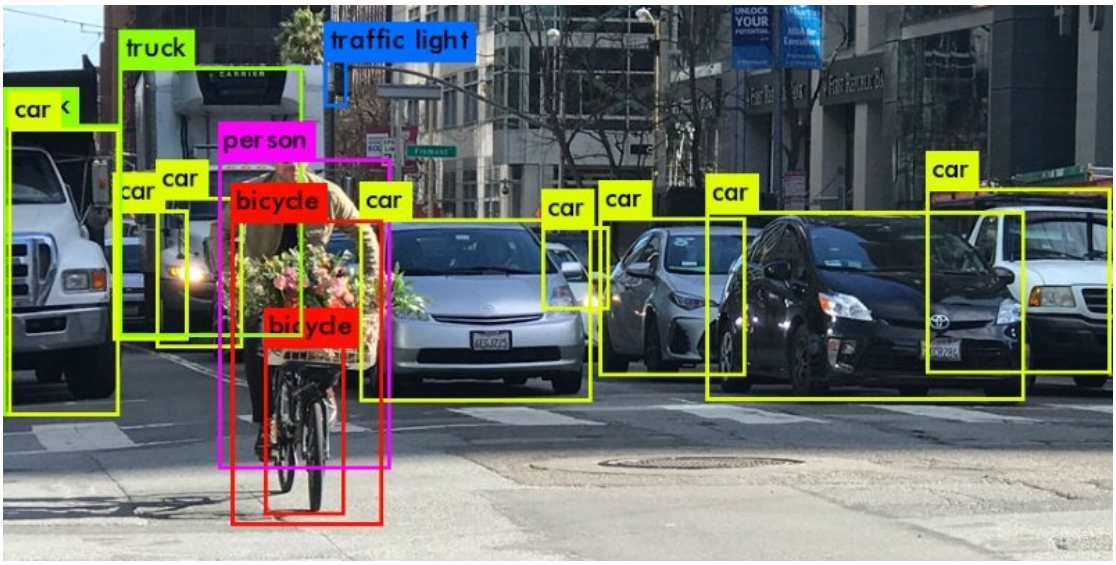
Masinõppe rakendamise faasis kasutatakse treenitud mudelit sisendandmetest soovitud tulemuste kättesaamiseks. Rakendamise faas sarnaneb treeningfaasile, kuid erinevalt treenimisfaasist ei toimu selle puhul kaalude uuendamist.

2.3 Näotuvastus ja hägustamine

Nägude hägustamiseks pildil tuleb kõigepealt üles leida neis olevate inimeste nägude koordinaadid. Nende ülesleidmiseks on mitmeid erinevaid lähenemisviise.

2.3.1 Objektide tuvastamine

Objektide tuvastamise algoritm tegeleb pildil olevate objektide klassifitseerimisega ning nende koordinaatide leidmisega pildil. Objektide klassifitseerimine üritab tuvastada pildil oleva objekti nimetuse, näiteks inimene, koer, valgusfoor ning selle koordinaadid pildil. Üks levinumaid erinevate objektide andmestikku ja objektide nimekirja haldav organisatsioon on COCO [17].



Joonis 3. Objektide tuvastamine [18].

Objektide tuvastamise algoritme saab kategoriseerida nende arhitektuuri järgi. Osa arhitektuure nagu R-CNN [18] eraldab objektide klassifitseerimise ning asukoha leidmise ülesanded ja arhitektuurid nagu YOLO, mis ühildab need ülesanded.

YOLO

YOLO on objektide tuvastamise ja klassifitseerimise arhitektuur, mis loodi Joseph Redmoni poolt aastal 2015. Antud arhitektuur hargneb omakorda mitmeks erinevaks tüübiks, vastavalt sellele, millises keskkonnas on mõeldud mudelit rakendada. YOLO klassifitseeritakse kui *single shot detector* ehk üritatakse tuvastada korraga mitut objekti selle asemel, et pilt erinevate objektide tuvastamiseks iga kord üle käia. Taoline lähenemine muudab YOLO kiireks ja vähem ressursse nõudvaks, kuid vähesel määral ebatäpsemaks väikeste objektide tuvastamisel. Hetkeseisuga on olemas kolm erinevat versiooni: yolov1, yolov2, yolov3 [27].

YOLO töötab põhimõttel, mis jagab sisendpildi väiksemateks ruudustikeks. Ruudustike arv oleneb YOLO mudeli tüübist. Ruudustikes on olenevalt mudelist teatud arv ankurkaste. Iga ankurkasti kohta leiab YOLO 6 väljundit:

- Objekti tõenäosus – näitab, kas ruudustikus leidus mõni objekt
- Objekti kasti koordinaati
- Nimekiri treenitud objektide esinemise tõenäosustest ruudus

Ennem, kui saab YOLO mudeli poolt väljastatud andmeid kasutada, tuleb rakendada neile meetodit *non max suppression* (NMS), sest YOLO võib leida samale objektile mitu erineva suuruse ja tõenäosusega ruudu koordinaate. NMS jätab alles kõige suurema tõenäosusega kastide jooned [27].

Joonis 4. Kõige suurema tõenäosusega kasti leidmine kasutades non max supression meetodit [25].

Faster R-CNN

Faster R-CNN on edasiarendus eelnevatest R-CNN-st ja Fast-RCNN-st, mis kasutasid selekteerivat otsimist [26], et välja selgitada millises pildi kohas on otsitavad objektid. Faster R-CNN puhul võtab konvolutsiooniline võrgustik sisendiks pildi ning tagastab pildil olevate objektide võimalikud asukohad. Järgmine konvolutsiooniline kiht kasutab võimalikke asukohti ja üritab tuvastada pildil olevad objektid. Viimane kiht selekteerib välja kõige suurema tõenäosusega objektide ning nende asukohtade ruudustikud [28].

Faster R-CNN puhul tuleb peale mudeli rakendamist siduda objektide nimed väljastatud tulemustega [28].

2.3.2 Näopõhitunnuste tuvastamine

Näopõhitunnuse leidmise algoritm tuvastab pildil olevate inimeste näo põhitunnusjoonte koordinaadid. Näopõhitunnuste koordinaatide leidmine on samuti esimene eeldus inimeste nägude äratundmisel. Erinevad algoritmid ja mudelid teevad seda erinevalt, kuid lihtsustatult leitakse järgnevad punktid inimese näos:

- Vasaku ja parema silma keskpunkt
- Vasaku ja parema silma sisemised ja välimised nurgad

- Vasaku ja parema kulmu keskkohad
- Vasaku ja parema kulmu välimised punktid
- Nina keskosa
- Suu keskosa
- Suu vasak ja parem välimine punkt

Joonis 5. Näo põhipunktide äratundmine [13].

Kuigi kvaliteetseid treeningandmeid inimeste nägude kohta on kogutud mitmeid, on nägude põhipunktide tuvastamine keerukas mitmete tingimuste pärast.

Pildil olevate inimeste nägude asend varieerub märkimisväärselt ning samuti mängib rolli välised tingimused nagu pildiheledus, näopositsioon (kas nägu on horisontaalselt või vertikaalselt), näosuurus kaadris, pildi resolutsioon, peegeldus, nahavärv, poolik nägu kaadris [11].

2.4 Masinõpe mobiilis

Mobiilide võimekuse kasv on loonud vajalikud riistvaralised ja tarkvaralised eeldused masinõppe rakendamiseks mobiiltelefonides. Mobiilide mälumaht on suurenenud mitme giga-ni, juurde on tulnud spetsiaalselt mobiilidele mõeldud võimsad mitmetuumalised protsessorid, mis võtavad vähe aku jõudlust ning masinõppe mudelite väljatöötamisel

mõeldakse üha enam mobiilide peale. Kuid mobiilide võimekuse kasvust hoolimata pole masinõppe neis triviaalne tegevus [21].

2.4.1 Masinõppe kasutusjuhud

Masinõpet on mobiilitootjad kasutanud juba mitmeid aastaid erinevate ülesannete täitmiseks.

Google on juba aastaid arendanud ning täiustanud oma häältuvastust, mis kasutab masinõpet, et aru saada, mida öeldi ning mis on küsimuse keeleline kontekst. Antud lahendust on integreeritud kõikvõimalike Androidi teenustega nagu kalender, helistamine, äratuskella valimine [22].

Telefonide kaamerate kvaliteet on läinud aastatega järjest paremaks. Progress on toimunud ühelt poolt riistvara pealt, kuid pildikvaliteeti on märkimisväärselt parandatud samuti masinõppe abil. Google Pixel 4 telefon kasutab masinõpet, mis rakendab telefoni mitme kaamera pildid üheks kombineerides neilt kõige kvaliteetsemad pikslid [23].

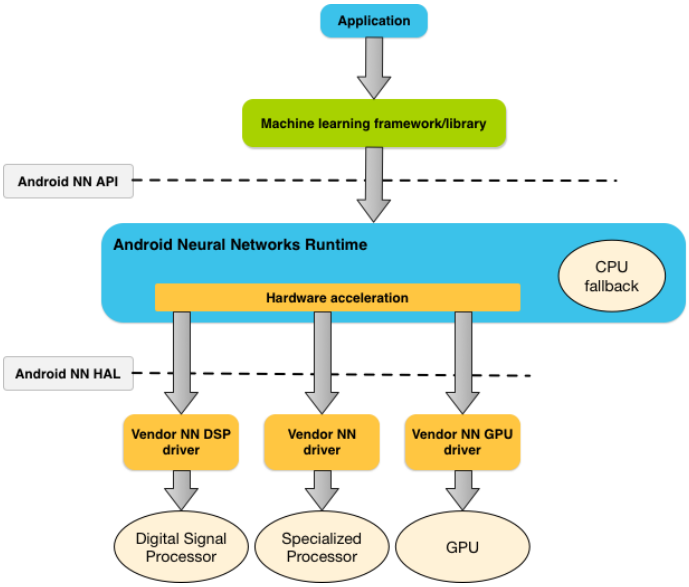
Masinõpet kasutatakse telefonides turvalisuse täiustamiseks. Paroolid on muutumas üha suuremaks ohuks inimeste andmetele, seda nii suurenenud arvutusvõimsuse tõttu, mis muudab paroolide lahtimuukimise lihtsamaks, kui ka ebaturvaliste paroolide ning nende korduvkasutamise tõttu erinevates rakendustes [24]. Masinõppe abil kasutatakse üha enam biomeetrilisi autentimise meetodeid. Masinõppe algoritmide abil on paljudesse telefonidesse jõudnud nii sõrmejälje kui ka näotuvastuse abil autentimine, mis muudab antud protsessi turvalisemaks [24].

2.4.2 Androidi riistvaraline tugi

Android operatsioonsüsteemi kasutavad paljud erinevad mobiilitootjad, kes kasutavad erinevaid riistvaralisi komponente. See tekitab olukorra, kus erinevate tootjate mikrokiipide võimekuse ära kasutamiseks on vaja kasutada nendele spetsiifilist tarkvara liidestust [21].

Android operatsioonisüsteemis on kaasatud alates versioonist 8.1 liidestus *Neural Network API*, mis on mõeldud arvutuslikult keeruliste operatsioonide rakendamiseks Android telefonidel. NNAPI on madala taseme liidestus, mille peale on ehitatud mitmed erinevad raamistikud nagu Tensorflow Lite ja Pytorch Mobile [20] [21].

NNAPI peamine eesmärk on aidata efektiivselt ära kasutada mobiilseadme riistvaralist võimekust ning jagada arvutuslik jõudlus olemasolevate riistvara komponentide vahel. NNAPI suudab otseselt suhelda ning jõudlust jagada spetsiaalsete masinõppe rakendamiseks mõeldud protsessoritega, graafiliste protsessoritega (GPU) , signaalprotsessoriga (DSP) ning eelnevate puudumisel keskprotsessoriga (CPU) [20][21].



Joonis 6. NNAPI süsteemine arhitektuur [20].

2.5 Olemasolevad lahendused

Android platvormi jaoks on väljaarendatud palju erinevad raamistikke, mis võimaldavad rakendada masinõpet mobiilis. Enamuses on need kas hobiprojekti tasandil või olemasolevate lahenduste peale ehitatud raamistikud, mis lihtsustavad kasutamist, kuid piiravad kasutamisviise. Kaks kõige populaarsemat ning paljudes rakenduses kasutatavat on Google poolt väljaarendatud Tensorflow Lite (endise nimega Tensorflow Mobile) ning Facebook poolt loodud Pytorch Mobile.

2.5.1 Tensorflow lite

Tensorflow Lite loodi aastal 2017, et rakendada masinõpet piiratud ressurssiga seadmetel, mille hulka kuuluvad Android ja iOS operatsioonisüsteemil põhinevad mobiilid, Raspberry Pi ning mikrokontrollerid. Tensorflow Lite koosneb paljudest erinevatest liidestustest, mille abil konverteerida ning rakendada mudeleid seadmetel.

Tensorflow Lite võimaldab kasutada enda treenitud mudeleid, kuid pakub eeltreenitud ning konverteeritud mudeleid järgnevatele levinud probleemidele: pildi klassifikatsioon, objekti tuvastamine, inimese kehaasendi tuvastamine, kõnetuvastus, pildi segmentatsioon, tekstiklassifikatsioon.

Tensorflow Lite liidestus võimaldab kasutada seda erinevates keeltes: Java, Swift, Objective-c, C++ ja Python.

Tensorflow Lite võimaldab riistvaralist kiirendust kasutades ära Android Neural Network API-t.

Tensorflow Lite liidestik võimaldab mudelit optimeerida mobiili jaoks. Tensorflow Lite võimaldab mudeleid kvantiseerida, mis kiirendab rakendamise protsessi ning vähendab mudeli suurust, kuid vähendab väiksel määral täpsust.

Tensorflow Lite koosneb peamiselt kahest suuremast osast: Tensorflow Lite interpretaator ja Tensorflow Lite konverter.

Tensorflow Lite konverteri abil toimub mudeli konverteerimine sobivaks .tflite formaadiks. Konverter võtab sisendiks Tensorflow formaadis olevad mudelid.

Tensorflow Lite interpretaatori abil toimub mudeli rakendamine. Kõigepealt luuakse interpretaatori isend. Seejärel tuleb interpretaatori laadida .tflite formaadis mudel. Vastavalt mudeli eripärale tuleb ette valmistada sisendandmed, mis pildi puhul tähendab pildi suuruse skaleerimist mudelile sobivaks suuruseks. Ettevalmistatud andmed tuleb konverteerida interpretaatorile sobivaks *flatbuffermodel* formaati. Kui mudel on laetud ning andmed õigesse formaati konverteeritud, siis toimub mudeli rakendamine, mille tulemusel tekib väljund.

2.5.2 Pytorch mobile

Pytorch Mobile on osa suuremast Pytorch-i vabavaralisest masinõppe raamistikust ning see avaldati 2019. aasta detsembris. Pytorch Mobile lihtsustab Pytorch formaadis mudelite jooksutamist nii Androidi kui ka iOS-i operatsioonisüsteemidel. Pytorch Mobile on hetkeseisuga veel katsetuse staadiumis [29].

Pytorch Mobile raamistikus võimaldab programmeerida Java ning c++ keeltes [29].

Pytorch Mobile liidestik võimaldab mudelit optimeerida mobiili jaoks. Pytorch Mobile võimaldab mudeleid kvantiseerida [29].

Pytorch Mobile hetkeversioon ei võimalda ära kasutada riistvaralist kiirendust, mistõttu mudel rakendatakse kasutades telefoni keskprotsessorit [29].

Pytorchi Mobile mudeli konvertaator võimaldab Pytorchi mudeleid ilma muudatusteta konverteerida sobivaks .pt formaadiks [29].

Pytorch mobile mudeli rakendamine toimib sarnaselt Tensorflow Lite omale. Kõige suurem erinevus on see, et Pytorch Mobile võtab sisendiks raster kujul pildi.

3 Tehniline teostus

Diplomitöö eesmärgiks on leida kõige optimaalsem viis, kuidas masinõppe mudeleid rakendada mobiilis ning uuringu tulemusi kasutada, et integreerida vastavad mudelid reaalses rakenduses. Masinõppe mudelite rakendamise raamistikke mobiili jaoks on mitmeid, kuid enamuse saab välja filtreerida, kuna neid ei arendata edasi. Sõelale jäävad Pytorch Mobile ja Tensorflow Lite, kuid tehnilises osas teostatakse rakendus Tensorflow Lite-s mitmel erineval põhjusel [35].

3.1 Masinõppe raamistiku valik

Pytorch Mobile suurim miinus on see, et see on veel katsetusfaasis. Pytorch mobile tuli välja eksperimentaalfaasis Pytorch versiooniga (1.3) 10.10.2019 [34] ning stabiilset versiooni ei ole veel välja tulnud. Ilma stabiilse versioonita ei saa seda kasutada reaalsetes rakenduses, kuna sellel põhinev rakendus võib tuua rakendust lõhkuvaid muudatusi ja sellega kaasnevat riski kasutajate turvalisusele. Tensorflow Lite-st on väljas mitmeid stabiilseid versioone [35].

Riistvaraline kiirendus saavutatakse kasutades telefonis olevaid GPU ja teisi spetsiaalselt arvutuste kiirendamiseks mõeldud protsessoreid. Arvutuslik koormus jagatakse erinevate protsessorite vahel, mis tõstab märkimisväärselt rakendamise kiirust, vähendab aku kasutust ning vähendab seetõttu telefoni kuumenemist. Pytorch Mobile ei toeta riistvaralist kiirendust, mis vähendab märkimisväärselt selle kasutusvõimalikkust [36]. Pytorch Mobile raamistik suudab hetke versiooniga rakendada mudeleid ainult kasutades mobiilide CPU-sid. Tensorflow Lite raamistik suudab ära kasutada enamustel seadmetel riistvaralist kiirendust [35].

Tensorflow kasuks räägib samuti laiaulatuslik kasutajaskond, mistõttu leidub hulganisti realiseeritud projekte, mis kasutavad Tensorflow-d ning ka õpivideoid -ja juhendeid [35].

3.2 Masinõppe mudeli valik

Masinõppe mudeliks sai valitud YOLO arhitektuuril põhinev mudel. Antud arhitektuuri mudelid on mõeldud töötama reaalajas ning samuti on sellel spetsiaalselt madalate ressurssidega keskkonna jaoks loodud mudeli arhitektuur YOLO tiny [30]. YOLO

muudab kiireks ja vähem ressurssi nõudvaks selle arhitektuur, mis rakendab ühte tehisnärvivõrku pildile selle asemel, et rakendada erinevateks ülesanneteks mitmeid võrke, mis oleks ressurssi nõudvam ja aeglasem.

Masinõppe mudeli valiku puhul on arvesse võetud mitmeid tegureid. Töö peamine eesmärk on mudeli rakendamine mitte mudeli treenimine, seega üheks mudeli valimise kriteeriumiks on mudeli rakendamise lihtsus ning eeltreenitud mudeli olemasolu. YOLO võimaldab mudeleid nullist treenida oma andmestiku peal, kuid olemas on ka Open Image andmestikul [31] treenitud mudelid [30]. Open Image on põhjalik andmestik, mis koosneb 59 919 574 treeningpildist. Eeltreenitud mudel on treenitud ära tundma 80 klassi, millest antud töös kasutatakse inimese äratundmise klassi.

Lihtsustamiseks valiti objekti tuvastamise mudel, mis leiab pildilt üles terve inimese ruudu koordinaadid ning hägustab terve pildil oleva inimese. Objektide tuvastuseelis näo põhipunktide leidmise juures on see, et näotuvastuse puhul peab pildil olema selgelt nähtavad näo põhipunktid, mis tihtipeale ei ole masinõppe mudelile selgelt eristatavad.

3.3 Video kaadriteks tegemine

Kuna masinõppe mudel võtab sisendiks pildi, tuleb video kaadrite kaupa piltideks teha. Alates versioonist 10 on Androidile sisse ehitatud rakendusliides *mediametadataretriever*, mis pakub liidestust meedia failidest kaadrite ja *metadata* kättesaamiseks [33]. Kasutades *mediadataretrieverit* implementeeriti funktsioon, millega saada vastavalt video kaadrisagedusele kätte kaadrid ja konverteerida bitmap formaati.

```
Fun getFramesFromVideo(videoPath: String, videoLength: Int, frameRate: Int) {
   //create new MediaMetadataRetriever instance
   val retriever = MediaMetadataRetriever()
   //set data source for video
   retriever.setDataSource(videoPath)

   // get total amount of frames in video
   val framesInVideo = videoLength * frameRate

   // loop over all frames and retrieve frame by frame
   for (i in 0 until framesInVideo ) {

      // multiply with 1000 to convert microseconds to milliseconds
      val bitmap = retriever.getFrameAtTime((i * 1000).toLong())

      // use bitmap as input for machine learning model

   }
}
```

Joonis 7. Koodinäidis videost kaadrite kättesaamisest (M. Lõhmus, 2020).

3.3.1 Mudeli konverteerimine

Tensorflow Lite raamistik nõuab, et enne mudeli rakendamist tuleb mudel konverteerida kõigepealt tflite formaati. Tensorflow Lite konvertaatori abil saab ainult konverteerida mudeleid, mis on tf.keras [38] või saved models Tensorflow formaadis (.pb) [37].

YOLO tiny mudeli konverteerimiseks .pb formaati kasutati teeki tensorflow-lite-YOLOv3 [39]. Pb formaadi konverteerimiseks tuleb kõigepealt alla laadida YOLO kaalud ja mudeli poolt otsitavate objektide nimekiri.

YOLO kaalusid ja mudeli otsitavate objektide nimekirja tuleb kasutada järgnevas käsus, mis genereerib valmis .pb formaadis Tensorflow Lite mudeli:

```
python ./convert_weights_pb.py.
```

Seejärel tuleb .pb formaadis mudelit kasutada Tensorflow Lite konvertaatori järgnevas käsus: tflite_convert –saved_model_dir/yolov3.pb/ --output_file yolo_v3.tflite.

3.3.2 Masinõppe rakendamine

Masinõppe mudelite rakendamisel mobiiltelefonidel kasutati olemasolevaid telefone Google Pixel 2 (Android 10) [40] ja Huawei P10 lite (Android 8) [41]. Hägustamiseks valiti liikluses filmitud 10 sekundilise video kaadrisagedusega 30 kaadrit sekundis. Tulemused on kõigi kaadrite keskmised väärtused.

3.3.3 Google Pixel 2

Tabel 1. Mudeli rakendamine resolutsiooniga 2688 * 1520.

	CPU	GPU	NNAPI
Kaadri kättesaamine	830 ms	831 ms	830 ms
Mudeli rakendamine	309 ms	136 ms	115 ms
Mudeli tulemuste töötlemine	5 ms	4 ms	5 ms

Joonis 8. Mudeli rakendamine resolutsiooniga 2688 * 1520 (M. Lõhmus, 2020).

Tabel 2. Mudeli rakendamine resolutsiooniga 1920 * 1080.

	CPU	GPU	NNAPI
Kaadri kättesaamine	509 ms	508 ms	509 ms
Mudeli rakendamine	305 ms	120 ms	110 ms
Mudeli tulemuste töötlemine	5 ms	4 ms	5 ms

Joonis 9. Mudeli rakendamine resolutsiooniga 1920 * 1080 (M. Lõhmus, 2020).

Tabel 3. Mudeli rakendamine resolutsiooniga 1024 * 576.

	CPU	GPU	NNAPI
Kaadri kättesaamine	316 ms	317 ms	316 ms
Mudeli rakendamine	310 ms	125 ms	115 ms
Mudeli tulemuste töötlemine	5ms	6ms	6 ms

Joonis 10. Mudeli rakendamine resolutsiooniga 1024 * 576 (M. Lõhmus, 2020).

3.3.4 Huawei P10 Lite

Tabel 4. Mudeli rakendamine resolutsiooniga 2688 * 1520.

	CPU	GPU	NNAPI
Kaadri kättesaamine	693 ms	puudub	puudub
Mudeli rakendamine	566 ms	puudub	puudub
Mudeli tulemuste töötlemine	15 ms	puudub	puudub

Joonis 11. Mudeli rakendamine resolutsiooniga 2688 * 1520 (M. Lõhmus, 2020).

Tabel 5. Mudeli rakendamine resolutsiooniga 1920 * 1080.

	CPU	GPU	NNAPI
Kaadri kättesaamine	534 ms	puudub	puudub
Mudeli rakendamine	562 ms	puudub	puudub
Mudeli tulemuste töötlemine	15 ms	puudub	puudub

Joonis 12. Mudeli rakendamine resolutsiooniga 1920 * 1080 (M. Lõhmus, 2020).

Tabel 6. Mudeli rakendamine resolutsiooniga 1024 * 576.

	CPU	GPU	NNAPI
Kaadri kättesaamine	338 ms	puudub	puudub
Mudeli rakendamine	560 ms	puudub	puudub
Mudeli tulemuste töötlemine	15 ms	puudub	puudub

Joonis 13. Mudel rakendamine resolutsiooniga 1024 * 576 (M. Lõhmus, 2020).

3.4 Töö rakendamise tulemus

Töös läbi viidud katsed näitasid, et mudelite rakendamine inimeste nägude hägustamiseks videotes on võimalik. Videote resolutsioon ei oma tähtsust mudeli rakendamise puhul, kuna sisendpildid skaleeritakse mudeli poolt nõutud 320 pikslit korda 320 pikslit suurusele. Oluline on aga pikslite suurus pildilt objektide ülesleidmisel – mida suurem on algne resolutsioon, seda suurem võimalus on eristada pildil ja üles leida objekte. Resolutsioon mõjutas videotest kaadrite kättesaamist. Kõige suurema ja väiksema resolutsiooni korral erinesid tulemused Google Pixel 2 puhul ligikaudu 2.4 korda ning Huawei P10 Lite puhul ligikaudu 2 korda. Kaadrite kättesaamine on üks kitsaskohtadest, mille optimeerimisel paraneks üldine rakenduse sooritus.

Läbi viidud katsed näitasid, et mobiilide riistvaralise võimekuse tõttu erineb mudeli realiseerimise aeg telefonides mitmeid kordi. Katse näitas, et kui kasutada riistvara kiirenduseks GPU-d või NNAPI, mis suudab jagada arvutuslikku koormust, väheneb rakendamiseks kuluv aeg märkimisväärselt. Google Pixel 2 puhul erines mudeli rakendamise aeg CPU ja GPU vahel ligikaudu 3 korda.

Mudeli tulemuste töötlemise aeg ei muutunud oluliselt erinevates katsetes ja oli triviaalselt väike.

3.5 Edasine optimeerimine

Masinõppe mudeli rakendamisel tuli ette mitmeid kitsaskohti, mida oleks hea parandada ennem reaalsete kasutajateni jõudmist. Üks aeganõudvamaid tegevusi on videotest

kaadrite kättesaamine, mis olemasoleva Android teegi *mediadataretrieveri* abil võtab liigselt aega. Töö käigus prooviti ka mõnda teist suuremat teeki, kuid nende tulemus oli veelgi aeglasem kui *mediadataretrieveril.*

Masinõppe mudelite rakendamise kiiruse puhul mängib samuti rolli seadme enda jõudlus ning edasise arenduse käigus on võimalik optimeerida rakendamise aega, jagades rakenduse jõudluse erinevate protsessorite tuumade vahel.

Masinõppe mudelite valik mõjutab oluliselt rakendamise aega. Seda saab vähendada katsetades erinevate mudelite arhitektuure.

Põhjalikum uurimus tuleks läbi viia mudelite rakendamise mõjust seadme akule ning mõõta temperatuuri tõusu.

Lisaks võiks masinõppe mudelite realiseerimise rakendus kustutada pärast edukat nägude hägustamist originaalse video minimeerimaks selle lekkimise riski.

4 Kokkuvõte

Käesoleva diplomitöö eesmärk oli uurida masinõppe mudelite rakendamise võimalusi Android operatsioonisüsteemiga mobiilitelefonidel inimeste privaatsuse kaitsmiseks, hägustades inimeste näod ennem, kui videod laetakse telefonidest internetti. Autor andis ülevaate, miks on pilveteenustes nägude hägustamine vähem privaatsem kui nägude hägustamine mobiilitelefonides.

Analüüsi peatükis andis autor ülevaate, mida kujutab endast masinõpe, selle mudeli treenimine ning erinevad treenimismeetodid. Püstitati nõuded valmivale prototüüplahendusele. Analüüsi käigus selgus veel, et masinõppe mudelite rakendamise raamistikke, mis on piisavalt küpsed reaalsetes rakendustes kasutamiseks, leidub ainult üksikuid. Autor analüüsis neid raamistikke andes ülevaate nende omadustest.

Teostuse osas, põhinedes tehtud analüüsile, valis autor välja raamistiku, mille põhjal teha prototüüplahendused. Selgus, et optimaalseid lahendusi, mis on sobilikud samuti reaalsetes rakenduses kasutamiseks, leidub hetkel vaid üks: Tensorflow Lite. Mudeli rakendamiseks kasutati YOLO arhitektuuril põhinevat objektide tuvastamise mudelit. Mudel oli eeltreenitud Open Image andmestikul.

Teostuse osas viidi läbi katsed erinevate resolutsioonidega 10 sekundiliste videote peal, kasutades kahte autoril olemas olnud mobiiltelefoni, mille jõudlused erinesid märkimisväärselt. Mõlemal telefonil viidi katsed läbi samuti kasutades kolme erinevat riistvaralist kiirendust – CPU, GPA ja NNAPI. Teostuse osas selgus, et üks suuremaid kitsaskohti oli videost kaadrite kättesaamine, mille aega oleks hea vähendada, et nägusid hägustada reaalsetes rakendustes. Läbiviidud katsetes selgus samuti, et mobiilide riisvaraline võimekuse tõttu erineb mudeli realiseerimise aeg telefonides mitmekordselt.

Diplomitöö käigus tehtud analüüs ja realiseeritud prototüüp annab tugipunkti, et aru saada, millised võimalused on Android operatsioonisüsteemil mudelite rakendamiseks. Realiseeritud prototüübi rakendamise käigus sai autor ülevaate, millised on lahenduse kitsaskohad ning millele tuleks tähelepanu pöörata, et vähendada video hägustamisele kuluvat aega.

Autor kavatseb võtta diplomitöös saadud prototüübi ja tehtud analüüsi, et rakendada nägude hägustamist reaalses Supervaisor-i rakenduses.

Kasutatud kirjandus

[1] Uswitch, History of mobile phones and the first mobile phone, 11.03.2020 [https://www.uswitch.com/mobiles/guides/history-of-mobile-phones/]

[2] Apple, iPhone 11 spetsifikatsioon, 11.03.2020 [https://www.apple.com/iphone-11-pro/specs/]

[3] Statista, Number of smartphone users worldwide from 2016 to 2021, 11.03.2020 [https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/]

[4] Business Insider, Clearview AI scraped billions of photos from social media to build a facial recognition app that can ID anyone, 11.03.2020 [https://www.businessinsider.com/what-is-clearview-ai-controversial-facial-recognition-startup-2020-3]

[5] Mlplatform, What is Machine Learning, 13.03.2020 [https://mlplatform.nl/what-is-machine-learning/]

[6] Expertsystem, What is Machine Learning? A definition, 19.03.2020 [https://expertsystem.com/machine-learning-definition/]

[7] BBC, AI: 15 key moments in the story of artificial intelligence, 19.03.2020 [https://www.bbc.co.uk/teach/ai-15-key-moments-in-the-story-of-artificial-intelligence/zh77cqt]

[8] Arstechnica, How computers got shockingly good at recognizing images, 19.03.2020 [https://arstechnica.com/science/2018/12/how-computers-got-shockingly-good-at-recognizing-images/]

[9] Statista, Business organizations' reasons for adopting artificial intelligence (AI) worldwide, as of 2017, 19.03.2020 [https://www.statista.com/statistics/747775/worldwide-reasons-for-adopting-ai/]

[10] Towardsdatascience, Types of Machine Learning Algorithms You Should Know, 20.03.2020 [https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861]

[11] Stanford, Facial keypoints detection using Neural Network, 24.03.2020 [http://cs231n.stanford.edu/reports/2016/pdfs/007_Report.pdf]

[12] Intechopen, Automated Face Recognition: Challenges and Solutions, 24.03.2020 [https://www.intechopen.com/books/pattern-recognition-analysis-and-applications/automated-face-recognition-challenges-and-solutions]

[13] Epic-face3, Facial Keypoint Detector, 25.03.2020 [https://epic-faces3.herokuapp.com/keypoints/]

[14] Towardsdatascience, How To Develop a Machine Learning Model From Scratch, 01.04.2020 [https://towardsdatascience.com/machine-learning-general-process-8f1b510bd8af]

[15] Machinelearningmastery, How to Manually Scale Image Pixel Data for Deep Learning, 01.04.2020 [https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/]

[16] Analyticsvidhya, Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization, 04.04.2020 [https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/]

[17] Cocodataset, Common Object in Context, 06.04.2020 [http://cocodataset.org/#home]

[18] Azati, Image detection, recognition, and classification with machine learning, 06.04.2020 [https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/]

[19] Arxiv, Rich feature hierarchies for accurate object detection and semantic segmentation, 08.04.2020 [https://arxiv.org/abs/1311.2524v5]

[20] Developer.android.com, Neural Networks API, 20.04.2020 [https://developer.android.com/ndk/guides/neuralnetworks]

[21] Arxiv, AI Benchmark: Running Deep Neural Networks on Android Smartphones, 20.04.2020 [https://arxiv.org/pdf/1810.01109.pdf]

[22] Assistant.google.cim, Google Assistant, 21.04.2020 [https://assistant.google.com]

[23] Googleblog, Improvements to Portrait Mode on the Google Pixel 4 and Pixel 4 XL, 21.04.2020 [https://ai.googleblog.com/2019/12/improvements-to-portrait-mode-on-google.html]

[24] Securityweek, The Kiss of Death for Passwords: Machine Learning?, 21.04.2020 [https://www.securityweek.com/kiss-death-passwords-machine-learning]

[25] Medium, YOLO v3 theory explained, 21.04.2020 [https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193]

[26] Arxiv, Rich feature hierarchies for accurate object detection and semantic segmentation, 21.04.2020 [https://arxiv.org/pdf/1311.2524.pdf]

[27] Arxiv, YOLOv3: An Incremental Improvement, 21.04.2020 [https://arxiv.org/pdf/1804.02767.pdf]

[28] Towersdatascience, R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms, 21.04.2020 [https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e]

[29] Pytorch, PyTorch Mobile dokumentatsioon, 21.04.2020 [https://pytorch.org/mobile/home/]

[30] Pjreddie, YOLO: Real-Time Object Detection, 24.04.2020 [https://pjreddie.com/darknet/yolo/]

[31] OpenImage, Open Images Dataset dokumentatsioon, 24.04.2020 [https://storage.googleapis.com/openimages/web/index.html]

[32] Github, FFmpegMediaMetadataRetriever teek, 25.04.2020 [https://github.com/wseemann/FFmpegMediaMetadataRetriever]

[33] Developer.android, MediaMetadataRetriever dokumentatsioon, 25.04.2020 [https://developer.android.com/reference/android/media/MediaMetadataRetriever]

[34] Ai.facebook.com, Pytorch Mobile versiooni 1.3 väljastamise dokumentasioon , 25.04.2020 Pytorch Mobile [https://ai.facebook.com/blog/pytorch-13-adds-mobile-privacy-quantization-and-named-tensors/]

[35] Github.com, Tensorflow Lite väljastamise dokumentatsioon, 25.04.2020 [https://github.com/tensorflow/tensorflow/blob/master/RELEASE.md]

[36] Pytorch.org, Pytorch Mobile mobiili jaoks mudeli optimeerimine, 25.04.2020 [https://pytorch.org/blog/pytorch-1-dot-3-adds-mobile-privacy-quantization-and-named-tensors/]

[37] Tensorflow.org, Tensorflow Lite dokumentatsioon, 25.04.2020 [https://www.tensorflow.org/lite/guide/get_started]

[38] Tensorflow.org, Tensorflow Lite keras mudelite kasutamise dokumentatsioon, 25.04.2020 [https://www.tensorflow.org/guide/keras/overview]

[39] Github.com, Yolov3 mudeli konverteerimise teek, 25.04.2020 [https://github.com/peace195/tensorflow-lite-YOLOv3]

[40] Android.com, Google Pixel 2 spetsifikatsioon, 25.04.2020 [https://www.android.com/intl/en_uk/phones/google-pixel-2/]

[41] Huawei.com, Huawei P10 Lite spetsifikatsioon, 25.04.2020 [https://consumer.huawei.com/en/support/phones/p10-lite]