TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taavi Martoja 155528

# NÄOTUVASTUS VIDEOTEST: TEHNILINE LAHENDUS JA EKSPERIMENDID

Bakalaureusetöö

Juhendaja: Innar Liiv
Doktorikraad

Tallinn 2019

# 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: Taavi Martoja

20.05.2019

# Annotatsioon

Tehisintellekt on tänapäeval üks kiiremini arenevaid teadusharusid. Näotuvastus on vaid üks näide valdkonnast, milles on tehtud suuri läbilööke just tänu tehisintellekti kasutusele võtule.

Töö põhieesmärgiks on luua näotuvastuse tehniline lahendus videopildist inimeste tuvastamiseks. Töö teised eesmärgid on seotud näotuvastuseks kasutatava närvivõrgu täpsuse parandamisega. Eksperimentidel kasutatakse närvivõrgu treenimisel erinev arv pilte, et leida optimaalne piltide arv mudeli õpetamiseks. Eksperimentidel võrreldakse ka mudeleid, mis erinevad õppimisandmete päritolu poolest, pildid pärinevad kas videokaadritest või fotodest. Mudelite täpsuse võrdlemiseks on luuakse testandmestik. Rakenduse prototüübile määratakse optimaalne enesekindluse lävend tundmatute inimeste tuvastamiseks.

Töö tulemusena valmis rakenduse prototüüp, mis võimaldab tuvastada inimesi Youtube keskkonnast pärit videotes. Prototüübis kasutusel oleva närvivõrgu treenimisel kasutati Tensorflow raamistikku.

Eksperimentide käigus selgus, et videokaadritest pärit piltidega treenitud mudel annab parema täpsuse kui fotodelt pärit piltidega treenitud mudel. Parim tulemus saavuti kui õppimispilte on iga inimese kohta 60 ning enesekindluse lävendiks on määratud 20%.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 13 joonist.

# Abstract

Face Recognition from Videos: a Technical Solution and Experiments

Artificial intelligence is currently one of the fastest developing fields of study. Face recognition is just one area where major breakthroughs have been made due to the introduction of artificial intelligence.

The main purpose of the thesis is to create a technical solution for face recognition in order to identify people from videos. Other goals of the thesis are related to improving the accuracy of the neural network used for facial recognition. Experiments are carried out with a different number of training images to determine the optimal number of images to train the model. Experiments are also carried out to compare models trained with training images of different origin. The images originate from photos or video frames. An optimal confidence threshold will be assigned for the prototype to identify unknown people.

As a result of this thesis, a prototype of an application was created that allows identifying people from Youtube videos. Tensorflow framework was used to train the neural network used in the prototype.

Experiments revealed that a model trained with images from video frames gives better accuracy than a model with pictures taken from photos. The best result was achieved when there are 60 learning images per person and the confidence threshold is set to 20%.

The thesis is in Estonian and contains 28 pages of text, 7 chapters, 13 figures.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| Ahenduskiht | *Pooling layer*, närvivõrgu kiht, mille eesmärk on vähendada andmete hulka, pildi puhul pikslite arvu vähendamine. |
| Arvutinägemine | Teadusharu, mis tegeleb arvuti võimega aru saada piltidest ja videotest |
| JSON | *JavaScript Object Notation*, lihtne ja inimesele arusaadav andmevahetusformaat |
| Klassifitseerimine | Protsess, mille eesmärk on otsustada, millisesse etteantud klassi objekt kuulub |
| Klasterdamine | Protsess, mille eesmärk on jagada objektid sarnaste tunnuste alusel gruppidesse. |
| Konvolutsiooniline kiht | *Convolutional layer*, närvivõrgu kiht, mille eesmärk on filtrite abiga tuvastada pildilt erinevaid mustreid ja kujundeid. |
| LFW | *Labeled Faces in the Wild*, andmestik, kuhu on kogutud suur hulk inimeste pilte koos nimedega, kasutatakse näotuvastuse algoritmide testimiseks |
| Masinõpe | Protsess, millega tehisintellektisüsteem täiustab oma otsuste tegemist, õpib |
| Normalisatsioonikiht | *Normalization Layer*, närvivõrgu kiht, mille eesmärk on teisendada väärtused sarnasele skaalale |
| Näotuvastus | Funktsioon, mida kasutakse automaatseks isikutuvastuseks, kasutades selleks näojooni |
| Teek | *Library*, on funktsioonide, makrode, klasside, moodulite vms komponentide kogu |
| URL | *Uniform Resource Locator*, internetiaadress, viide veebis leiduvale ressursile |
| XML | *Extensible Markup Language*, märgistuskeel, kasutatakse ka andmevahetusformaadina |
| YAML | *YAML Ain't Markup Language*, andmeedastusformaat, mis on inimloetav |

# Sisukord

# Jooniste loetelu

# 1 Sissejuhatus

Tehisintellekt on tänapäeval üks kõige kiiremini arenevaid teadusharusid. See on lahutamatu osa meie igapäevasest tehnoloogiakasutusest ja töötab tihti erinevate tegevuste taustal pealtnäha märkamatult. Nutitelefonides mitmeid rakendusi, mis kasutavad tehisintellekti. Näiteks erinevad häälkäskluste rakendused kasutavad masinõppe tehnoloogiaid, et arendada kõnest arusaamist. Meilirakendused suudavad tuvastada rämpsposti saabunud kirjade seast. Facebook kuvab sisu eelnevat tegevust analüüsides ning Spotify soovitab uut muusikat jälgides eelnevalt kuulatut.

Tehisintellekti idee alguseks peetakse 70 aasta vanust arvamust, et arvutid suudavad ühel päeval mõelda nagu inimesed [1]. Sel ajal puudus arvutitel vajalik jõudlus ning uuringud ja teadustööga tegelemine jäi pigem väheste uskujate pärusmaaks. Alles 1997. aastal loodi Deep Blue malemasin, mis suutis malemängus alistada maailmameistri Garry Gasparovi  ning näidata maailmale tehisintellekti potentsiaali [2].

Viimastel aastatel on tänu arvutite ja protsessorite kiirele arengule ning laialdasele uurimustööle toimunud läbimurre tehisintellekti valdkonnas. 2015. aastal suudeti luua arvuti, mis oli edukam nägude tuvastamises kui inimene [3]. USA postisüsteemis on kasutusel arvutid, mis suudavad tuvastada käekirja, et leida kirja sihtkoht [4]. Finantsturgudel tegutsevad tehisintellektipõhised automaatsed kauplejad, kes teevad ostu- ja müügiotsuseid millisekunditega. See annab neile selge eelise inimeste ees [5].

Näotuvastus on arvutinägemise üks põhisuund. Näiteks tuvastab Facebook väga täpselt pildi üleslaadimisel pildil olijad, kasutades selleks üleslaadija kontakte ning näotuvastustehnoloogiat[6].  Ka videopildilt inimeste tuvastamine on laialt levinud ning leiab kasutust näiteks suurte kasiinode turvasüsteemides[7].

## 1.1 Ülesande püstitus ja metoodika

Töö põhieesmärgiks on luua tehniline lahendus, mis suudab näotuvastuse tehnoloogia abil tuvastada videopildilt inimesed. Näotuvastus on üks valdkond, kus arvutid on

praeguseks juba edukamad ning täpsemad kui inimesed. Turvakaamerate ning muu videosalvestuse kasutamine on populaarsemad kui kunagi varem. Seetõttu otsustas autor need ühendada.  Töö põhifookuseks on näotuvastuseks kasutatud närvivõrgu õpetamine masinõppe abil. Seoses sellega on töö eesmärgiks uurida, kuidas mõjutab andmehulga suurus ehk piltide arv närvivõrgu efektiivsust klassifitseerimisel. Samuti on eesmärgiks võrrelda erinevate pilditüüpidega välja töötatud närvivõrkude efektiivsust. Esialgselt on pilditüüpideks fotod ning videotest välja lõigatud pildid. Seoses näotuvastuse kasutamisega on viimaseks eesmärgiks määrata optimaalne enesekindluse lävend, mille eesmärgiks on määrata kindlaks tundmatud isikud.

Tuvastamiseks valitakse 101 tuntud Eesti inimest ning kasutatatakse juba olemasolevaid rakendusi näotuvastuse mudeli loomiseks. Kuna Youtube on kõige levinum video jagamise ja vaatamise veebileht siis kasutatakse videosi, mis just sealt keskkonnast pärit. Video võetakse lahti kaadriteks ning igal kaadril tuvastatakse seal olevat inimest. Inimese tuvastamiseks kasutatakse masinõppe meetodil saadud mudelit. Töö käigus viiakse läbi eksperimente, seos mudeli õpetamisega, et leida kõige optimaalsem õppeandmestik ning enesekindluse lävend inimese tuvastamiseks.

## 1.2 Ülevaade struktuurist

Käesolev töö koosneb 5 osast.

Esimeses osas antakse selgitatakse näotuvastustehnoloogia alustalasid, kirjeldatakse näotuvastuses peamiseid kasutuselolevaid närvivõrgu kihte ning antakse ülevaade varasemalt tehtud uuringutest, mis olid aluseks käesoleva töö koostamisel.

Teises osas tuuakse välja lahenduse tehniline arhitektuur. Selgitatakse programmeerimiskeele valikut ning tuuakse välja rakenduse prototüübis kasutusel olevad teegid ja raamistikud.

Kolmandas osas kirjeldatakse töös kasutatavaid andmeid ning nende kogumise protsessi. Lisaks kirjeldatakse tehtavaid eksperimente ning selgitatakse nende eesmärke seoses realiseeritud prototüübiga.

Neljandas osas analüüsitakse eksperimentide tulemusi. Kirjeldatakse, kuidas eksperimentide tulemusi kasutada rakenduse töö parandamiseks ning millised järeldused eksperimentide põhjal tehti.

Viiendas osas kirjeldatakse prototüübi võimalikke edasiarendusi.

# 2 Seotud kirjandus

## 2.1 Masinõpe

Masinõpe on tehisintellekti valdkond, mis kasutab statistilisi tehnikaid, et anda arvutisüsteemidele võime ’õppida’ olemasolevate andmete põhjal, ilma lisanduva programmeerimiseta [8][9]. Antud kontekstis tähendab õppimine, et süsteem loob andmete põhjal mudeli, mis vastab võimalikult täpselt õppimiseks kasutatud andmetele. Mudelit kasutatakse uute andmete analüüsimiseks [10].

Masinõppe eesmärk on kasutada arvutisüsteemi andmete töötlemiseks, et lahendada ette antud probleem. Selle käigus tuleb programmeerida arvuti nii, et ülesande lahendamiseks kasutatataks minimaalselt ressursse [11]. Masinõpet kasutatakse olukordades, kus töötava algoritmi defineerimine ning rakendamine oleks liialt keeruline [12].

Masinõppe meetodid saab jagada kaheks: juhendajaga õppeks ning juhendajata õppeks. Juhendajaga õpe, nimetatakse ka induktiivõppeks, on protsess, kus õpitakse näidete kogumi järgi. Õppimisandmestik koosneb sisenditest ja väljunditest ning masin(a eesmärk on) leiab algoritmi, mis viib sisendid ja väljundid maksimaalse täpsusega kokku. Juhendajata õppest kõige üldisem on kinnitusega õpe. Juhendajata õppe korral puudub õppimisandmestik ning masin ehk agent töötab loodud keskkonnas. Sisendiks on hetkeolek ning agent peab valima tegevuse. Tegevuse tagajärjel muutub hetkeolek ning agent saab keskkonnalt kas positiivse või negatiivse tagasiside. Sõltuvalt sellest muudetakse otsuse tegemise algoritmi. Peale väga paljude otsuste tegemist, jõutakse algoritmini, mis suudab teha tarku otsuseid [12]. Juhedajata õppe alla liigitub ka klasterdamine. Klasterdamise käigus jagab agent andmestiku sarnaste omaduste põhjal gruppidesse. Grupid tuleb koostada selliselt, et ühe grupi esindajatel on alati rohkem ühist kui kahe erineva grupi esindajatel [11].

Masinõpet rakendatakse erinevate ülesannete lahendamisel. Klassifitseerimine on üks põhilisi ülesandeid, mida lahendatakse juhendatud õppe meetodil. Klassifitseerimisel

jaotatakse süsteemi väljundväärtused kaheks või enamaks klassiks. Näiteks rämpsposti filtreerimise ülesande lahendamisel jagatakse andmestik kaheks klassiks: õiged e-kirjad ja rämpspost. Näotuvastuse tehnoloogia juures on kõik tuvastatavad näod omaette klassid [11].

Regressiivsed ülesanded on samuti juhendajaga õppe abil lahendatavad. Sisenditele on sellisel juhul määratud vastavaks pideva suurusega väljundid. Näiteks korteri müügihinna ennustamisel on väljundiks vastava korteri müügihind, sisenditeks võib võtta tubade arvu ning üldpinna suuruse [11].

## 2.2 Tehisnärvivõrk

Tehisnärvivõrk jäljendab inimaju sisendinformatsiooni töötlemisel ning muutmisel väljundiks. Inimese aju koosneb neuronitest ehk närvirakkudest, mida on ligikaudu 85 miljardit. Närviraku dendriit saab sisendi kas väliskeskkonnalt või eelnevalt närvirakult. Sisendit töödeldakse ning väljund saadetakse edasi, kas järgmisele närvirakule või organile, näiteks lihasele, mis selle peale reageerib. Tehislik neuron töötab sarnaselt. Sisenditeks on sisendmuutujad, millele rakendatakse aktiveerimisfunktsioon ning arvutatakse väljund. Väljund läheb sarnaselt edasi kas järgmisele neuronile või on süsteemi väljundiks. Klassifitseerimise puhul võib olla väljundiks klassi kuulumise tõenäosus [13].

Neuronid jagatakse kihtidesse. Esimeseks kihiks on sisendkiht. Viimast kihti nimetatakse väljundkihiks. Vahepealseid kihte nimetatakse varjatud kihtideks. Täielikult ühendatud närvivõrgu korral on kõik ühe kihi neuronid ühendatud järgmise kihi kõigi neuronitega. Sisendkihil on neuroneid sama palju kui on sisendmuutujaid. Näiteks puuvilja tuvastamisel võime võtta sisendmuutujateks massi ja mõõtmed. Kui puuvilja tuvastamise süsteemi luues võtame 3 sisendmuutujat, mass, laius ja pikkus, siis on sisendkihil 3 neuronit. Kui tuvastame 4 klassi, näiteks õun, pirn, ploom ja virsik, siis väljundkihil on 4 neuronit [13].

## 2.3 Konvolutsiooniline närvivõrk

Näotuvastuse korral on kõige paremaid tulemusi saavutatud konvolutsioonilise närvivõrguga. Närvivõrgu ülesehitusel kasutatakse konvolutsioonilisi kihte vaheldumisi ahendus- ning normalisatsiooni kihtidega. Viimase kihina kasutatakse klassifitseerimisel täielikult ühendatud kihti [14].

Konvolutsioonilise närvivõrgu korral ei ole neuronid omavahel täielikult ühendatud kuna sisendmuutujate hulk on liiga suur. See tekitaks ülesobivuse probleemi ning närvivõrk oleks liialt mahukas ega skaleeruks suurte resolutsioonidega piltidele. Samuti pole vajadust täielikult ühendatud võrgustiku järgi, sest pildil moodustavad kindla kujutise vaid üksteisele lähedal asuvad pikslid. Üksteisest eemal asuvate pikslite koos vaatlemine ei ole otstarbekas [14].

Konvolutsioonilise närvivõrgu ülesehitus võimaldab leida tähendusrikkaid mustreid väga suurtest ning mitmemõõtmelistest andmehulkadest. See omadus on võimaldanud olulisi läbimurdeid pildi, video ja heli tuvastamise valdkondades. Konvolutsiooniline kiht suudab leida sarnaseid mustreid üle kogu andmehulga kasutades konvolutsioonilisi filtreid, mis leitakse õppimise käigus. Konvolutsiooniliste filtritega käiakse üle kogu andmehulk, mis võimaldab leida mustreid sõltumata reaalsest asukohast [14].

Ahenduskihi eesmärk on vähendada sisendmuutujate arvu. Pildi puhul tähendab see sisuliselt pikslite ühendamist. Üks levinumaid meetodeid on 2*2 pikslite ruudu ühendamine üheks. Üldjuhul kasutatakse maksimaalset ahendamist, mis tähendab et 4 piksli seast valitakse piksel, millele vastab kõige kõrgema väärtusega arv. See meetod on näidanud paremaid tulemusi kui näiteks keskmise arvutamine. Maksimaalse ahendamise puhul väheneb pikslite ehk sisendmuutujate arv 75% [15].

Normalisatsioonikihi eesmärk on teisendada sisendväärtused sarnasele vahemikule. Näiteks teisendatakse kõik sisendite väärtused vahemikku 0-1. Erinevad väärtuste vahemikud vähendavad õppimise kiirust [16]. Normalisatsioonikihil kasutatakse väärtuste teisendamiseks mittelineaarseid funktsioone, millest levinuim on Relu funktsioon oma lihtsuse ning positiivsete tulemuste tõttu [17].

## 2.4 FaceNet ja näotuvastustehnoloogia

FaceNet on rakendus, mille eesmärk on mõõta piltidel olevate nägude sarnasust. Nägude sarnasuse mõõtmist saab kasutada näiteks näotuvastuses, nägude kontrollimisel, kinnitamisel ning nägude klasterdamisel. FaceNetis kasutatakse nägude sarnasuse määramisel konvolutsioonilist närvivõrku. FaceNeti rakendusel on õnnestunud saada fotodest koosneval “Labeled Faces in the Wild” (LFW) andmekogumil ennustuse täpsus 99.63%. LFW on andmebaas, millesse on kogutud üle 13000 pildi peaaegu 6000 inimese kohta. Andmekogumit kasutatakse laialdaselt näotuvastuse algoritmide testimiseks. FaceNeti süsteem vähendas valede otsuste hulka eelnevate avalikustatud tulemustega võrreldes 30% [18].

Käesolevale tööle on FaceNet rakendus eeskujuks, sest rakenduse tulemused on olnud märkimisväärselt paremad kui varasemad. Loodavas prototüübis kasutatakse sama närvivõrgu struktuuri ning nägude väljalõikamise algoritmi, mis on kasutusel FaceNet rakenduses. Kui FaceNet keskendub piltidelt nägude tuvastamisele, siis prototüübi eesmärk on tuvastada inimesi videotest.

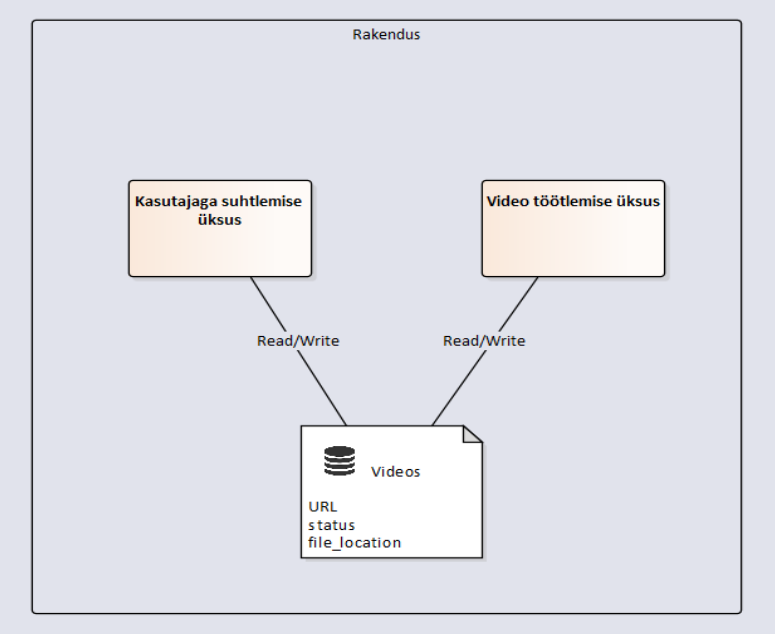
# 3 Tehniline arhitektuur

## 3.1 Python – programmeerimiskeele valik

Rakenduse prototüübi realiseerimise keeleks on valitud Pythoni programmeerimiskeel. Peamine põhjus peitub selles, et pildilt nägude välja lõikamiseks ning mudelite treenimiseks kasutatakse suures ulatuses Facenet rakenduse meetodeid. Facenet rakendus ise on realiseeritud Pythoni keeles [19]. Sammuti oli 2018. aastal GitHubi andmete põhjal Python kõige aktiivsem masinõppe keel. Tensorflow oli kõige aktiivsem masinõppe projekt GitHubis [20].

## 3.2 Prototüübi komponendid

Töö käigus luuakse rakenduse prototüüb, mis võimaldab Youtube keskkonnast pärit videotelt nägusid tuvastada.




Joonis 1. Prototüübi skeem

Joonisel 1 on kujutatud prototüübi skeem. Rakenduse prototüüp on jagatud kaheks üksuseks. Ühe üksuse ülesanne on kasutajaga suhtlemine veebibrauseri kaudu. Teise üksuse ülesanne on kasutaja soovitud videote allalaadimine ning nägude tuvastamine videolt. Prototüübi jaotus on tingitud sellest, et video allalaadimine ning töötlemine nõuab palju aega. Jagates prototüübi kaheks üksuseks saab kasutaja kohe tagasisidet soovitud video staatuse kohta. Mõlemad üksused realiseeritakse täielikult Python programmeerimiskeeles.

## 3.3 Kasutajaga suhtlemise üksus

Kasutajaga suhtlemise üksuses kasutatakse põhifunktsionaalsuse loomiseks Pythoni Flask raamistikku. Veebivormide loomiseks ning andmete vormindamiseks kasutatakse WTForms teeki. Veebivormi kasutajaliides on väga lihtne. Vormil on üks tekstisisestuse väli, millele tuleb kirjutada Youtube video internetiaadress (URL). Kui videot pole eelnevalt töödeldud või video töötlus on veel pooleli, siis teavitatakse sellest kasutajat. Video URL kirjutatakse andmebaasi tabelisse koos staatusega ning selle järgi teab video töötlemise üksus, et video vajab töötlemist. Kui video on eelnevalt töödeldud siis andmebaasist leitakse töötlemisel saadud faili asukoht ning tagastatakse kasutajale JSON formaadis vastus, kus on ära toodud igas kaadris tuvastatud inimesed ning nende ennustuste enesekindluse tulemused. Lisas 1 on toodud näide saadud vastusest. Prototüübis imiteeritakse andmebaasi tabelit tekstifailiga.

## 3.4 Video töötlemise üksus

Video töötlemise üksus on pidevalt aktiivne ning saadab andmebaasi päringuid, et leida uus video, mida töödelda. Video leidmisel uuendatakse andmebaasis staatust ning viiakse läbi töötlemise protsess. Protsess algab video allalaadimisega, milleks kasutatakse Youtube-dl teeki. Seejärel lõigatakse video kaadriteks OpenCV teegi abil. Teek võimaldab täpsustada, mitu kaadrit igast sekundist välja lõigata. Seejärel lõigatakse igast kaadrist välja näod. FaceNet rakenduses on loodud algoritm, mis võimaldab näo pildilt üles leida ning lõigata nägu välja õige suuruse ja õige resolutsiooniga. Selles prototüübis kasutatakse sama algoritmi. Iga leitud näo kohta

tehakse ennustus masinõppega saadud mudelite põhjal. Ennustused kirjutatakse faili JSON formaadis. Andmebaasis uuendatakse staatust ning lisatakse JSON formaadis oleva faili asukoht.

Video töötlemise üksuse juures kõige olulisem komponent on närvivõrgu mudel, mille põhjal tehakse ennustused. Prototüübi täpsus ning kasulikkus on otseses sõltuvuses mudeli ennustuste täpsusega. Seetõttu on uuringu põhifookuses just mudeli täpsuse parandamine. Mudel peab suutma ennustada tuvastatavaid nägusid õigesti ning tuvastatamatute nägude korral peab süsteem määrama näo tundmatuks. Mudeli treenimise eksperimentidel keskendutakse õppimisandmestikule ehk õppimispiltidele. Fotodega treenitud mudeli täpsust võrreldakse videokaadritest pärit piltidega treenitud mudeli täpsusega. Samuti vaadeldakse kuidas mõjub õppimispiltide arv mudeli täpsusele, eesmärgiga leida kõige optimaalsem arv õppimispilte. Mudeli täpsuse hindamisel on oluline, et parem on määrata inimene tundmatuks kui teha vale ennustus.

## 3.5 Prototüübis kasutatud raamistikud ja teegid

### 3.5.1 Flask

Flask on Pythoni veebiserverirakenduse raamistik. Sellega saab kiiresti teha veebirakenduse, mis on skaleeritav keerulistele rakendustele. Tänapäevaks on saanud sellest üks levinumaid veebirakenduste raamistikke. Flaski kasutades on arendajal võimalik ise valida oma teegid, mis teiste sarnaste raamistike puhul alati võimalik ei ole [21].

### 3.5.2 WTForms

WTForms on paindlik vormindamise ja vormi valideermise teek, mis sobib erinevate veebirakenduste raamistikega. WTForms ei sõltu valitud veebirakenduse raamistikust. WTForms võimaldab genereerida vorme, kuid võimaldab samas arendajal ise luua kasutajaliidese [22].

### 3.5.3 Youtube-dl

Youtube-dl on käsurearakendus, millega on võimalik alla laadida videoid Youtube'ist ja teistest veebikeskkondadest. Youtube-dl on võimalik integreerida pea iga programmeerimiskeelega, aga kuna rakendus on kirjutatud Python keeles, siis Pythonis on võimalik seda kasutada ka teegina [23].

### 3.5.4 OpenCV (open source computer vision)

OpenCV on arvutinägemise ja masinõppe teek. Teegis on üle 2500 algoritmi, mida saab kasutada nägude tuvastamiseks, objektide identifitseerimiseks, liikuvate objektide jälgimiseks, 3D mudelite loomiseks jne [24].

### 3.5.5 Pandas

Pandas on Pythoni teek, mis pakub kiireid, paindlikke ja väljendusrikkaid andmestruktuure, mille eesmärk on muuta struktureeritud andmetega töötamine lihtsamaks. Pandase andmestruktuurid imiteerivad maatrikse, tabeleid ja vektoreid [25].

### 3.5.6 Numpy

Numpy on teek, mis toetab võimsaid mitmemõõtmelisi massiive, koos suure hulga matemaatiliste funktsioonidega, mida saab massiividel rakendada. Numpy võimaldab ka andmebaasi integratsiooni [26].

### 3.5.7 Tensorflow

Tensorflow on masinõppe algoritmide kirjeldamise teek ning sammuti nende algoritmide käivitamise implementatsioon. Tensorflow'd saab kasutada närvivõrkude mudelite kirjeldamiseks, ennustuste tegemisel ning mudelite treenimisel. Seda on kasutatud uuringute tegemisel ning masinõppe süsteemide rakendamisel toodangus [27].

### 3.5.8 Scikit-learn

Scikit-learn on Pythoni teek, mis pakub laialdasi masinõppe algoritme juhendajaga ja juhendajata õppe jaoks. Fookuses on kasutamise lihtsus ning jõudlus [28].

# 4 Eksperimendid

Rakenduse prototüübi testimiseks ning tulemuste analüüsimiseks viidi läbi mitmed eksperimendid. Eksperimentide põhifookus on näotuvastuseks kasutatavate mudelite täpsuse ja enesekindluse tulemuste analüüs. Eksperimentidel võrreldakse mudeleid mis on õpitud, erinevate õppimisandmestike järgi. Õppimisandmestikel on erinev andmehulk ehk piltide arv ning andmete päritolu, ehk kas pildid on pärit videotest või fotodelt. Eksperimentidel eristatakse kahte olukorda. Esimeses on kõik testimisel kasutatud inimeste pildid mudelile tuvastatavad. Teises olukorras on testandmestiku seas nii tuvastatavate inimeste pilte kui ka süsteemile tundmatute inimeste pilte. Kõik testandmestiku pildid pärinevad videokaadritest.

## 4.1 Andmed

Prototüübi testimiseks valiti tuvastatavateks nägudeks XIII Riigikogu koosseis ehk 101 inimest. Eksperimentide tegemisel eristatakse õppimis- ning testimisandmeid.

Õppimisandmed on jaotatud kaheks piltide päritolu järgi. Esiteks on iga tuvastatava inimese kohta 10 pilti, mis pärinevad erinevatelt fotodelt. Kokku on kasutuses seega 1010 fotot. Teiseks on iga tuvastatava inimese kohta 60 pilti, mis on välja lõigatud videote kaadritest. Videotest pärit pilte on kokku 6060. Iga inimese kohta on valitud 3-4 videot. Eksperimentide käigus kasutatakse erineva päritoluga andmeid eraldi, omavahel võrdlemiseks ning ka koos, üldise analüüsi tegemiseks.

Testandmed pärinevad kõik videokaadritest. Testvideoid on kokku 10. 8 neist on erinevate erakondade valimisreklaamid, 1 juhul on tegemist Riigikogu juhatuse pressikonverentsiga ning 1 video on lühike lõik Riigikogu istungi salvestusest. Testandmete seas ei ole videosi, mida kasutati õppimisandmete kogumisel, ehk ennustuse tegemisel on kindel, pilt ei olnud õppimisandmete seas. Testandmetel on esindatud 26 tuvastatavat inimest, nende kohta on kokku 498 pilti.

Testandmete hulka kuuluvad ka testvideotest pärit mittetuvastavate inimeste näod. Mittetuvastatavaid pilte on välja valitud 10 inimese kohta ning kokku 100 pilti.

## 4.2 Eksperimentide eesmärgid

Eksperimentide eemärgiks on välja uurida, kuidas mõjutab erineva suurusega õppimisandmete hulk treenitud mudeli täpsust ning enesekindluse tulemust. Selle järgi on võimalik valida optimaalne õppimisandmete hulk, et õppimisprotsess ei võtaks liigselt aega ja oleks samal ajal võimalikult täpne. Tundmatute inimeste tuvastamiseks kasutatakse enesekindluse tulemust, kui vastav mõõt jääb alla määratud enesekindluse lävendile loetakse inimese tundmatuks. Samuti on eesmärgiks leida kõige sobivam enesekindluse lävend, et oleks võimalik tuvastatavad inimesed tuvastada ning tundmatud inimesed lugeda tundmatuteks. Tundmatute inimeste tuvastamine on oluline, sest olukord, kus kõik videos esinenud inimsesed olid õppimisandmete seas, on väga haruldane. Sammuti peab süsteem teatama, et leiti nägu, mida tuvastada ei suudetud.

## 4.3 Eksperimentide kirjeldused

Esimese eksperimendina kasutati õppimisandmetena nii fotodelt pärit nägusid kui ka videokaadridest pärit pilte, ilma neid eristamata, kokku 70 pilti inimese kohta. Eksperimentidel kasutatakse mudeli õpetamisel erinevat arvu pilte iga inimese kohta. Testandmete hulka kuuluvad selles eksperimendis ainult tuvastatavad näod. Kasutades õpetatud mudelit tehakse ennustused ning jälgitakse täpsust, ning enesekindluse tulemust. Igal katsel valitakse õppimiseks juhuslikud pildid. Testimiseks kasutatakse alati kõiki tuvastatavaid nägusid.

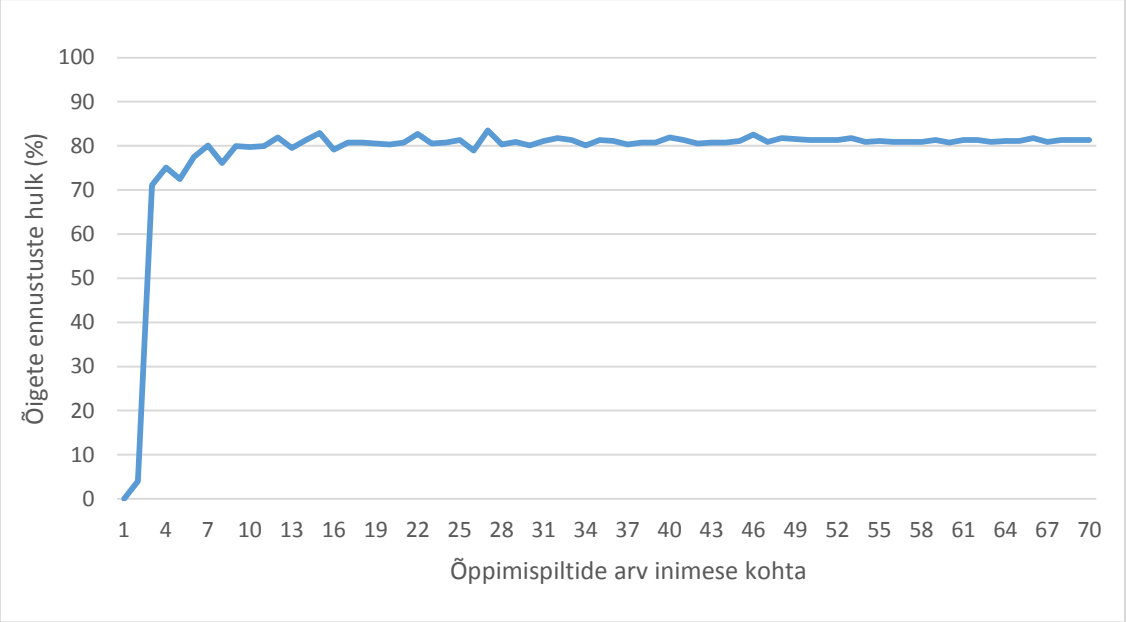
Teise eksperimendina kasutati õppimisandmetena nii fotodelt pärit nägusid kui ka videokaadridest pärit pilte, kasutades neid eraldi. Testandmetena kasutatakse ainult tuvastatavaid nägusid. Teise eksperimendi eesmärk on võrrelda õpetatud mudeli tulemusi erineva päritoluga piltidega. Mudelite õpetamisel kasutatakse erinevat arvu pilte 1-10 ning pildid valitakse juhuslikult. Mudelite põhjal tehakse ennustused.

Kolmandas eksperimendis kasutatakse õppimisandmetena nii fotodelt pärit nägusid kui ka videokaadridest pärit pilte, ilma neid eristamata, kokku 70 pilti inimese kohta.

Testandmete hulka kuuluvad nii tuvastavad näod kui ka mittetuvastavad näod. Õppimisandmete põhjal treenitakse mudel, mille põhjal tehakse ennustused. Katse jooksul jälgitakse ennustuste enesekindluse tulemust. Määratakse enesekindluse lävend, ning kui ennustuse enesekindlus on madalam määratust, siis määratakse ennustuse tulemus tundmatuks. Ennustuse tulemus loetakse õigeks, kui tuvastatav inimene tuvastatakse õigesti või kui tuvastamatu inimene loetakse tundmatuks. Ennustus loetakse valeks, kui tuvastatav inimene tuvastatakse valesti või kui tuvastamatut inimest ei loeta tundmatuks. Kui tuvastatav inimene määratakse tundmatuks, siis ei loeta tulemust ei õigeks ega valeks. Katset korratakse erinevate enesekindluse lävenditega.

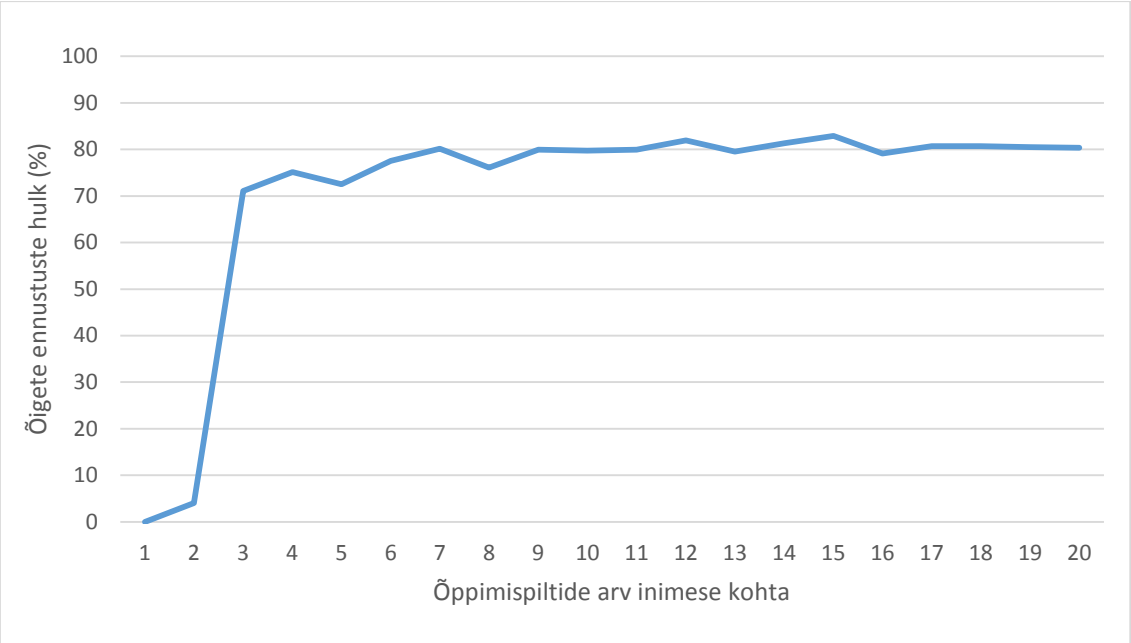
# 5 Tulemused ja analüüs

## 5.1 Eksperiment ainult tuvastatavate nägudega



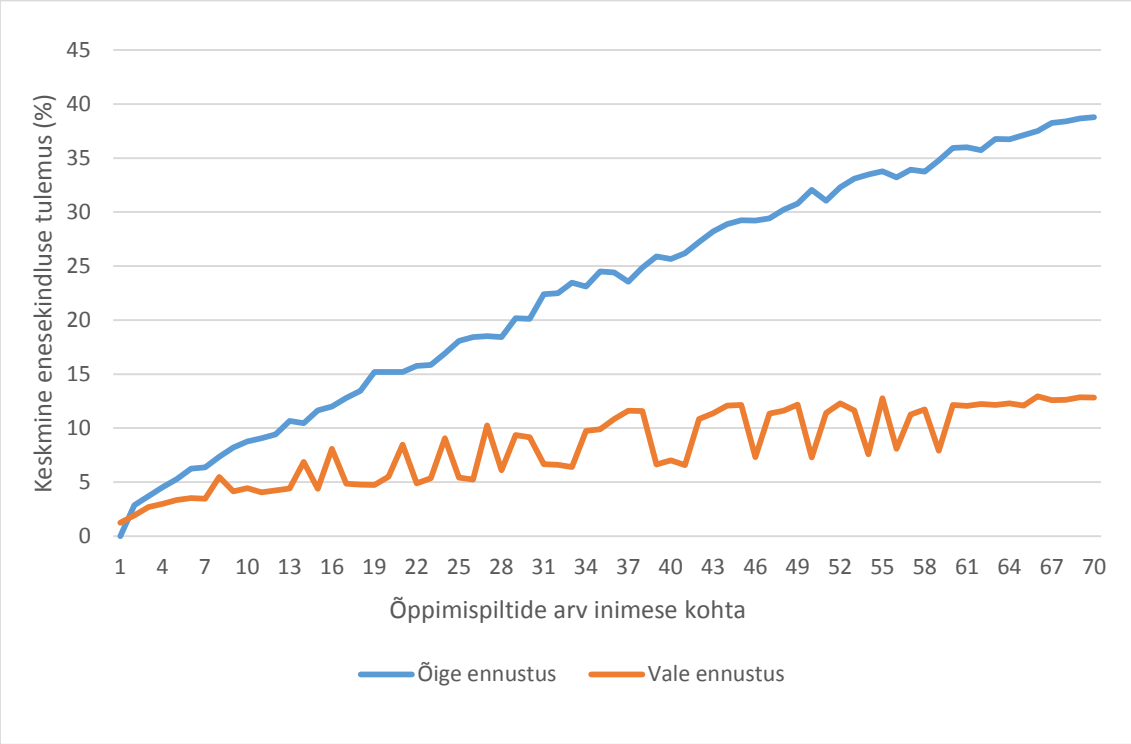

Joonis 2. Õigete ennustuste hulk sõltuvualt õppimispiltide arvust

Joonisel 2 on toodud tuvastatavate piltide kohta tehtud ennustuste hulk sõltuvalt mudeli treenimiseks kasutatud piltide arvust. Iga pildi kohta ennustati kes on pildil ning hinnati kas ennustus oli õige või vale. Treenimiseks kasutati iga tuvastatava inimese kohta kuni 70 juhuslikult valitud pilti. Kasutades treenimiseks ainult ühte pilti iga inimese kohta, ei tehtud mitte ühtegi õiget ennustust. Maksimaalne ennustuse täpsus tuli veidi üle 80%. Üsna kiiresti jõuti kõrgele täpsuse tasemele, treenimisel ainult 10 pildiga on jõuti peaaegu maksimaalsele täpsuse tasemele ning lisanduvad pildid seda ei tõstnud. Võib järeldada, et olukorras, kus on ainult tuvastatavad pildid, siis ainult täpsust arvesse võttes võiks piisata umbes 10 pildist iga inimese kohta, et saada piisavalt hea täpsus.




Joonis 3. Õigete ennustuste hulk sõltuvualt õppimispiltide arvust (Piltide arv 1-20)

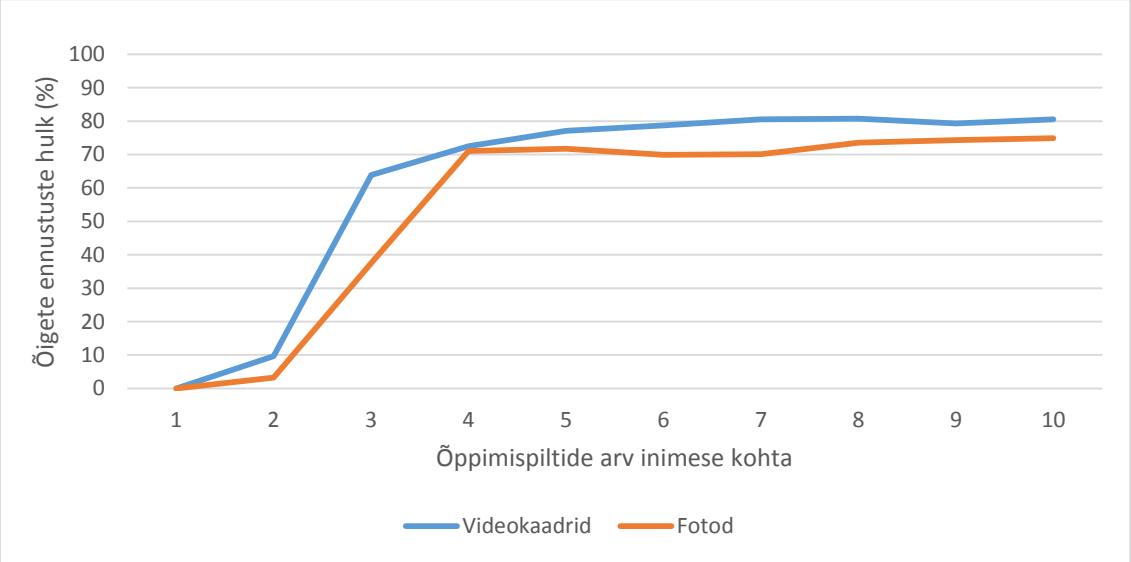
Joonisel 3 on toodud ennustuse sõltuvus treeningpiltide arvu 1-20 kohta. Jooniselt on näha, et suurem hüpe tekib, kui iga inimese kohta on kõigest 3 pilti. Selle juures on täpsus juba 70%. Keeruline on põhjendada, miks on tõus niivõrd kiire ja järsk. 80% täpsuse pidevaks saavutamiseks on tarvis umbes 10 pilti.




Joonis 4. Keskmine enesekindluse tulemuse õige ja vale ennustuse kohta sõltuvalt õppimispiltide arvust.

Joonisel 4 on toodud keskmised õige ja vale enesekindluse tulemused sõltuvalt inimese kohta kasutatud piltide arvust. Õige ennustuse tulemus on oodatult ühtlaselt kasvav. Kuigi oleks võinud oodata, et piltide kasvamisel hakkab valede ennustuste enesekindluse tulemus langema, siis seda ei juhtunud. Valede ennustuste keskmine enesekindlus on väga kõikuv ning pigem kasvavas suunas. Süsteemi jaoks on olukord seda positiivsem, mida kõrgem on positiivsete ennustuste enesekindlus valede omast. Graafikult on võimalik välja lugeda, milline võiks olla minimaalne enesekindluse tulemus, mille puhul määratakse isiku. Kui võetakse õppimisandmeteks 55 pilti, ning määrata enesekindluse lävendiks 20%, siis sellisel juhul võiks saada väga kõrge täpsuse määratud piltide kohta ning ülejäänud loetaks tundmatuks.

## 5.2 Eksperiment piltide päritolu võrdlemiseks



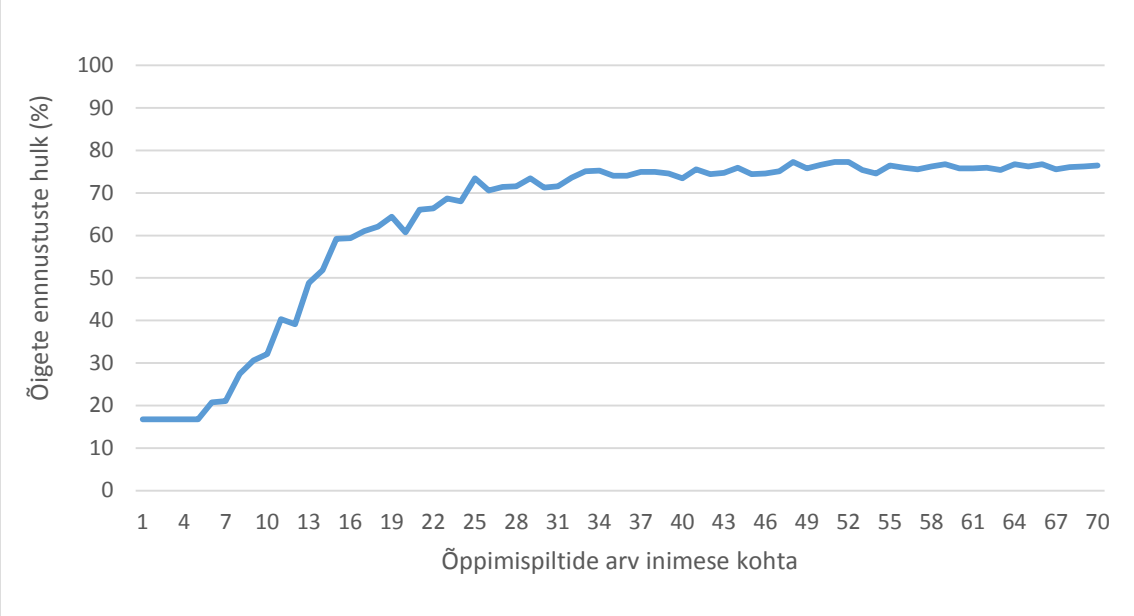

Joonis 5. Õigete ennustuste hulk sõltuvalt õppimispiltide arvust, eristades piltide päritolu videokaadritest ja fotodest.

Joonisel 5 on toodud õigete ennustuste hulk sõltuvalt õppimisel kasutatud piltide arvust. Joonisel võrreldakse mudeleid, mille õpetamisel on ühel juhul kasutatud pilte videokaadritest ning teisel juhul fotodelt pärit pilte. Õppimisel kasutatud piltide arv mõjutab õigete ennustuste hulka sarnaselt nii videokaadrite kui ka fotode korral. Tulemused on sarnased, aga igas punktis on videokaadritest pärit piltidega treenitud mudel veidi täpsem. Tulemus on ootuspärane, sest testimisel kasutatud andmed

pärinevad videotest. Videokaadrite kasuks räägib ka asjaolu, et kiiresti jõuti maksimaalse täpsuseni, mis saavutati varasemas eksperimendis.

## 5.3 Eksperiment tuvastatave ja tuvastamatute nägudega

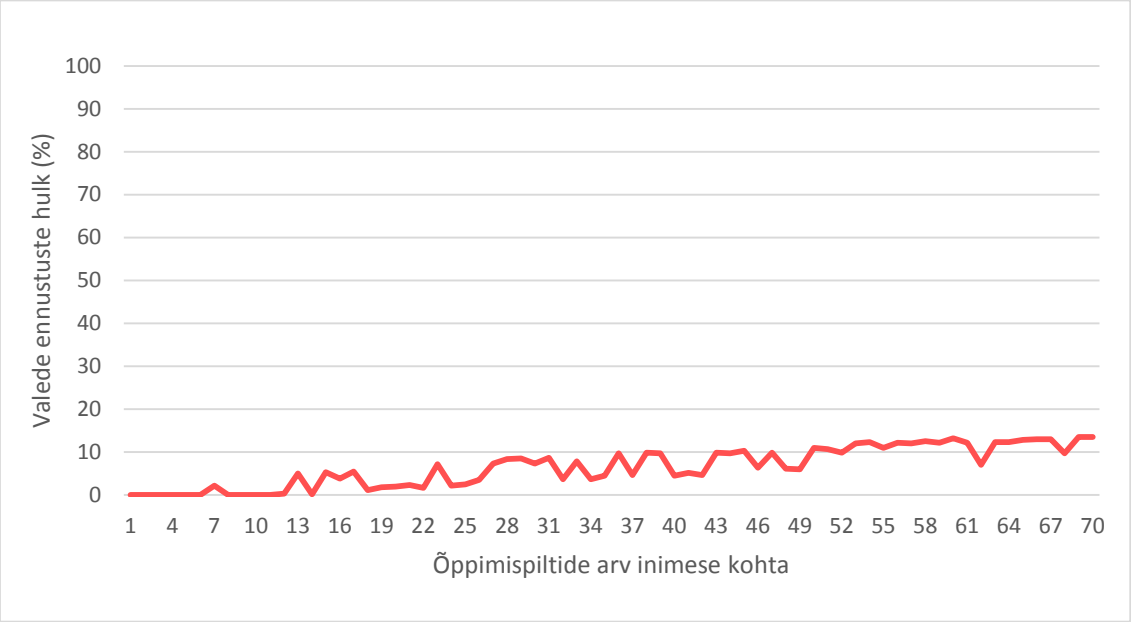
3. eksperimendis on õppimisandmeteks näod nii fotokaadritest kui ka videotest. Testandmed on kõik tuvastatavad ja ka tuvastamatud näod. Esimeses katses määratakse enesekindluse lävendiks 10%, ehk kui ennustuse enesekindluse tulemus on alla 10% siis määratakse nägu tundmatuks. Katses loetakse ennustus õigeks, kui tuvastatav nägu tuvastatakse õigesti ning kui tuvastamatu näo enesekindluse tulemus on alla 10% ehk määratakse tundmatuks. Katses loetakse ennustus valeks, kui tuvastatavale näole tehakse vale ennustus või kui tuvastamatule näole tehtava ennustuse enesekindlus on üle 10%. See tähendab, et kui tuvastatavale näole määratakse tundmatu, siis ei loeta ennustust ei õigeks ega valeks.




Joonis 6. Õigete ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui enesekindluse lävend on 10%

Joonisel 6 on esitatud õigete ennustuste hulk, kui enesekindluse lävendiks määrati 10%. Kui õppimisel kasutati iga inimese kohta alla 6 pildi siis tulid õiged ennustused ainult tuvastamatutest piltidest, sest kõigi ennustuste enesekindluse tulemus on väga väike.

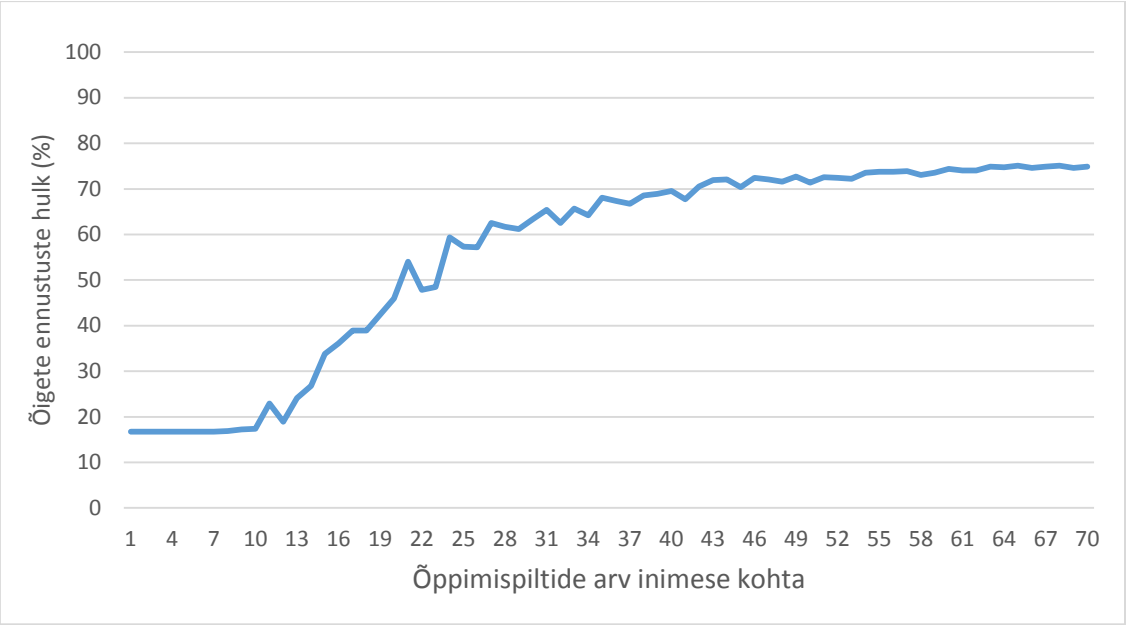
75% õigete ennustusteni jõutakse, kui õppimiseks kasutatakse 30 pilti iga inimese kohta. Suurema arvu piltide korral õigete ennustuste hulk ei suurene märkimisväärselt.




Joonis 7. Valede ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui enesekindluse lävend on 10%

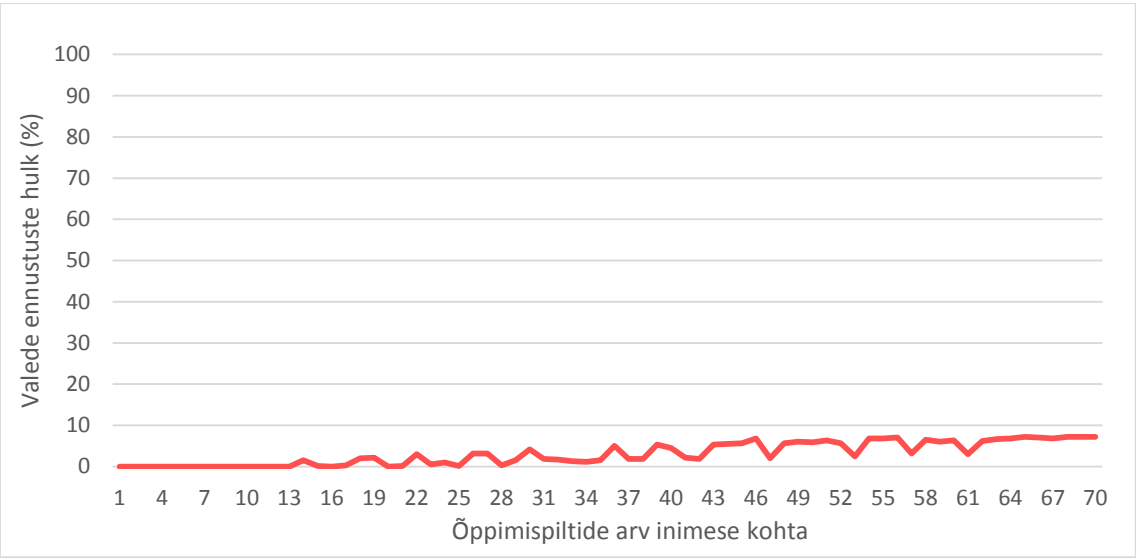
Joonisel 7 on välja toodud valede ennustuste hulk sõltuvalt mudeli õpetamiseks kasutatud õppimispiltide arvust iga inimese kohta. Väikese arvu õppimispiltide korral on enesekindluse tulemused madalad ning seetõttu määratakse kõik ennustused tundmatuteks, ning tundmatut ei loeta ei õigeks ega valeks, seetõttu on ka valede hulk väike. Kui iga inimese kohta on 30 õppimispilti siis on valede hulk lausa 10% ning 70 pildi juures jõuab valede hulk 15% lähedale. Süsteemi töös on positiivsem määrata inimene tundmatuks kui valeks inimeseks.

2. katseks on määratud enesekindluse lävendiks 15%. Test- ning treeningandmed on samad, mis eelmises eksperimendis.




Joonis 8. Õigete ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui minimaalne enesekindluse lävend on 15%

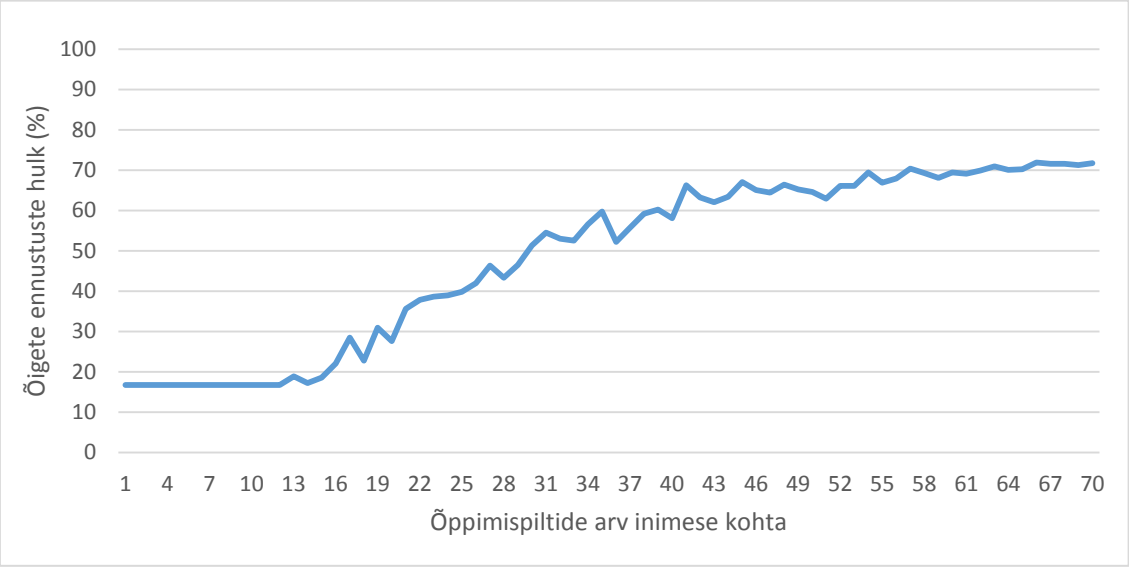
Joonisel 8 on välja toodud õigete ennustuste hulk, kui enesekindluse lävendiks määrati 15%. Sarnaselt varasemaga on madala piltide arvuga treenitud mudeli ennustused madala enesekindluse tulemusega ning jäävad alla määratud lävendile. 75% õigete ennustusteni jõuti umbes 60 õppimispildiga iga inimese kohta. Rohkemate piltidega ennustuste täpsus stabiliseerub. Katse juures on positiivne, et enesekindluse lävendi tõstmisel ei vähenenud õigete ennustuse hulk suurte õppimisandmete korral.




Joonis 9. Valede ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui minimaalne enesekindluse lävend on 15%

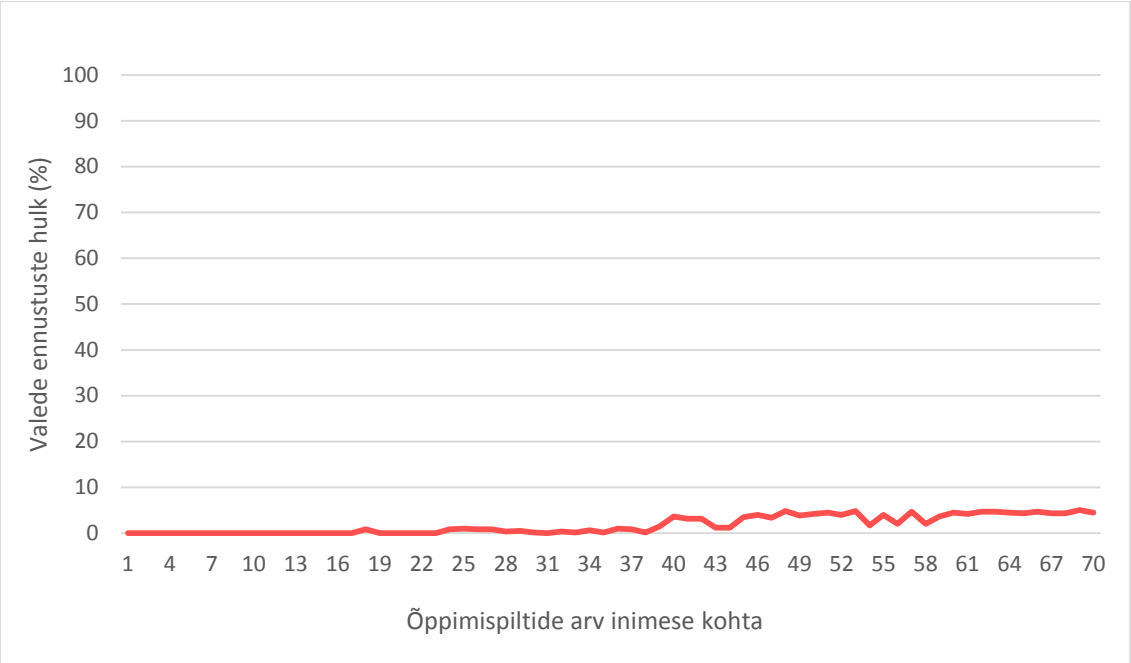
Joonisel 9 on välja toodud valede ennustuste hulk sõltuvalt mudeli õpetamiseks kasutatud õppimispiltide arvust iga inimese kohta. Valede ennustuste hulk väheneb igal võimalikul õppimispiltide arvul võrdluses eelmise katsega. Maksimaalne valede ennustuste arv jäi alla 10%, mis näitab, et määrates kõrgema minimaalse enesekindluse tulemuse, saame vähem valesid ennustusi. Tulemus on positiivne, sest tehti vähem valesid otsuseid ning tõenäoliselt on rohkem tundmatuks määratud pilte.

3. katseks määrame minimaalseks enesekindluse lävendiks 20%. Testandmed ning treeningandmed on samad, mis eelmises katses.



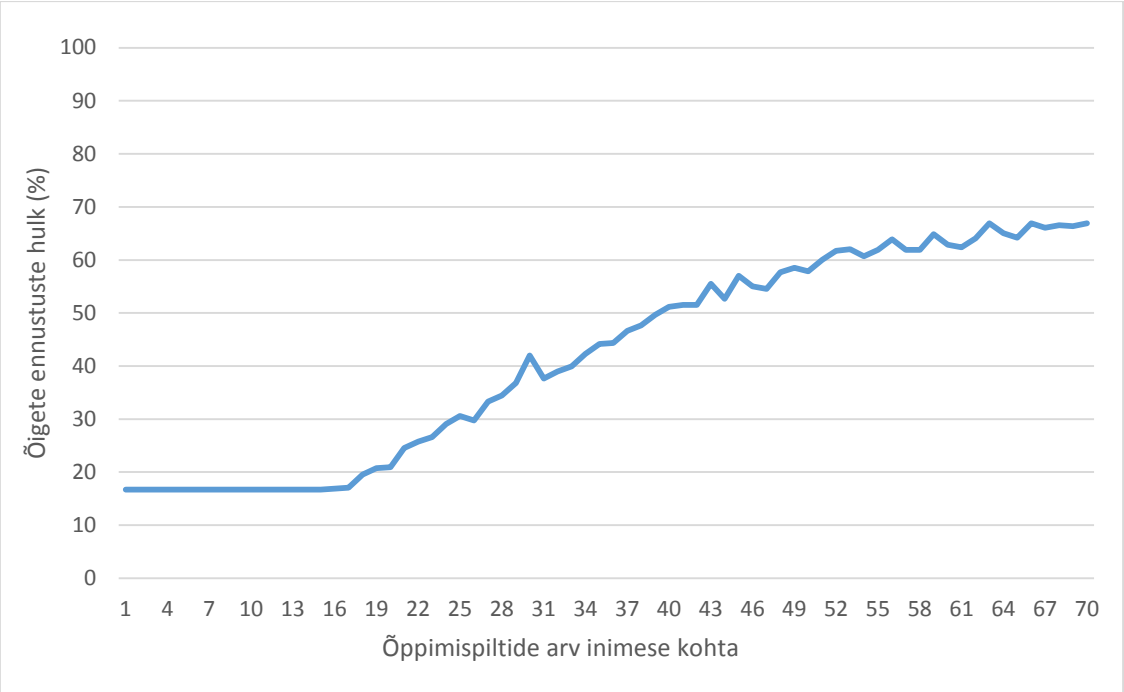

Joonis 10. Õigete ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui minimaalne enesekindluse lävend on 20%

Joonisel 10 on välja toodud õigete ennustuste hulk, kui enesekindluse lävendiks määrati 20%. Sarnaselt varasemaga jäävad madala piltide arvuga treenitud mudeli ennustused alla määratud enesekindluse lävendi. Periood, kus ennustused jäävad alla määratud piiri muutub oodatule pikemaks. 70% õigete ennustusteni jõuab umbes 60 õppimispildiga iga inimese kohta. Rohkemate piltidega on tõuseb õigete ennustuste hulk vaevu paar protsenti. Endiselt on õigete ennustuste hulk kõrge, ehki isegi 5% langus eelmisega võrreldes on negatiivne.



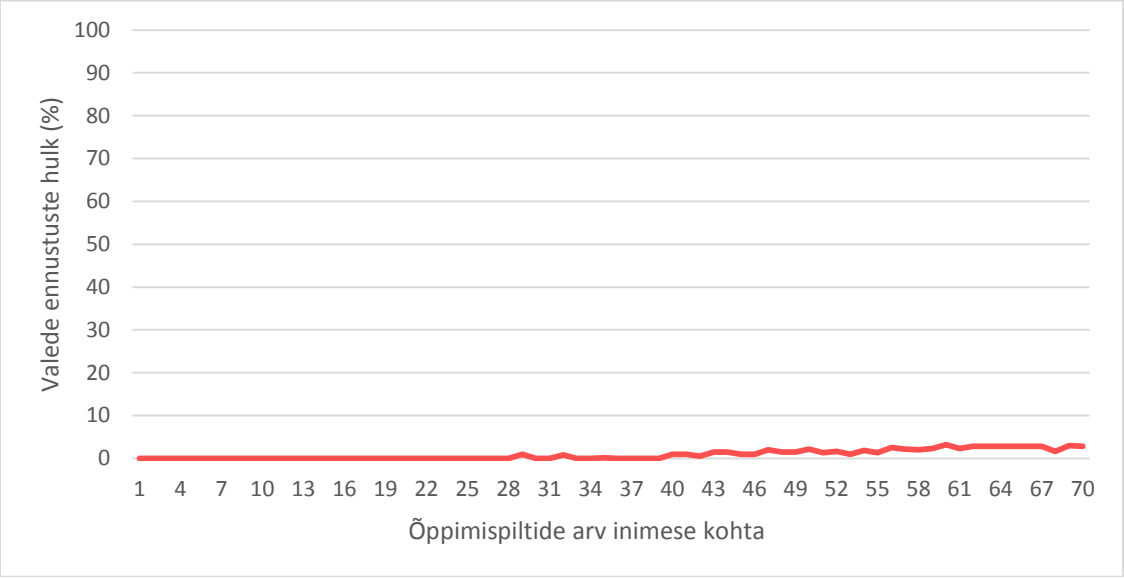

Joonis 11. Valede ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui minimaalne enesekindluse lävend on 20%

Joonisel 11 on välja toodud valede ennustuste hulk sõltuvalt mudeli õpetamiseks kasutatud õppimispiltide arvust iga inimese kohta. 20% enesekindluse lävendiga, ei määratud inimesi valesti põhimõtteliselt kuni 39 õppimispildini iga inimese kohta. Maksimaalne valede ennustuste hulk oli 5% mis on 3 korda väiksem kui 10% määratud minimaalse tulemuse juures. Ideaalses olukorras, ei tehtaks ühtegi vale ennustust, samas nõuab see veelgi kõrgemat määratud minimaalset enesekindluse tulemust. Siis võib tekkida olukord, kus meie õigete ennustuste arv väheneb, ning lõppkokkuvõttes on süsteemi tulemused kehvemad. Hetkel langes õigete ennustuste hulk maksimaalse piltide arvu korral 4%.




Joonis 12. Õigete ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui enesekindluse lävend on 25%

Joonisel 12 on välja toodud õigete ennustuste hulk, kui enesekindluse lävendiks määrati 25 %. Maksimaalne õigete ennustuste hulk on langenud alla 70%, võrreldes 20% lävendiga on täpsus langenud 5%.




Joonis 13. Valede ennustuste hulk sõltuvalt õppimispiltide arvust iga inimese kohta, kui enesekindluse lävend on 25%

Joonisel 13 on välja toodud valede ennustuste hulk sõltuvalt mudeli õpetamiseks kasutatud õppimispiltide arvust iga inimese kohta. 25% lävendi juures on valede

ennustuste arv väga väike, maksimaalselt 3%, siiski pole tulemus ideaalne ning muutus on vaevu märgatav.

Selgub, et lävendi tõstmine 20%-lt 25%-le ei avaldanud väga positiivset mõju. Õigete ennustuste hulk langes 5%, valede ennustuste hulk aga vähenes kõiges 2%. Samas kui tõsteti lävendit 15%-lt 20%-le langes nii õigete kui valede ennustuste hulk 5%. Selle põhjal võib öelda, et lävendi tõstmine 20% on õigustatud, aga edasine tõstmine enam mitte.

## 5.4 Järeldused ja diskussioon

Eksperimentide tulemused olid ootuspärased. Suurema õppimisandmestikuga mudelid saavutasid parema tulemuse kui väiksemaga. Ainult tuvastatavate nägude eksperimendist selgus, et maksimaalse ennustuse täpsuse saavutamiseks, piisab ainult 10 pildist iga inimese kohta. Samas olid sellisel juhul enesekindluse tulemused väga madalad, õigete ja valede ennustuste enesekindlused olid üksteisest eristamatud. Piltide arvu suurendamisel suuri muudatusi ennustuste täpsusel ei esinenud aga õigete ennustuste enesekindluse tulemus tõusis stabiilselt. See on oluline, sest olukordades, kus on tegemist ka tundmatute nägudega, on vaja enesekindluse lävendit, et määrata tundmatuid inimesi. Eksperimendi tulemustest võib välja lugeda, et optimaalne enesekindluse lävend võiks olla umbes 20% juures, ning seda katsetati hilisemas eksperimendis. Samuti saab järeldada, et suurem piltide arv annab suurema enesekindluse tulemuse, aga piltide arvu tõstmine muudab ka õppimisprotsessi pikemaks.

Teises eksperimendis uuriti mudelite tulemusi juhul kui õppimisandmed on erinevad päritolult. Selgus, et videokaadritest pärit õppimisandmed annavad alati parema tulemuse kui fotodelt pärit pildid. Erinevused olid küll marginaalsed, kuid siiski alati olid videokaadritest treenitud mudelid paremad. Tulemus on ootuspärane, sest testandmestik pärineb just videokaadritest.

Kolmandas eksperimendis püüti leida optimaalset enesekindluse lävendit, millele alla jäädes loetakse inimene tundmatuks. Alustati 10% ning lõpetati 25% protsendi juures, kasutades 5% samme. Kõige paremad tulemused saavutati 20% ja 25% lävendite juures.

Mõlemas katses jäi maksimaalne valede ennustuste hulk 5% lähedale. Õigete ennustuste hulk oli 20% lävendi juures maksimaalselt 72% ning 25% lävendi korral oli sama tulemus 68%. Kuna valede ennustuste arvu vähenemine on peaaegu märkamatu, aga õige ennustuste hulk langes 4% siis ei ole lävendi tõstmine 25% õigustatud. Tegemist on hinnanguga, mis võib sõltuvalt lõpprakenduse vajadustest muutuda kuid esialgne optimaalne lävend on 20%.

Kolmandas eksperimendis on jõutud maksimaalse täpsuseni kui õppimispilte on iga inimese kohta 60. Järgnevad lisanduvad pildid ei too suurt muutust õigete ennustuste arvule, samas valede ennustuste hulk kergelt tõuseb. Seetõttu võib pidada optimaalseks 60 pilti iga inimese kohta. Kokkuvõtlikult, kõige paremad tulemused saavutati 60 õppimispildiga iga inimese kohta, kui enesekindluse lävend on 20%.

# 6 Prototüübi võimalikud edasiarendused

Töö käigus valmis rakendusest kõigest põhiülesannet täitev prototüüp. Prototüüpi on võimalik täiendada lisavõimalustega, mis võivad pakkuda kasutajale lisandväärtust.

Esimene edasiarendus võiks olla võimalus muuta tagastatavat faili formaati. Lisaks JSONile saaks valida ka näiteks XML või YAML formaate. See võib muuta rakenduse kasutamise kasutajatele mugavamaks.

Teise arendusena võiks luua võimaluse kasutajale defineerida, milliseid andmeid ta soovib video kohta saada. Rakenduse prototüübis tagastatakse alati kaadri number, ennustatud inimese nimi ning ennustuse enesekindluse tulemus. Edasiarenduses oleks võimalus valida, milliseid nendest väljadest soovitakse. Lisaks võiks olla võimalik tagastada lihtsamas vormis andmeid, näiteks tagastada ainult nimekiri videos tuvastatud inimestest. Selline arendus võimaldaks kasutajal saada ainult need andmed, mis on talle kasulikud.

Kolmas edasiarendus, võiks lubada kasutajal määrata enesekindluse lävendit. Nagu varem mainitud, siis kõige õigem enesekindluse lävendi määr sõltub rakenduse kasutaja konkreetsest eesmärgist ning seetõttu oleks mõistlik kasutajal see ise määrata. Siinkohal oleks oluline eelnevalt kasutajat informeerida, millises vahemikus võiks tema määratud enesekindluse vahemik olla, ning millist mõju see avaldab tagastatud andmetele.

# 7 Kokkuvõte

Töö põhieesmärgiks oli luua näotuvastuse tehniline lahendus videopildist inimeste tuvastamiseks. Töö käigus loodi rakenduse prototüüp, mis võimaldab Youtube videokeskkonnast pärit videotest tuvastada inimesi. Prototüüp saab kasutajalt URL'i ning tagastab JSON formaadis tuvastatud inimesed koos kaadri numbri ning ennustuse enesekindluse tulemusega. Sellega võib lugeda töö põhieesmärgi täidetuks.

Töö keskendus näotuvastuses kasutatud närvivõrgu õpetamisele. Seoses sellega, oli töö eesmärgiks uurida, kuidas mõjutab erinev õppepiltide arv närvivõrgu täpsust ning ennustuste enesekindluse tulemust. Eksperimentidest selgus, et optimaalselt võiks õppeandmestikus olla 60 pilti iga inimese kohta. Vähendades piltide arvu väheneb ka õigete ennustuste hulk, eriti olukorras, kus arvestatakse ka enesekindluse lävendit. Tõstes piltide arvu hakkab tõusma ka valede ennustuste enesekindluse tulemus ning seetõttu suureneb valede ennustuste hulk.

Seoses närvivõrgu õpetamisega oli teine eesmärk võrrelda erineva fotodelt pärit piltidega treenitud mudeli efektiivsust videokaadritest pärit piltidega treenitud mudeli efektiivsusega. Kuna testimiseks kasutati videosi, siis oodatult andis paremaid tulemusi õppeandmestik, mis koosnes videokaadritest pärit pildidest.

Seoses näotuvastuse kasutamisega oli eesmärgiks määrata optimaalne enesekindluse lävend tundmatute inimeste tuvastamiseks. Eksperimentide tulemuste analüüsil leiti, et optimaalne lävend oleks 20%. Sellise lävendiga tuli saadi madal valede ennustuste hulk, alla 5%, samas jäi õigete ennustuste hulk endiselt 70% juurde. Sellega saab lugeda töö eesmärgid täidetuks.

# Kasutatud kirjandus


[1] Kaplan, A., Haenlein, M. Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. – Business Horizons, 2019, 62(1), 15-25. [E-ajakiri] ScienceDirect (28.03.2019)

[2] Campbell, M., Hoane, A. J., Hsu F., Deep Blue – Artificial Intelligence, 2002, 134(1-2), 57-83. [E-ajakiri] ScienceDirect (28.03.2019)

[3] Lu, C., Tang, X. Surpassing Human-Level Face Verification Performance on LFW with GaussianFace – Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, 3811-3819. [Online] https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9845 (28.03.2019)

[4] Srihari S. N., Kuebert E. J. Integration of hand-written address interpretation technology into the United States Postal Service Remote Computer Reader system - Proceedings of the Fourth International Conference on Document Analysis and Recognition, 1997, 2, 892-896. [E-ajakiri] IEEE Xplore (28.03.2019)

[5] The Growing Impact of AI in Financial Services: Six Examples. [Online] https://towardsdatascience.com/the-growing-impact-of-ai-in-financial-services-six-examples-da386c0301b2 (08.05.2019)

[6] Stone, Z., Zickler, T., Darrell, T. Autotagging Facebook: Social network context improves photo annotation - 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008, 1-8. [E-ajakiri] IEEE Xplore (28.03.2019)

[7] Face Recognition in Casinos [Online] https://www.usaonlinecasino.com/blog/face-recognition-in-casinos/ (08.05.2019)

[8] Samuel, A. Some Studies in Machine Learning Using the Game of Checkers. – IBM Journal of Research and Development, 1959, 3(3), 210-229. [E-ajakiri] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5392560 (02.11.2018)

[9] Koza, J. R., Bennet, F. H., Andre, D., Keane, M. A. Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. - Artificial Intelligence in Design '96, 1, 151-170, Springer, Dordrecht, 1996. [E-ajakiri] CiteSeerX (02.11.2018)

[10] Bishop, C. M. Pattern Recognition and Machine Learning. New York : Springer, 2006.

[11] Alpaydin, E. Introduction to Machine Learning. London : MIT Press. 2014

[12] Koit, M., Roosmaa, T. Tehisintellekt. Tartu : Tartu Ülikooli Kirjastus 2011

[13] Zhang Z. A gentle introduction to artificial neural networks. – Annals of Translational Medicine, 2016, 4(19), 370-375. [E-ajakiri] ResearchGate (02.11.2018)

[14] Defferrard, M., Bresson X., Vandergheynst P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering - Neural Information Processing Systems 2016, Spain, Barcelona, 2016, 3837 – 3845.

[15] Scherer, D., Müller, A., Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. - Artificial Neural Networks - ICANN 2010 - 20th International Conference, Proceedings, Part III, Greece, Thessaloniki, 2010, 92-101

[16] Batch normalization in Neural Networks. [Online] https://towardsdatascience.com/the-growing-impact-of-ai-in-financial-services-six-examples-da386c0301b2 (08.05.2019)

[17] Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks - Communications of the ACM, 2017, 60(6), 84-90, [E-ajakiri] ACM Digital Library (03.11.2018)

[18] Schroff, F., Kalenichenko, D., Philbin, J. FaceNet: A unified embedding for face recognition and clustering - 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, 815-823. . [E-ajakiri] IEEE Xplore (28.03.2019)

[19] Face Recognition using Tensorflow – Facenet [WWW] https://github.com/davidsandberg/facenet/wiki (20.01.2019)

[20] Elliot T. The State of the Octoverse: machine learning – The Github Blog, 2019 [WWW] https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/ (20.02.2019)

[21] Flask - Python Software Foundation [WWW] https://pypi.org/project/Flask/ (21.01.2019)

[22] WTForms - Python Software Foundation [WWW] https://pypi.org/project/WTForms/ (21.01.2019)

[23] Youtube-dl – youtube-dl developers [WWW] https://github.com/rg3/youtube-dl/blob/master/README.md (21.01.2019)

[24] OpenCV – OpenCV team [WWW] https://opencv.org/about.html (21.01.2019)

[25] Pandas - Python Software Foundation [WWW] https://pypi.org/project/pandas/ (21.01.2019)

[26] Numpy - Python Software Foundation [WWW] https://pypi.org/project/numpy/ (21.01.2019)

[27] TensorFlow: Large-scale machine learning on heterogeneous systems – tensorflow.org [WWW] https://www.tensorflow.org/about/bib (22.01.2019)

[28] Pedregosa F., Varoquaux, G., Gramfort, A. Scikit-learn: Machine Learning in Python - Journal of Machine Learning Research, 2011, 12(10), 2825-2830. [E-ajakiri] Journal of Machine Learning Research (22.01.2019)

# Lisa 1 – Lõik rakenduses saadud vastusest JSON formaadis


```
{
 "frame": 7,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.6061128130536779
},
{
 "frame": 8,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.42608116282140973
},
{
 "frame": 9,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.46850907784837986
},
{
 "frame": 10,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.3760561150297073
},
{
 "frame": 11,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.3393470368820102
},
{
 "frame": 11,
 "prediction": "Barbi Pilvre",
 "confidence": 0.3115665555479389
},
{
 "frame": 11,
 "prediction": "Eiki Nestor",
 "confidence": 0.27781626169957396
},
{
 "frame": 12,
 "prediction": "Unknown",
 "confidence": 0.04560715985758144
},
{
 "frame": 12,
 "prediction": "Jevgeni Ossinovski",
 "confidence": 0.30860120215397635
},
```

# Lisa 2 – Rakenduse lähtekood

Kasutajaga suhtlemise üksuse lähtekood (user_component.py)

```
1.  from flask import Flask, render_template, flash, request, Response
2.  from wtforms import Form, TextField, TextAreaField, validators, StringField, S
    ubmitField
3.  import pandas
4.  import os
5.  import json
6.  import utils
7.
8.  app = Flask(__name__)
9.  app.config.from_object(__name__)
10. app.config['SECRET_KEY'] = os.urandom(16)
11.
12. class ReusableForm(Form):
13.     url = TextField('url:', validators=[validators.required()])
14.
15. @app.route('/', methods=['GET', 'POST'])
16. def main():
17.     form=ReusableForm(request.form)
18.     if request.method == 'POST':
19.         if form.validate():
20.             URL=request.form['url']
21.             filename = utils.getJSONFileName(URL)
22.             status = resolveStatus(URL)
23.             if status==3:
24.                 outfile = open(filename, 'r')
25.                 return outfile.read()
26.             if status==2:
27.                 flash("Video processing is not yet completed")
28.             if status==1:
29.                 flash("Video added but is not yet processed")
30.         else:
31.             flash('Form validation failed')
32.
33.     return render_template('mainPage.html', form=form)
34.
35. def resolveStatus(url):
36.     filename = "urls.csv"
37.     file_exists = os.path.isfile(filename)
38.
39.     if(not file_exists):
40.         data = {"urls":[url], "status":[1]}
41.         dataFrame = pandas.DataFrame(data=data)
42.         dataFrame.to_csv(filename, index=False)
43.         return 1
44.     else:
45.         file_data = pandas.read_csv(filename)
46.         if set([url]).issubset(file_data.urls):
47.             row = file_data.loc[file_data.urls==url]
48.             return row.status.item()
49.         else:
50.             data = {"urls":[url], "status":[1]}
51.             file_data = file_data.append(pandas.DataFrame(data=data),ignore_in
    dex=True)
```

```
            file_data.to_csv(filename, mode='w', header=True, index=False)
            return 1

if __name__ == "__main__":
app.run()
```

Video töötlemise üksuse lähtekood (video_component.py)

```
import youtube_dl
import cv2
import pandas
from scipy import misc
import os
import time
import predict
import utils

def findUrl(filename):
    file_exists = os.path.isfile(filename)
    if (file_exists):
        file_data = pandas.read_csv(filename)
        if set([1]).issubset(file_data.status):
            rows = file_data.loc[file_data.status==1]
            url = rows['urls'].iloc[0]
            status = rows['status'].iloc[0]
            file_data.at[rows.iloc[0].name, 'status']=2
            file_data.to_csv(filename, mode='w', header=True, index=False)
            return url
    return None

def setStatusComplete(url):
    file_data = pandas.read_csv("urls.csv")
    file_data.loc[file_data['urls']==url, 'status']=3
    file_data.to_csv("urls.csv", mode='w', header=True, index=False)
    return



def deleteImages(image_path):
    files = os.listdir(image_path)
    for i in files:
        os.remove(image_path+"/"+i)
    return


def process(url):
    vid_file =  'videos/video.mp4'
    image_path = 'images'

    if not os.path.exists(image_path):
        os.makedirs(image_path)
    ydl_opts={
        'outtmpl': vid_file
    }
    video=None
    with youtube_dl.YoutubeDL(ydl_opts) as ydl:
       video = ydl.download([url])

    video_capture = cv2.VideoCapture(vid_file)
    video_capture.set(cv2.CAP_PROP_POS_MSEC, 0)
    img_found,image = video_capture.read()
    img_number = 1;
    time = 0;
```

```
    while img_found:
      img_number_zeros = str(img_number).zfill(5)
      img_filename=image_path+"/frame"+img_number_zeros+".jpg"
      cv2.imwrite(img_filename, image)
      image = misc.imread(img_filename);
      img_found,image = video_capture.read()

      if cv2.waitKey(10) == 27:
          break
      video_capture.set(cv2.CAP_PROP_POS_MSEC, time)
      img_number += 1
      time += 500
    video_capture.release()

    predictions = predict.getPredictionsFromImages(image_path)
    JSON_filename=utils.getJSONFileName(url)
    with open(JSON_filename, 'w') as JSON_file:
        JSON_file.write(predictions)

    os.remove(vid_file)
    deleteImages(image_path)
    setStatusComplete(url)
    return


filename = "urls.csv"
url=None
polling = True
while polling:
    url = findUrl(filename)
    if url is not None:
        process(url)
    time.sleep(2)
```

Ennustuste tegemise lähtekood (predict.py)

```
import argparse
import facenet
import os
import sys
import math
import pickle
import json
import tensorflow as tf
import numpy as np
from sklearn.svm import SVC
from scipy import misc
import align.detect_face
from six.moves import xrange

def loadAndAlignData(image_paths, image_size, margin, gpu_memory_fraction):
    minsize = 20
    threshold = [ 0.6, 0.7, 0.7 ]
    factor = 0.709
    with tf.Graph().as_default():
        gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=gpu_memory
_fraction)
        sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options, log_d
evice_placement=False))
        with sess.as_default():
            pnet, rnet, onet = align.detect_face.create_mtcnn(sess, None)

```

```
    nrof_samples = len(image_paths)
    img_list = []
    count_per_image = []
    for i in xrange(nrof_samples):
        img = misc.imread(os.path.expanduser(image_paths[i]))
        img_size = np.asarray(img.shape)[0:2]
        bounding_boxes, _ = align.detect_face.detect_face(img, minsize, pnet, rnet, onet, threshold, factor)
        count_per_image.append(len(bounding_boxes))
        for j in range(len(bounding_boxes)):
                det = np.squeeze(bounding_boxes[j,0:4])
                bb = np.zeros(4, dtype=np.int32)
                bb[0] = np.maximum(det[0]-margin/2, 0)
                bb[1] = np.maximum(det[1]-margin/2, 0)
                bb[2] = np.minimum(det[2]+margin/2, img_size[1])
                bb[3] = np.minimum(det[3]+margin/2, img_size[0])
                cropped = img[bb[1]:bb[3],bb[0]:bb[2],:]
                aligned = misc.imresize(cropped, (image_size, image_size), interp='bilinear')
                prewhitened = facenet.prewhiten(aligned)
                img_list.append(prewhitened)
    images = np.stack(img_list)
    return images, count_per_image, nrof_samples

def findImages(image_path):
    image_paths=[]
    for file in os.listdir(image_path):
       image_paths.append(image_path+"/"+file)
    return image_paths

def getPredictionsFromImages(image_path):
    image_paths=findImages(image_path)
    model_path="Model/1"
    classifier=model_path+"/myclssifier.pkl"
    image_size=160
    seed=7
    margin=44
    threshold=0.2
    gpu_memory_fraction=0.7


    images, count_per_image, nrof_samples = loadAndAlignData(image_paths, image_size, margin, gpu_memory_fraction)
    with tf.Graph().as_default():
        with tf.Session() as sess:
            facenet.load_model(model_path)
            images_placeholder = tf.get_default_graph().get_tensor_by_name("input:0")
            embeddings = tf.get_default_graph().get_tensor_by_name("embeddings:0")
            phase_train_placeholder = tf.get_default_graph().get_tensor_by_name("phase_train:0")

            feed_dict = { images_placeholder: images , phase_train_placeholder:False}
            emb = sess.run(embeddings, feed_dict=feed_dict)
            classifier_filename_exp = os.path.expanduser(classifier)
            with open(classifier_filename_exp, 'rb') as infile:
                (model_path, class_names) = pickle.load(infile)

            predictions = model_path.predict_proba(emb)
            best_class_indices = np.argmax(predictions, axis=1)
            best_class_probabilities = predictions[np.arange(len(best_class_indices)), best_class_indices]

            predictions_data = []
```

```
            k=0
            for i in range(nrof_samples):
                for j in range(count_per_image[i]):
                    if best_class_probabilities[k]>threshold:
                        p = Prediction(i,class_names[best_class_indices[k]], best_class_probabilities[k])
                        predictions_data.append(p)
                    else:
                        p = Prediction(i,"Unknown", best_class_probabilities[k])
                        predictions_data.append(p)
                    k+=1

            predictions_JSON = json.dumps(predictions_data, default=obj_dict, ensure_ascii=False, indent=2)
            return predictions_JSON

def obj_dict(obj):
    return obj.__dict__


      class Prediction:
          def __init__(self, frame, prediction, confidence):
              self.frame=frame
              self.prediction=prediction
              self.confidence=confidence
```

Mudeli laadimise lähtekood (model.py)

```
import os
import tensorflow as tf
import numpy as np
import re
from tensorflow.python.platform import gfile

def prewhiten(x):
    mean = np.mean(x)
    std = np.std(x)
    std_adj = np.maximum(std, 1.0/np.sqrt(x.size))
    y = np.multiply(np.subtract(x, mean), 1/std_adj)
    return y


def load_model(model, input_map=None):
    model_exp = os.path.expanduser(model)
    if (os.path.isfile(model_exp)):
        print('Model filename: %s' % model_exp)
        with gfile.FastGFile(model_exp,'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            tf.import_graph_def(graph_def, input_map=input_map, name='')
    else:
        print('Model directory: %s' % model_exp)
        meta_file, ckpt_file = get_model_filenames(model_exp)

        print('Metagraph file: %s' % meta_file)
        print('Checkpoint file: %s' % ckpt_file)

        saver = tf.train.import_meta_graph(os.path.join(model_exp, meta_file), input_map=input_map)
        saver.restore(tf.get_default_session(), os.path.join(model_exp, ckpt_file))

```

```
33. def get_model_filenames(model_dir):
34.     files = os.listdir(model_dir)
35.     meta_files = [s for s in files if s.endswith('.meta')]
36.     if len(meta_files)==0:
37.         raise ValueError('No meta file found in the model directory (%s)' % mo
    del_dir)
38.     elif len(meta_files)>1:
39.         raise ValueError('There should not be more than one meta file in the m
    odel directory (%s)' % model_dir)
40.     meta_file = meta_files[0]
41.     ckpt = tf.train.get_checkpoint_state(model_dir)
42.     if ckpt and ckpt.model_checkpoint_path:
43.         ckpt_file = os.path.basename(ckpt.model_checkpoint_path)
44.         return meta_file, ckpt_file
45.
46.     meta_files = [s for s in files if '.ckpt' in s]
47.     max_step = -1
48.     for f in files:
49.         step_str = re.match(r'(^model-[\w\- ]+.ckpt-(\d+))', f)
50.         if step_str is not None and len(step_str.groups())>=2:
51.             step = int(step_str.groups()[1])
52.             if step > max_step:
53.                 max_step = step
54.                 ckpt_file = step_str.groups()[0]
55.     return meta_file, ckpt_file
```

Lisanduvate meetodite lähtekood (utils.py)

```
1.  def getJSONFileName(url):
2.      url = url.replace('/', '_').replace(':','_').replace('.', '_').replace('?'
    , '_')
3.      fileName = url+'.json'
4.      return fileName
```