TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu Kolledž

# DUCHENNE'I JA VÕLTSNAERATUSE TUVASTAMINE MASINÕPPE KAUDU

## THE DETECTION OF DUCHENNE AND FAKE SMILES USING MACHINE LEARNING

RAKENDUSKÕRGHARIDUSTÖÖ

Üliõpilane: Henri Seppel

Üliõpilaskood: 178428EDTR

Juhendaja: Ants-Oskar Mäesalu, Rinaldo Rüütli

2023

## AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

04.01.2023

Autor: Henri Seppel

/ allkiri /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

“.....” ..... 202

Juhendaja: .....

/ allkiri /

Kaitsmisele lubatud “.....”.....202

Kaitsmiskomisjoni esimees .....

/ nimi ja allkiri /

# Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina Henri Seppel 21.05.1996

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **DUCHENNE'I JA VÕLTSNAERATUSE TUVASTAMINE MASINÕPPE KAUDU**

mille juhendajad on Ants-Oskar Mäesalu ja Rinaldo Rüütli

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1]Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.

_____(allkiri)

04.01.2023

**Inseneriteaduskond**

# LÕPUTÖÖ ÜLESANNE

**Üliõpilane**: Henri Seppel 178428EDTR
Õppekava, peaeriala: EDTR17/17 - Telemaatika ja arukad süsteemid
Juhendaja(d): Ants-Oskar Mäesalu, Rinaldo Rüütli

**Lõputöö teema**:

**DUCHENNE'I JA VÕLTSNAERATUSE TUVASTAMINE MASINÕPPE KAUDU**

**Lõputöö teema inglise keeles:**

**THE DETECTION OF DUCHENNE AND FAKE SMILES USING MACHINE LEARNING**

**Lõputöö põhieesmärgid**:
1. Andmestiku koostamine päris- ja võltsnaeratustest
2. Masinõppe mudeli treenimine eristama päris- ja võltsnaeratust
3. Tulemuste analüüs

**Lõputöö etapid ja ajakava:**

| Nr | Ülesande kirjeldus | Tähtaeg |
|---|---|---|
| 1. | Lõputöö praktiline osa | 02.12.2022 |
| 2. | Lõputöö kirjalik osa | 04.01.2023 |
| 3. | Lõputöö valmisversiooni esitamine Moodle'is | 04.01.2023 |

**Töö keel:** Eesti keel **Lõputöö esitamise tähtaeg:** 04.01.2023

**Üliõpilane:** …….. ..... 04.01.2023
/allkiri/
**Juhendaja:** …….. ..... 04.01.2023
/allkiri/
**Konsultant:** …….. ..... 04.01.2023
/allkiri/
**Programmijuht:** ……..... 04.01.2023
/allkiri/

*Kinnise kaitsmise ja/või lõputöö avalikustamise piirangu tingimused formuleeritakse pöördel*

# SISUKORD

# EESSÕNA

Lõputöö teema sai valitud juhendaja Ants-Oskar Mäesalu välja pakutud teema "Inimese emotsioonide tuvastamine kaamerapildi alusel" põhjal. Teema tausta uurides jõudis töö autor Duchenne'i naeratuseni ja valis teemaks "uus pealkiri – masinõppe

Töö eesmärk on koostada on ehitada masinõppe mudel, mis suudab eristada päris ehk Duchenne'i naeratust võltsnaeratusest. Töö läbiviimiseks tuli panna kokku andmebaas piltidest. Töö sisu tutvustab masinõpet ja konvolutsioonilist tehisnärvivõrku. Kasutatakse Tensorflow raamistikku ja Keras teeki, et ehitada eel-treenitud mudeli abiga programm, mis eristab pildi pealt kolm elementi: nägu, suu ja silmad, ning on võimeline tegema vahet, kas element kuulub päris- või võltsnaerastusega kokku.

Võtmesõnad: Masinõpe, Duchenne'i naeratus, Konvolutsiooniline närvivõrk, TensorFlow, Keras, CNN, Confusion matrix

# LÜHENDITE JA MÕISTETE SÕNASTIK

| | |
|---|---|
| JPG (JPEG) | Standartne digitaalsete piltide tihendus mis lubab neid tõhusalt salvestada [1] |
| Elemendid | Nägu, suu ja silm |
| Faili/kausta tee | *Path*<br>Faili kausta asukohani arvutis jõudmise tee |
| CSV | Comma separated values<br>Komaeraldusega väärtused Porditav failivorming, kus andmebaasikirjed on üksteisest eraldatud komadega. Selles vormingus on iga rida üks kirje, mille väljad on üksteisest komadega eraldatud. Komade järel võib olla suvaline arv tühikuid ja/või tabeldusmärke (tab character), sest neid ignoreeritakse. Kui väli ise sisaldab koma, siis peab kogu väli olema ümbritsetud jutumärkidega. |
| RGB | Põhivärvused punane, roheline ja sinine (Red–Green–Blue) |
| CNN | Convolutional Neural Network (Konvolutsiooniline Närvivõrk) |

# SISSEJUHATUS

55% inimese emotsioonidest väljendub näoilmete kaudu [2], seega on oluline näo väljendusi õigesti lugeda. Emotsioonid jagunevad baasemotsioonideks: viha, kurbus, hirm, üllatus, vastikus ja rõõm [2]. Rõõmu kõige äratuntavam tunnus on naeratus. Pärisnaeratus ehk Duchenne'i naeratus hõlmab endas suunurkasid ülespoole liigutavate lihaste kokkutõmbeid (*zygomaticus major muscle*) ning erinevalt võltsnaeratusest põsesarnade tõusu ning silmakurdude teket (*orbicularis oculi muscle*) [2]. Emotsioone saab jagada makro- ja mikroväljendusteks. Makroväljendused on lihtsasti tuvastatavad, hõlmavad tervet nägu ning kestavad kuni 4 sekundit [3]. Mikroväljendused on emotsioonide väljendid, mis annavad täieliku pildi emotsioonist, kuid kestavad vaid murdosa nende tavapärasest kestusest [3]. Need kestavad niivõrd lühikest aega, et jäävad tihti inimsilmale märkamatuks [4]. Mikroväljendusi kasutatakse emotsioonide varjamiseks või nende kohta valetamiseks [5]. Naeratus on kõige levinum viis varjatud emotsioonide maskeerimiseks [3]. Seega, on Duchenne'i naeratuse tuvastamisel kriitiline tähtsus, kuna selle ülepingutatud vormi seostatakse valetamisega [3].

Valetamine on oluline aspekt õigsussüsteemis. Inimeste hinnangud selle kohta, kas valetatakse või mitte, on sagedamini pigem valed kui õiged [6]. Varasemalt on õigussüsteemides kasutatud valetamise tuvastamiseks polügraafe, kuid hiljem on selgunud, et nendel puudub tugev teaduslik alus, mis tõestaks nende tulemuste õigsust [7]. Üha enam uuringuid tõstab esile asjaolu, et inimesed suudavad sotsiaalsetes olukordades tekitada võltsitud emotsioone, mis ei ole teiste inimeste poolt tuvastatavad [8, 9, 10]. Siinkohal on võimalik võtta appi arvutiprogrammid, mis on suutelised sirvima läbi suurtes kogusestes andmebaase võlts naeratustest ja päris naeratustest, ning suudavad anda objektiivse hinnangu, kas emotsioon on ehtne.

Masinõpe on kiiresti kasvav valdkond, millel on potentsiaali revolutsioneerida lai valik tööstusharusid ning mis leiab rakendusi kõigis teadusvaldkondades. Kasutades algoritmide võimet masinõppes, on võimalik leida kasulikku informtatsiooni väga keerulistest andmestruktuuridest. Seda oskust saab kasutada ära andmete esindamisel ja hindamisel [11].

Kuna inimsilmal on keeruline teiste inimeste mõistmiseks olulisi mikroemotsioone märgata, on käesoleva töö eesmärk ehitada programm, mis treenib masinõppe mudelit eristama võlts- ja pärisnaeratust, analüüsima tulemusi ja võimaldama kasutajal laadida üles pilt naeratusest ning seejärel saada mudelilt vastus, kas tegu on võltsnaeratusega või mitte.

Töös on masinõppe mudeli ja närvivõrkude implementeerimiseks kasutaud vabavara TensorFlow ning Keras. Andmete visualiseerimiseks on kasutatud teeke Matplotlib ja Pillow. Töö on kirjutatud Python programeerimiskeeles.

Koodibaas ning pildid on avalikustatud GitHub'i leheküljel:
https://github.com/HenriSe/DuchenneNaeratus

Lõputöö on kirjutatud eesti keeles ning sisaldab 5 peatükki, 30 joonist ja 6 tabelit.

# 1 DUCHENNE'I NAERATUS

Naeratust saab tuvastada põhiliselt kolme aspekti järgi: suu, silmad ja näo kuju (edaspidi elemendid). Duchenne'i naeratus, tuntud ka kui tõeline naeratus, on näoilme, mida iseloomustab silmaümbruse lihaste kokkutõmbumine ning suunurki ülespoole tõmbavate lihaste aktiveerumine. See on nime saanud 19nda sajandi Prantsusmaa neuroloogi Guillaume Duchenne'i järgi [12]. Duchenne'i naeratust vastandatakse sageli võltsnaeratustega, mille puhul tõusevad ainult suunurgad.

Tõstes suunurkasid tõusevad ka põsesarnad, mida liigutavad lihased nimega *Zygomaticus major* [2]. Liigutades neid lihaseid, muutub näo kuju ja üleüldine ilme. Duchenne'i naeratust kirjeldatakse kui positiivset emotsiooni, mis näitab ehtsust ja usaldusväärust. Võltsnaeratust kirjeldatakse kui viisakat ja sõbralikku ilmet, kuid mis ei näita välja tegelikku emotsionaalset seisundit ning võib viidata petmise kavatsustele [5].

# 2 MASINÕPE

Masinõpe on arvutiteadus haruga, mille eesmärk on anda võimalus arvutitel "õppida" ilma neid otseselt selleks programeerides, selle jaoks treenivad masinõppe algoritmid mudeleid näidisandmete põhjal. Arvutid õpivad parandades enda sooritust, täites mingit kindlat ülesannet ning kogudes "kogemusi". Praktikas tähendab see andmete sobitamist soovitud parameetridesse [13].

Üks peamiseid masinõppe eeliseid on omadus lubada arvutil, täites mingit ülesannet, oma sooritust aja jooksul parandada. Näiteks võib masinõppe algoritmi treenida lugema mustreid meditsiinilistes andmetes. Mida rohkem algortim töötleb andmeid läbi, seda täpsemaks see muutub haiguste diagnoosimisel [14]. Lisaks on masinõppe üks olulisi aspekte see, et see võimaldab arvutitel täita ülesandeid, mis võivad inimestele olla liiga keerulised või liialt aeganõudvad. Selleks võib olla näiteks aktsiaturu andmete läbi töötamine ja ennustamine [15], või siis saab masinõpe olla abiks piltide ja videote klassifitseerimisel [16].

Tehisnärvivõrk (*Artificial neural network*) on masinõppe süsteem, mis jäljendab struktuuri ja funktsionaalsuse poolest aju [13]. Võrk koosneb omavahel seotud töötlemisüksustest, mida nimetatakse neuroniteks (*neurons*). Ehitades arvutis algoritme, mis kopeerivad päris neuronite tööd, saame luua tehisliku raamistiku, mis "õpib" lahendama probleeme. Õppimine toimub kui mudeli neuronile või neuronite grupile antakse sisend, mida ta "kaalub" ja annab vastavalt etteantud arvutustele mingi

tulemi ehk väljundi. Kui väljundite summa ületab määratletud lävendi, siis on mõõdetavaks tulemuseks "1" ehk päris, aga kui lävendit ei ületata on tulemuseks "0" ehk võlts [17]. Süsteem, kus väljundiks saab olla 2 valikut, kutsutakse binaarseks süsteemiks. Masinõppes kasutatakse binaarset klassifikatsiooni mudelite puhul, mille ülesanne on sisendit klassifitseertida kahe kategooria vahel. Neuronid on omavahel organiseeritud kihtideks, mis on ühendatud nendevaheliste "aksonitega", sarnaselt nagu närvirakkuide vahel on telgniidid [13]. Tüüpiliselt koosneb mudel sisendi- ja väljundikihist ning vahepealsetest "varjatud" kihtidest (*hidden layers*). Keskmised ehk varjatud kihid koosnevad lävendiga ühikutest, mis teostavad sisendite osalise klassifitseerimise ja saadavad oma tulemused järgmistesse kihtidesse. Selliseid mudeli võrke nimetetakse mitmekihilisteks võrkudeks või pärilevivõrkudeks (*feed-forward network*) [13].

Esitusõpe (*representation learning*) on masinõppe valdkond, kus süsteemi eesmärk on sisendandmetele omaduste leidmine ja nende klassifitseerimine. Sügavõpe on (*deep learning*) masinõppe valdkond, mis kasutab esitusõpet ja tehislikke närvivõrke. Sügavõpe erineb selle poolest, et sisendandmeid kujutavaid omadusi muudetakse kontseptuaalsemaks. Masin suudab tunnustada objekti põhilisi omadusi võttes arvesse ka erinevaid variatsioone [18]. Näiteks, kui mudel on õppinud ära klassifitseerima tavalisi tomateid, siis sügavõppega oskab masin ka ära tunda kollaseid või ebaühtlase kujuga tomateid.

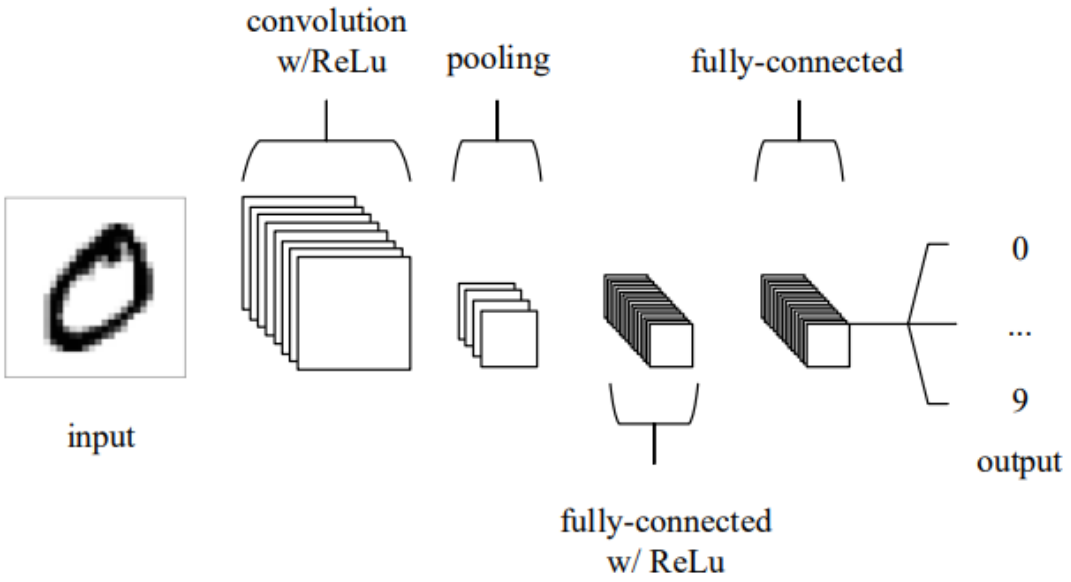
Konvolutsiooniline võrk (*convolutional neural network - CNN*) on tehisnärvivõrgu liik, kus neuronid optimeerivad iseennast õppimise käigus ja on optimeeritud töötlemaks andmeid, mis esinevad võrgutaolise topoloogiaga, nagu näiteks pilt. CNN on osa sügavõppest, mis spetsialiseerub mustrite tajumisele [19]. Alates esimestest näidetest 1990ndate alguses, on konvolutsioonilised tehisvõrgud olnud põhilised tööriistad masinõppes piltide klassifitseerimises [20].

Konvolutsioonilise võrgu eripära on, et kindlad kihid võrgu sees sisaldavad filtreid (*kernel*), mida kasutatakse mustrite leidmisel. Joonisel 2.2 on toodud välja, kuidas kernel võrdleb ja liigub mööda sisendi võrestikku ning edastab väljundväärtuse. Kui jaotada pilt võrestikuks, siis võib filterit kujutada ette kui 3 x 3 plokki, mille sees saame täpsustada, millist mustrit me otsime. Filter liigub mööda võrestikku edasi ja võrdleb, kui sarnane on pildi peal olev osa filtri mustriga. Vastavalt sarnasusele antakse sellele asukohale arvuline väärtus. Tulemuseks saame massiivi numbritest, mis ütleb kui sarnane on pilt otsitavale. Erinevaid objekte otsivaid filtreid võib ühes kihis olla mitu ning kui võtta kokku kõikide filtrite tulemused ja need kombineerida (*pooling*), tekib mudelile parem arusaam, mida see pilt sisaldab. Võrgu järgmistel kihtidel otsivad uued filtrid keerukamaid objekte või võivad täita abstraktsemaid ülesandeid [17][19]. Näiteks

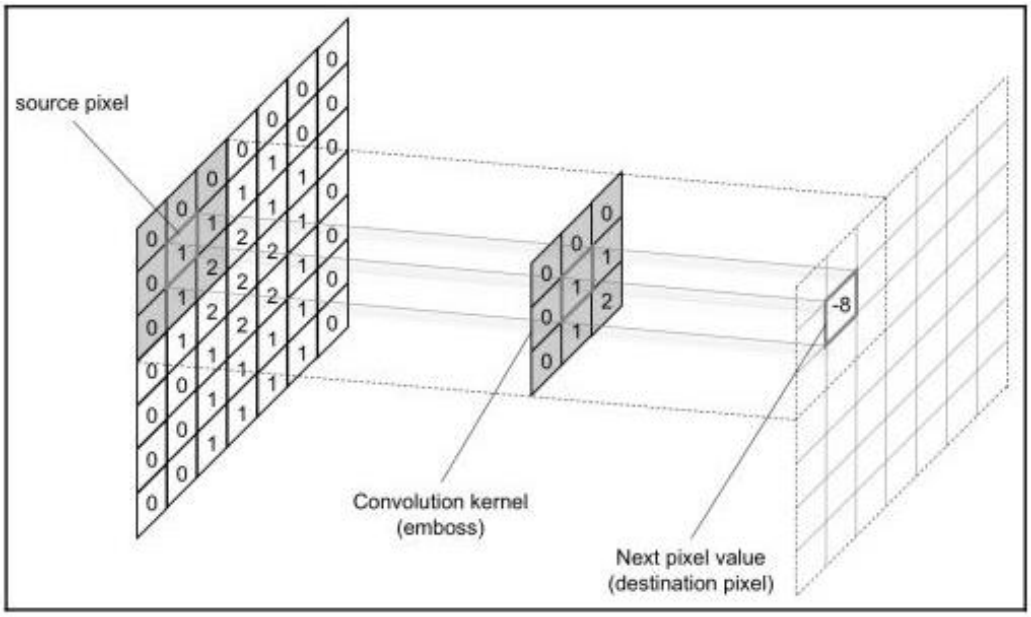
otsitakse esimese kihi peal ruudukujulist objekti, järgmise kihi peal akent ja viimase kihi peal maja. Joonisel 2.1 on näha konvolutsioonilise võrgu ehitust, mis koosneb sisendkihist, sisemistest konvolutsioonilistest kihtidest, *pooling* kihist, täielikult ühenduses olevast *Dense* kihist ReLu aktiveerimisfunktsiooniga ja väljundkihist.

Tüüpiliselt koosneb konvolutsiooniline võrk neljast erinevast unikaalsest osast [19]:

1. Sisendkiht (*input layer*) – sisaldab endas sisendi väärtuseid
2. Konvolutsioonikiht (*convolutional layer*) – toimib aktivatsioonikihina ehk kiht, mis rakendab filtreid andmete peale ja arvutab selle põhjal väljundväärtuse. Väljund on tunnuste kaart, mis muudab sisendandmeid ja tõstab esile otsitavad iseärasused.
3. Ahenduskiht (*pooling layer*) – selle kihi eesmärk on tunnuste kaardi suuruse vähendamine, mis vähendab mudeli treenimiseks vajalikku arvutuslikku jõudlust ja parandab mudeli üldistamise oskust.
4. Täissidus kiht (*fully-connected* / *dense layer*) – kihi ülesanne on võtta kokku väljatoodud tunnused ja nende põhjal ennustuse tuletamine.




Joonis 2.1 – Konvolutsioonilise võrgu ehitus [19]



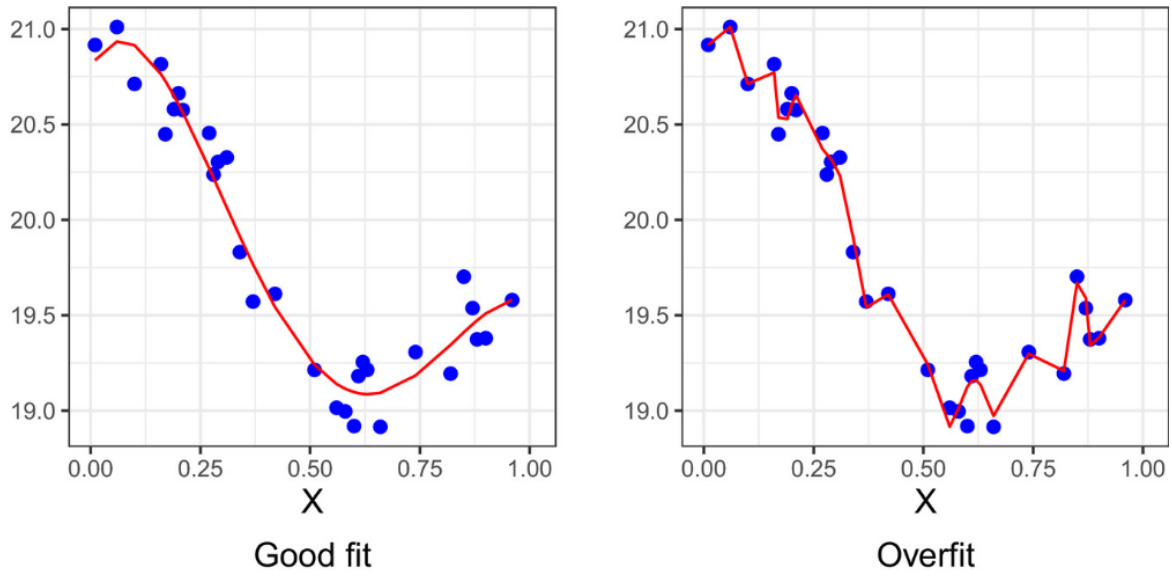

Joonis 2.2 – Filter võrldeb pildi peal olevat asukohta ja annab sellele väärtuse [19]

Üks levinumaid treenimisviise masinõppes on juhendatud õpe (*supervised learning*), kus andmestik on varasemalt sildistatud õigetesse kategooriatesse. Treenimise eesmärk on treenida mudelit nii, et ta suudaks võimalikult täpselt tuletab uute, veel nägemata, andmete väärtuseid. Masinale söödetakse ette komplekt andmeid, mille kategooriad on teada ning seejärel hinnatakse, kui täpne on mudeli hinnang eeldatud tulemusele. Seejärel optimeerib mudel oma parameetreid, et järgmisel korral oleksid ennustused täpsemad. Nii läbitakse mitu treenimise iteratsiooni, senikaua kuni mudel suudab andmeid klassifitseerida võimalikult täpselt [18].

Populaarne optimeerimise algoritm on stohhastiline gradientlaskumine (*stochastic gradient descent - SGD*), mis otsib selliseid parameetreid, mis vähendavad võimalikult palju mudeli kahjufunktsiooni (*loss function*). Kahjufunktsioon on skalaarne väärtus, mis esindab mudeli ennustuste kvaliteeti. Kahjufunktsioon sümboliseerib erinevust ennustatava tulemuse ja tegeliku tulemuse vahel ning seda kasutatakse juhina mudeli optimeerimise suunamiseks. Kahjufunktsiooni gradient näitab mudeli suunda, millises suunas peaks mudeli parameetreid muutma, et vähendada kahju. Treenimise andmed jaotatakse väikesteks partiideks ning mudeli parameetreid kohandatakse peale igat partiid vastavalt kahjufunktsiooni gradiendile. Kui treenimise käigus on saavutatud soovitud täpsus, kontrollitakse mudelit sooritust kontrollandmete partiiga, et mõõta mudeli tulemust väljaspool treeningandmeid. Sellise kontrollimise eesmärk on anda mudeli sooritusele erapooletu hinnang ja testida selle võimet esitada uue andmestikuga võimalikult häid tulemusi [18]. Juhendatud õppe vastand on juhendamata õpe (*unsupervised learning*), kus algoritm üritab iseseisvalt leida andmete naturaalset seost ja grupeeringut ilma kõrvalise abita [13].

Tensorflow on populaarne avalik tarkvara raamistik, mida kasutatakse sügavõppes. Tarkvara on välja arendanud Google [21]. Raamistik on mõeldud mudelite treenimiseks ja ehitamiseks, kasutades erinevaid täiendavaid tööriistu, nagu näiteks Keras. Kerase teek käitub kui Tensorflow liides, millega saab ehitada masinõppe mudeleid. Kerases kirjutatud kood kompileeritakse Tensorflowsse, mille fookuseks on mudeli treenimine. Kerase üheks eeliseks on võimalus kasutada suure andmestiku peal eeltreenitud mudeleid. Neid mudeleid saab lisada ehitatavasse projekti lähtepunktina, kiirendades treenimise protsessi. Eeltreenitud ja teadlikud mudelid on kasulikud, kui arendajal puudub suur andmestik, kuna aitavad mudeli konkreetse ülesande täitmiseks ümber häälestada [22].

Üks suurimaid ja levinumaid probleeme masinõppes on ülesobitamine (*overfitting*). See võib juhtuda siis, kui tehisnärvivõrku tekib liiga palju konkreetseid jälgitavaid parameetreid ehk kaale (*weights*). Need kaalud määravad, kui tugevad on neuronite vahel olevad ühendused ehk milliseid parameetreid jälgitakse rohkem ja milliseid vähem. Kui kaalud muutuvad liialt "raskeks", siis on mudel ületreenitud kindla andmestiku suhtes ja jälgib ülemääraltsuure tähelepanuga andmestiku kindlaid aspekte, mistõttu on mudeli sooritus uute andmestike testimisel madal [23]. Joonisel 2.3 on näha, kuidas ületreenimise korral liigub mudel üldistamise asemel mööda liialt konkreetset teekonda.




Joonis 2.3 – võrdlus hästi treenitud ja ületreenitud mudeli vahel [24]

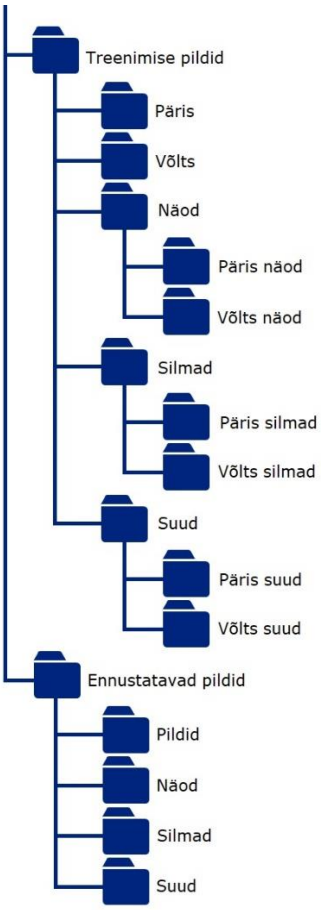
# 3 METOODIKA

Kuna töö autorile teadaolevat pole kindlat viisi, kuidas masinõppes adekvaatselt eristada Duchenne'i naeratust, siis tuli probleemile läheneda praktiliselt ning tuginedes kirjandusele. Duchenne'i puhul muutuvad näo juures kolm põhilist aspekti: näo kuju, suu ja silmad [2]. Mitte-Duchenne'i ehk võlts naeratuse puhul muutvad samuti näo juurs need kolm elementi, kuid mitte samamoodi nagu Duchenne'i naeratuse puhul, põhiliselt silmade erinevus [2]. Tuginedes nende kolme aspekti muutumisele, saab ehitada kolm eraldi masinõppe mudelit, millest igaüks on treenitud jälgima ainult ühte elementi. Kui võtta nende kolme mudeli ennustatavad tulemused ühe ja sama pildi kohta, kuid arvestades seda, et nad on jälginud sama pildi peal erinevaid elemente ning andnud iseseisvalt elemendile tulemuse, on võimalik võrrelda neid kolme tulemust ning anda pildile nende põhjal lõplik tulemus (päris või võlts). See kas pilt on päris või võlts, otsustatakse häälteenamusega.

## 3.1 Andmestik

Kuna töö autorile teadaolevalt pole olemas spetsiifilist võltsnaeratuse piltide andmebaasi, siis oli vajalik pildid andmebaasi jaoks iseseivalt otsida. Piltide otsimiseks kasutati põhiliselt Google otsingumootorit. Hetkel on andmebaasis 200 käsitsi märgistatud pilti – pooled on päris naeratusega ja pooled võlts naeratusega. Andmetöötluses kasutatakse pildil leiduvat kolme elementi: nägu, silmad ja suu. Analüüsides neid elemente, saame määrata kindlaks naeratuse tõepärasuse [3].

Piltide andmestik on jaotatud kahe kausta vahel laiali: treenimise pildid ning ennustatavad pildid. Eelnevalt kategoriseeritud treenimise pildid on omakorda jaotatud kaustadesse: "Päris" ja "Võlts". Lisaks on nende kahe kausta kõrvale lisatud tühjad kaustad järgnevate nimedega: "Näod", "Silmad" ja "Suud". Kaust "Näod" sisaldab veel kahte tühja kausta nimedega: "Päris näod" ja "Võlts näod". Samad kaustad on lisatud ka "Silmade" ja "Suude" kaustadesse vastavate nimedega. Kaustade hierarhia on toodud välja joonisel 3.1.

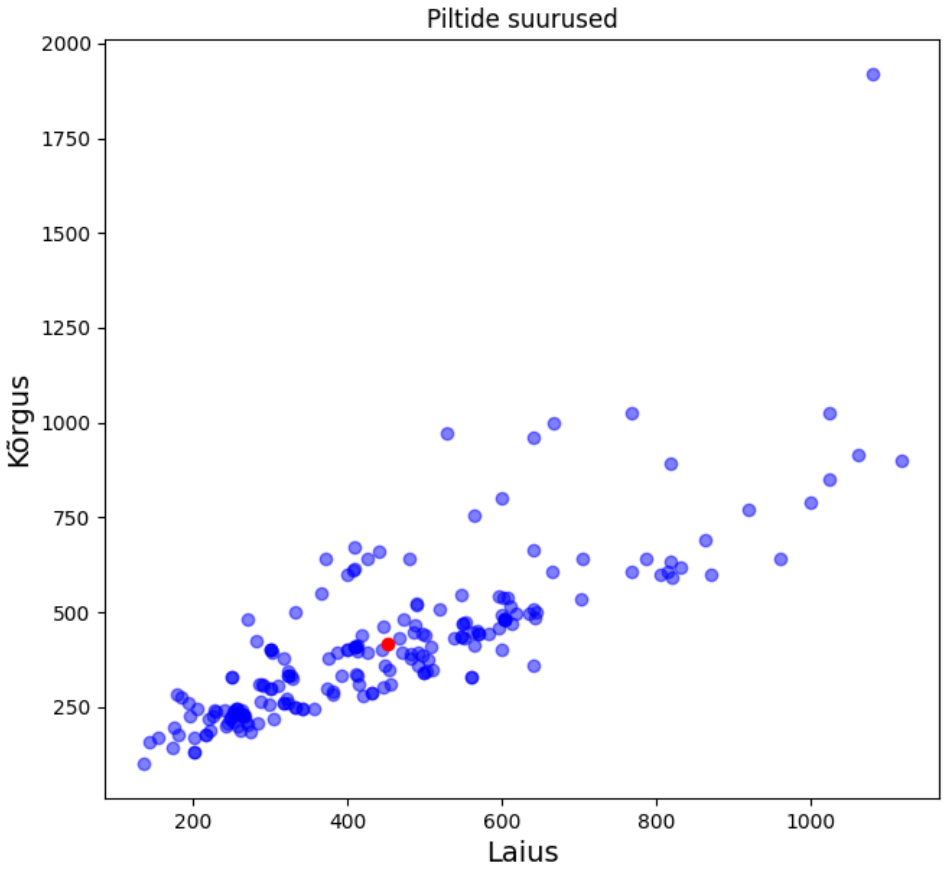



Joonis 3.1 - Kaustade hierarhia arvutis

Programm loeb kahest kaustast "Võlts näod" ja "Päris näod" pilte ning tuvastab nende pealt otsitavad elemendid: nägu, silmad ja suu. Elemendid salvestatakse eraldi jpg failidena nende vastavatesse kaustadesse. Järgnevalt koostab programm elementide piltidest andmeraamistikud. Andmeraamistike abil treenitakse välja kolm erinevat masinõppe mudelit – eraldi nägude, silmade ja suude põhjal.

Kaustast "Ennustused" loeb programm pildid ning jaotab need samuti kolmeks elemendiks eraldi kaustadesse. Treenitud mudelid kategoriseerivad antud elemendid kas tõeseks (tulemus – 1) või võltsiks (tulemus – 0). Nende tulemuste põhjal koostatakse andmeraamistik, kus kuvatakse ennustatava pildi lõplik tulemus.

Mudeli ehitamise üks etapp on andmestikus olevate piltide universaalse suuruse määramine. Kasutades failis *andmed.py* funktsiooni *piltide_suurused* saame visualiseerida (Joonis 3.2) kõikide andmestikus olevate piltide suuruseid. Punane täpp näitab kõikide piltide keskmist kõrgust ja laiust, milleks on 451,8 x 417,8 pikslit. Mudelis treenitavate piltide suurusteks sai valitud 300 x 300 vastavalt treenimiseks kasutatava arvuti jõudlusele.




Joonis 3.2 – Andmestikus olevate piltide suuruste visualisisatsioon kasutades funktsiooni piltide_suurused

## 3.2 Eel-treenitud mudelid

Kerase teegis on võimalik kasutada mitmeid eeltreenitud mudeleid. Neist levinumad mudeli moodulid on: *ResNet, DenseNet, InceptionNet, XceptionNet, VGG19 ja VGG16* [25]. Allpool on toodud välja erinevate eel-treenitud mudelite treenimisandmete graafikud. Mudelite treenimisel varieerub epohhide arv, kuna kasutati Kerase meetodit *EarlyStopping*, mis kujutab endas mudeli treenimise enneaegset lõpetamist, kui on näha ületreenimise märke. Kui treenimisandmete kahju väheneb aga valideerimisandmete kahju hakkab suurenema, siis treenimisprotsess lõpetatakse ning tagastatakse selle hetke mudeli parameetrid [26]. *EarlyStopping* meetodile kaasa antud *patience* väärtus on viis ehk kui üle viie epohhi ei ole muutust toimunud, lõpetatakse treenimine.

Graafiku peal välja toodud tähistused:

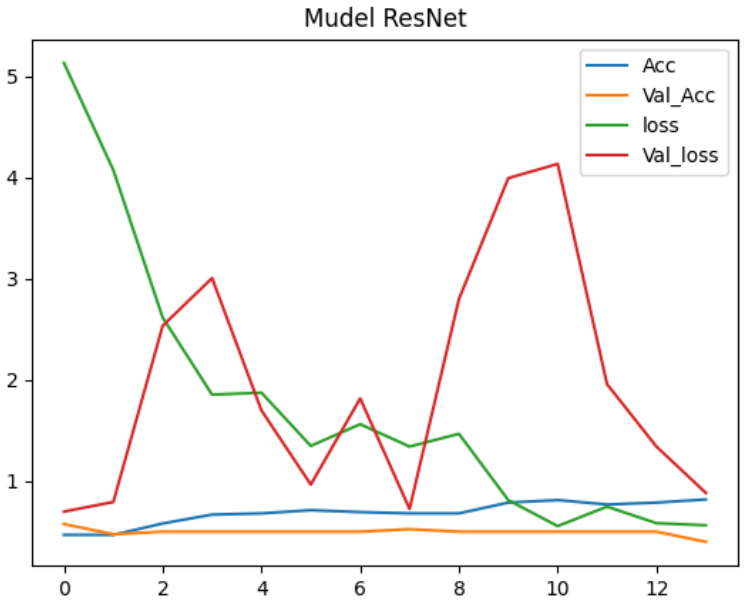
Acc (*Accuracy*) – treeningandmete täpsus

Loss (*Loss*) – treeningandmete kahju

Val_Acc (*Validation Accuracy*) – valideerimisandmestiku täpsus

Val_loss (*Validation Loss)* – valideerimisandmestiku kahju

1. ResNet152 (*Residual Network*) on sügavõppe närvivõrk, mis koosneb 152 kihist. Mudeli arhitektuuri eesmärk on minna närvivõrgu õppega sügavale. Mudeli kihtide vahel on jääkühendused, ehk iga kiht ei ole mitte otseses ühenduses ainult oma eelmise kihiga, vaid ka kihtide enne seda. Selline ehitus on võimalik, kui kasutatakse peale igat konvolutsioonilist kihti *BatchNormalization* kihti. *BatchNormalization* kiht ühsustab andmed enne nende edasi andmist [25].

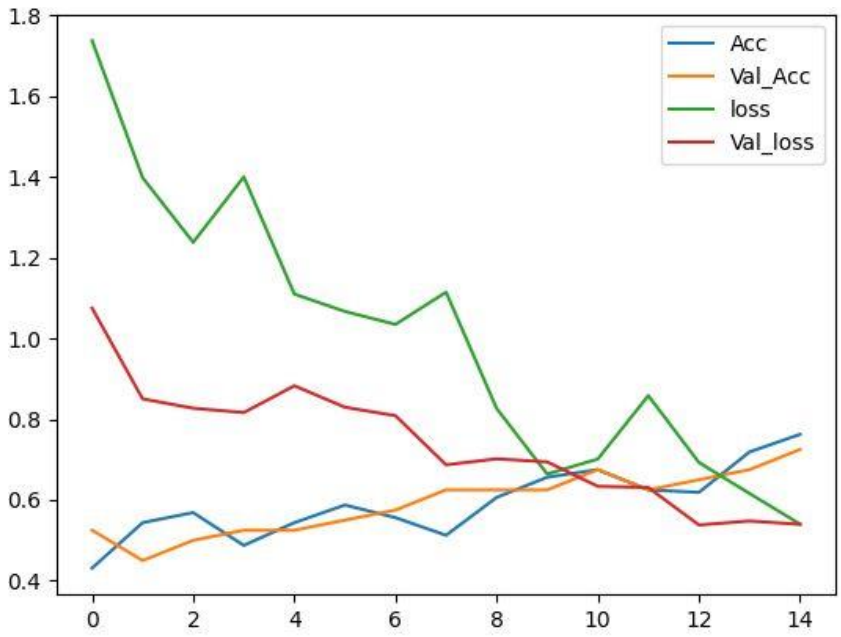



Joonis 3.3 - Mudeli ResNet treenimise graafik

2. DenseNet201 arhitektuuri on iga kiht ühendatud kõikide teiste kihtidega. Tänu sellele on vaja aksutada vähem filtreid ja see hõlbsustab tööd väiksemate andmestikega [25]

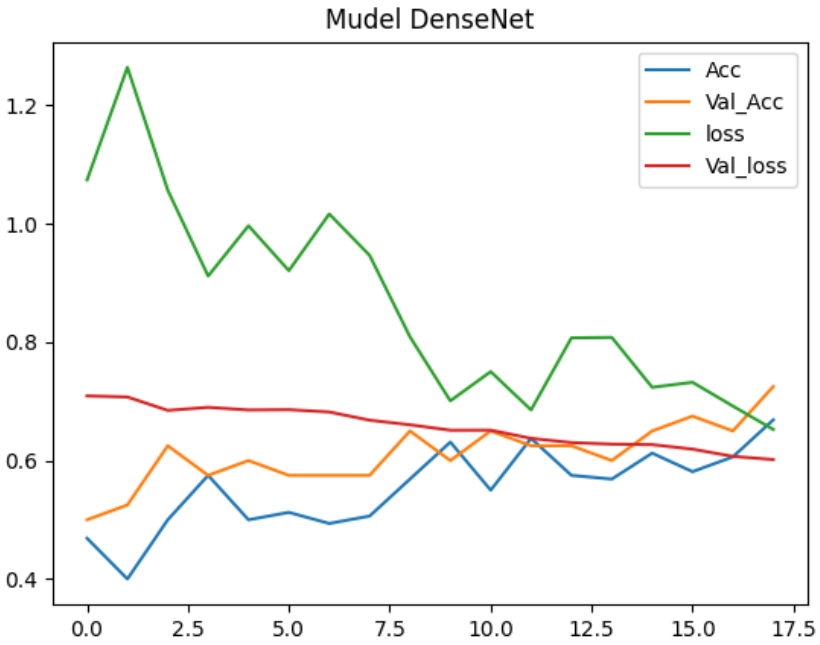

Joonis 3.4 – Mudeli DenseNet201 treenimise graafik

3. InceptionNetV3 on sügavõppe närvivõrk, mille eesmärk on minna sügavale laia võrguga. See saavutatakse kasutades üksteisega kohakuti olevaid kihte, milles igasühes on erinev filter. Sellist arhitektuuri eelistatakse sügavates mudelites, kuna võrk sisaldab endas vähem parameetreid mis vajavad treenimist [25].



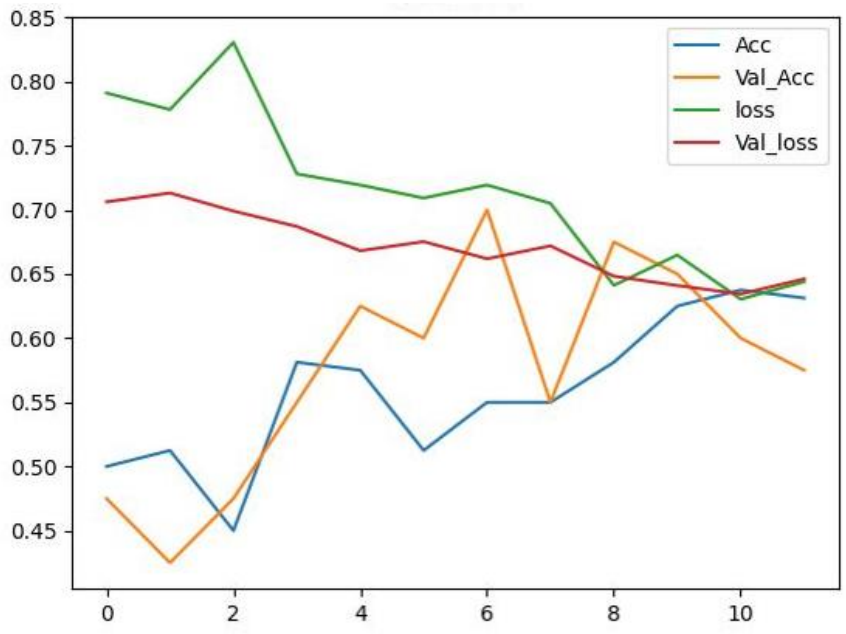

Joonis 3.5 – Mudeli InceptionNetV3 treenimise graafik

4. XceptionNet on InceptionNeti uuendus. Nende erinevus seisneb sellest, et XceptionNet vajab vähem arvutuslikku jõudlust, kuna mudeli sees teostatakse *Depthwise Separable Convolutional* arvutusi [25].



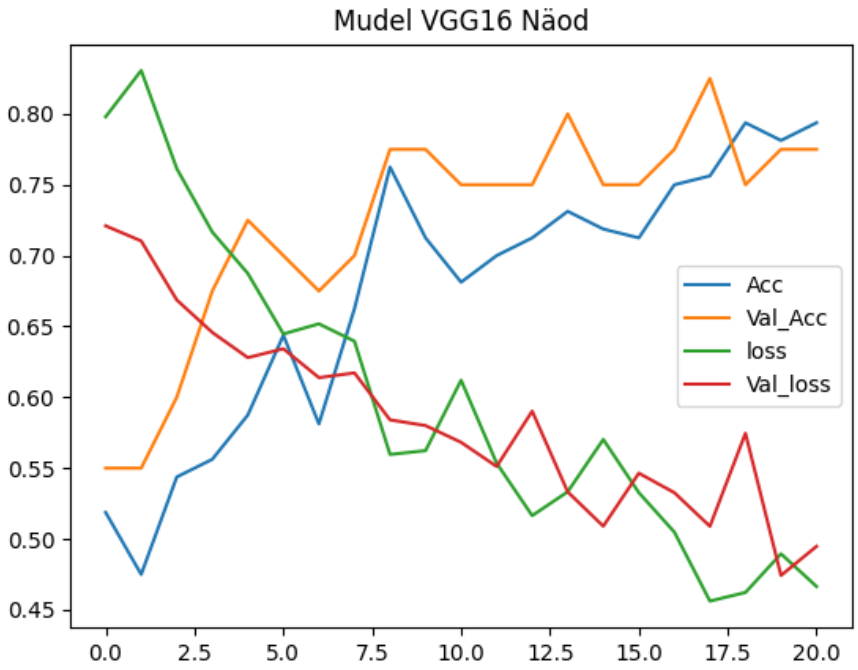

Joonis 3.6 – Mudeli XceptionNet treenimise graafik

5. VGG19 (*Visual Geometry Group*) on CNN arhitektuur, mis on otsekohene ja ehituslikult lihtne, kuid vaatamata sellele on suuteline saavutama paremaid tulemusi kui mitmed teised keerulisema arhitektuuriga mudelid. Numbrid nimetuse lõpus viitavad mudeli sees olevatele kihtidele. Nende hulgast on VGG16 ja VGG19 kõige populaarsemad. Kõik filtrid VGG sees on suurusega 3x3, kuna nad katavad peaaegu sama suure ala kui 5x5 filter, aga nõuavad palju vähem arvutsulikku jõudlust. Kaks 3x3 filtrid on vähem nõudlikumad kui üks 5x5 filter. VGG arhitektuurid on kasulikud väikeste andmestike puhul [25].




Joonis 3.7 – Mudeli VGG19 treenimise graafik

6. VGG16 on sarnase arhitektuuriga kui VGG19, kuid sisemised kihte on 16 [25].




Joonis 3.8 – Mudeli VGG16 treenimise graafik

Mudelid on treenitud nägude andmete peal, kuna selle elemendi treenimise tulemused näitasid kõige paremaid tulemusi ja sujuvamat õppimiskurvi graafiku peal.

. Mudeli valimisel jälgiti kolme omadust:

1. Acc (*Accuracy*) ja Val_Acc (*Validation Accuracy*) liiguvad enam-vähem koos ja on sarnase kujuga.
2. Loss (*Loss*) ja Val_loss (*Validation Loss)* liiguvad enam-vähem koos ja on sarnase kujuga.
3. Acc ja Val_Acc näitaksid esimesed pooles tõusu ning see järel sujuvat stabiliseerumist ning Loss ja Val_loss näitaksid alguses langust ning seejärel sujuvat stabiliseerumist

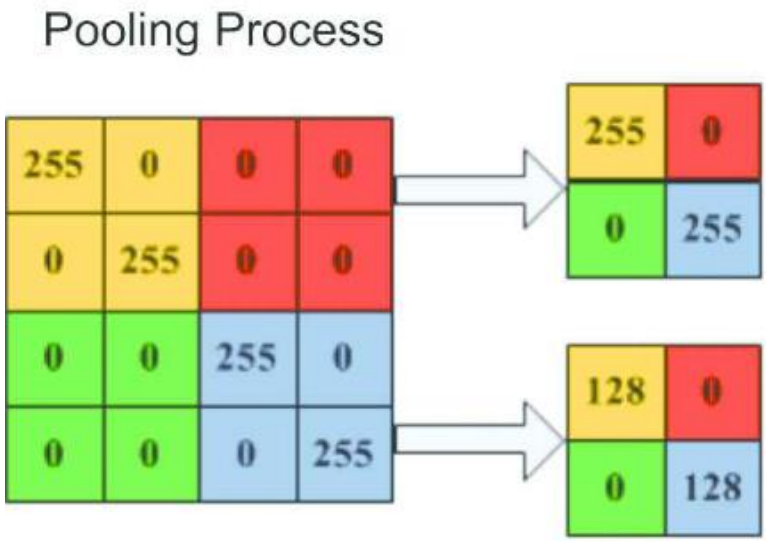
Kuna otsitavad omadused väljuendusid kõige rohkem mudeli VGG16 juures, otsustati valida selle eel-treenitud mudeli kasuks.

## 3.3 Mudeli kompileerimine

Kerase teegis on võimalik kasutada "Sequential" klassi, mille sisse on võimalik lisada erinevaid kihte mudeli treenimiseks. Mudelis on kasutusel järgnevad kihid:

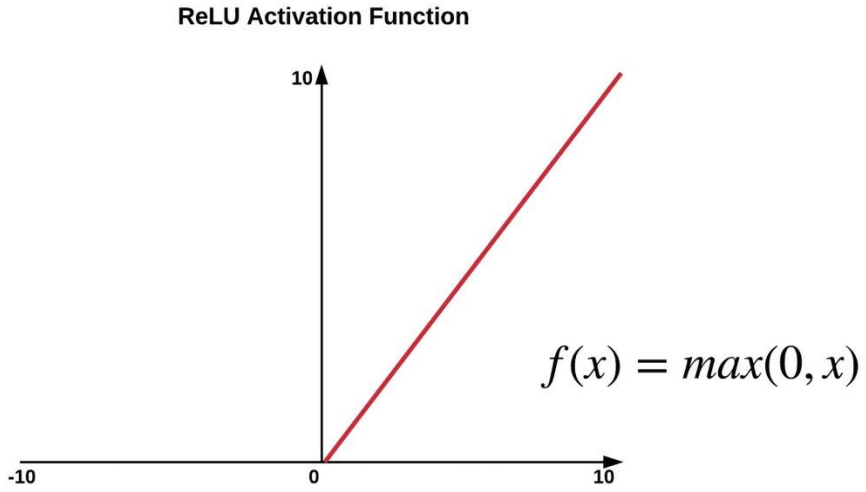
Esimene kiht mudelis on sisendkiht (*input layer*), mis võtab vastu sisendandmed. VGG16 mudelis on viis blokki, millest igaüks koosneb konvolutsioonilistest kihtidest ja ahenduskihtidest. Kokku on 13 konvolutsioonilist kihti ja 5 ahenduskihti.

Ahendamine (*pooling*) on protseduur, mis vähendab sisendi mõõtmeid tuues sealt välja ainult kõige tähtsama info. Tavaliselt kasutatakse sellist tehnikat peale konvolutsioonilises kihtides tehtud arvutusi, et parandada mudeli arvutuslikku jõudlust ja arendada mudeli võimet üldistada [27]. Peale blokke on lisatud (*GlobalMaxPooling2D*) kiht, mis teostab peale konvolutsiooniliste kihtide plokke maksimaalsete väärtuste ahendamise (*max pooling*) arvutusi üle terve sisendi mõõtmete. Protsessi on näha joonise 3.9 peal, kus toimub maksimaalsete väärtuste ning keskmiste väärtuste ahendamine. Kasutades lõpus ahenduskihi *Max* versiooni, säilitatakse mudeli kõige peaasjalikumad tunnusjooned.




Joonis 3.9 – Maksimaalsete väärtuste ahendamine (Max Pooling) protsess [28]

Järgnevalt on lisatud Dense kiht, mis on ühenduses (*fully-connected*) kõikide eelnevate kihtidega. Dense kiht on lisatud , kuna see sisalda kõiki eelnevalt konvolutsioonilistest kihtidest õpitud kaale, mille põhjal sooritatakse lõplik ennustus. Kihile on lisatud aktiveerimisfunktsiooniks *Rectified Linear Unit (ReLU)*, mis tagastab väljundiks 0, kui saab negatiivse sisendi ja jääb samaks missuguse positiivse sisendi puhul. Funktsiooni eesmärk on aidata mudelil õppida andmestikus leiduvaid keerulisemaid mustreid ja tunnuseid [28]. Joonisel 3.10 on toodud välja ReLu aktiveerimisfunktsiooni väärtuste väljundi arvutamise graafik, mille peal on näha kuidas sisendid muutuvad kas nulliks või jäävad samasuguseks positiivseks arvuks.

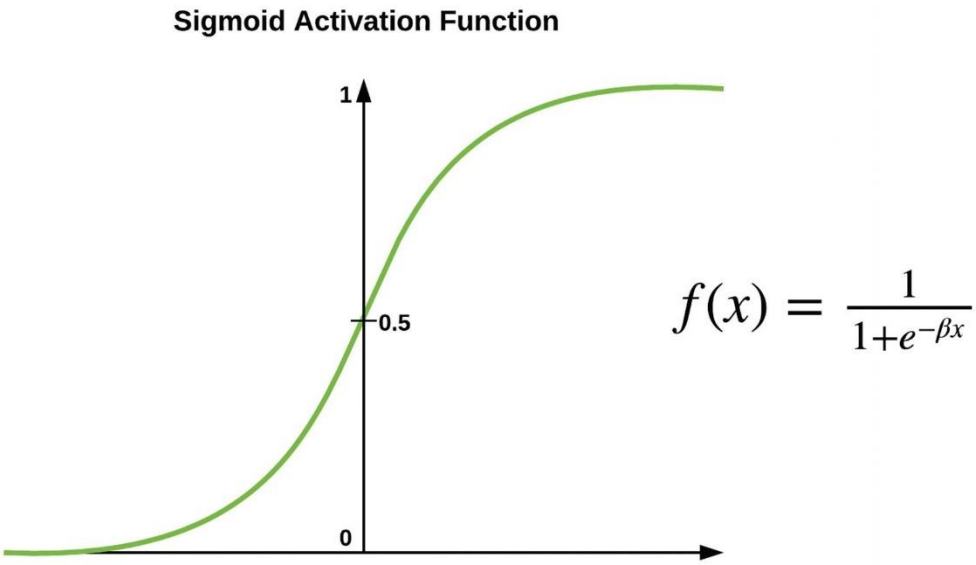



Joonis 3.10 – ReLU aktiveerimisfunktsioon – negatiivne sisend teisendatakse nulliks, positiivsed sisendid jäävad samaks [26]

Eelviimasena on lisatud Dropout kiht koos Flatten kihiga. Dropout kiht aitab ületreenimise vastu, heites välja peale igat tsüklit juhusliku osa neuronite ühendustest. See aitab kaasa uute sõltumatute ühenduste loomisele, mis omakorda soodustab mudeli

üldistusvõimet. Flatten kihi ülesanne on võtta talle antud sisendi andmed ning teisendada need üksikusse andmete vektorisse. Kui pildil on mõõtmed kõrgus ja laius, siis see arvutatakse ümber ainsaks arvuks. Muutes andmed ühedimensiooniliseks vektoriks, on need võimalik sööta järgnevasse Dense(1) kihti, mille ainuke ülesanne on anda lõplik tulemus kahe valiku vahel.

Viimane osa on taaskord Dense kiht, millel on aktivatsiooni parameeter "*sigmoid".* Kuna otsitav tulemus on binaarne klassifikatsioon (on või ei ole võlts), siis on viimasel Dense kihil üks väljund, mille väärtus on arvu 0 ja 1 vahel. *Sigmoid* funktsioon teisendab enda tulemused tõenäosust väljendavaks väärtuseks 0 ja 1 vahel [26]. Kasutades seda kihti saame lõpliku tulemuse, mis näitab kui suure tõenäosusega on pildi peal võltsi või päris naeratust sümboliseerivad elemendid. Joonisel 3.11 on näha Sigmoid funktsiooni graafikut koos selle valemiga.




Joonis 3.11 – Sigmoid aktiveerimisfunktsioon [26]

Kasutatavad parameetrid ja kihtid on valitud testimise tagajärjel. Enne mudeli treenimist on vajalik mudel kihtidest kokku kompileerida. Selle jaoks on antud mudelile optimeerija gradientlaskumine (*Gradient Descent optimizer – SGD*) koos õpisammu (*learning rate*) ja inertsiteguriga (*momentum*). Õpisammu ja inertsiteguri väärtused on valitud vastavalt parimatele tulemustele mitme erineva mudeli versiooni läbi testimisel.

Kompileerimise käigus antakse mudelile kahjufunktsioon *binary_crossentropy*. Kuna mudel näitab klassifitseerimise tõenäosust, kasutame funktsiooni koos mõõtmismeetrikaga, mis juhendab mudelit treenima lähtudes *Accuracy* hinnangust, mis loendab, kui mitu korda ennustus läheb täppi antud pildi klassifikatsiooniga [27].

# 4 PROGRAMMI ARHITEKTUUR

Programm koosneb kolmest osast:

1. Andmete töötlemine (Andmed.py)
2. Mudelite treenimine (Mudelid.py)
3. Tulemuste ennustamine (Tulemused.py).

## 4.1 Andmete töötlemine

Programmile Andmed.py on sisendiks antud kausta asukoha tee (*path*) arvutis, mis sisaldab varasemalt kategoriseeritud piltide kaustasid.

Piksel on digitaalse pildi väikseim element, mis kannab endaga kaasas arvulist väärtust, mis sümboliseerib selle pildi peal oleva asukoha infot näiteks värvi ja intensiivsuse kohta [29]. Kui võtta värviline pilt, mille suurus pikslites on näiteks 128 x 128 ja käideldes seda pilti kui ruumilist objekti, millel on kõrgus, laius ja sügavus, siis pikkus ja kõrgus näitavad pildi mõõtmeid pikslites ning sügavus sisaldab infot pildi värvi kohtas (rgb väärtused). Sellist pilti on võimalik sümboliseerida väärtustega andmestikus 128 x 128 x 3 [19]. Kui kasutada tavalist tehisvõrku piltide peal, siis oleks võrgu mõõtmed väga suured ning mudeli treenimise protsess aeglane, kuna pildi üks neuron sisaldaks 49,152 (128*128*3) andmeväärtust [17].

Pildi peal olevaid piksleid muudetakse niimoodi, et nad väljendaksid oma väärtusega valguse kogust. Piksel kannab sel juhul endaga kaasas informatsioon selle pildi peal oleva asukoha valguse intensiivsusest. Pildituvastuses kasutatakse sellist protseduuri, kuna see lihtsustab algoritmi tööd ja vähendab arvuti arvutusnõudeid [30]. Nähes värvilist pisklit, kuvatakse arvutile selle piksliga kaasa kolme tooni (punane, roheline ja sinine) väärtused, mis arvutatakse kokku värviks.

Masinõppe andmestiku eeltöötlemise juures üks võimalik samm on piltide hallidesse toonidesse töötlemine (*grayscale*). Funktsioon *näo_töötlemine* laeb kaustas oleva pildi programmi ning muudab selle hallidesse toonidesse, kasutades Pythoni teegi PIL meetodit *convert*. Nüüd kus piksel väljendab ennast ainult ühe väärtusega, siis on pildi tuvastamiseks vajaliku arvutuse maht väiksem [31]. Seejärel muudab funktsioon pildi programmi poolt loetavaks, muutes selle arvuliseks jadaks.

Järgnevalt leitakse ette antud pildi pealt üles kolme otsitava elemendi (nägu, suu, silm) koordinaadid, kasutades selleks Cascade Classifier meetodit. *Haar Cascade Classifiers* on sari klassifitseerijad, mis tuvastavad pildi peal olevaid objekte [32].

Pildi peal, kasutades leitud koordinaate, joonistatakse ümber otsitava elemendi ristkülik ning seejärel lõigatakse see välja. Selleks kasutatakse teegi OpenCV sisse ehitatud meetodeid *rectangle* ja *resize*.

Funktsioon *piltide_töötlemine* itereerib läbi kaustades olevate piltide ning kasutades funktsiooni *näo_töötlemine* salvestab leitud elemendi pildi jpg formaadis vastavatesse kaustadesse. Välja lõigatud piirkond salvestatakse eraldi jpg formaadis vastavasse kausta. Elementide pildid salvestatakse suurustega 300 x 300 pikslit. Protsessi on näha joonisel 4.1, kuidas algsest pildist eristatakse elemendid.

Joonis 4.1 – Funktsiooni näo_töötlemine pildi pealt elementide välja lõikamine

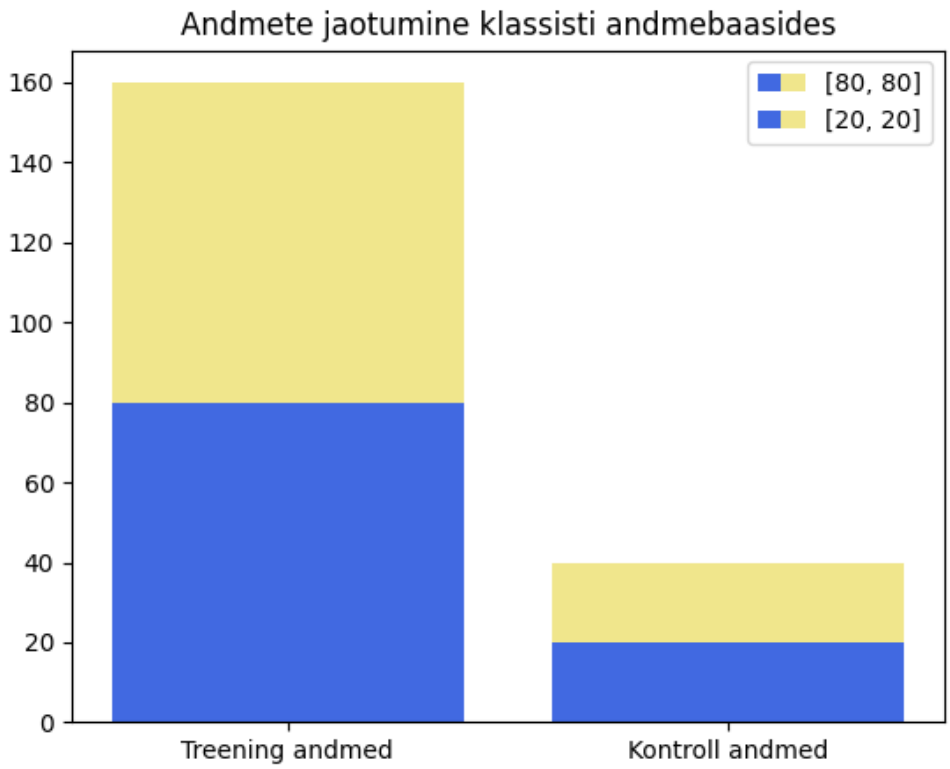
Seejärel itereerib funktsioon *andmebaaside_tekitamine* läbi elementide kaustades olevaid eelnevalt välja lõigatud näo osasid ja ehitab nende põhjal kolm andmeraamistiku. Andmeraamistikud sisaldavad pildi nime, asukohta ning klassifikatsiooni (1 – päris, 0 – võlts). Joonise 4.2 peal on toodud välja andmeraamistiku sisu peale funktsiooni käitamist. Pildile lisatakse juurde klass, asukoht arvutis ja nimi.

```
1.      klass  tee                                      nimi
2.  47  0      C://pildid//näod//fake_näod//f51.jpg     f51.jpg
3.  157 1      C://pildid//näod//real_näod//r60.jpg     r60.jpg
4.  180 1      C://pildid//näod//real_näod//r81.jpg     r81.jpg
5.  120 1      C://pildid//näod//real_näod//r27.jpg     r27.jpg
6.  42  0      C://pildid//näod//fake_näod//f47.jpg     f47.jpg
```

Joonis 4.2 – Funktsiooni *andmebaaside_tekitamine* näo piltide andmeraamistiku esimesed 5 pilti

Peale seda kasutatakse andmeraamistike peal teegi *sklearn* meetodit *train_test_split*, mis jaotab andmeraamistikud kaheks, eraldades sealt randomiseeritult etteantud koguse pilte. Nii eraldatakse treeningandmed ja valideerimisandmed. Meetodi

parameetriks on valitud 0.2 (20%) rakendades 60/20/20 reeglit [26]. Andmestikus olevad 200 pildist jaotatakse treeningandmeteks 160 ja valideerimisandmeteks 40. Kasutades meetodi parameetrit *stratify* garanteerime selle, et mõlemasse andmestikku on jaotatud võrdselt tõeseid ja võlts pilte. Kasutades funktsiooni *andmete_jagamise_visualiseerimine* failis Mudelid.py toome välja graafiku pealk andmete jaotumise mida on näha joonisel 4.3. Sinisega on tähistatud päris naeratusega piltide arvu ning kollasega võlts naeratusega piltide arvu. Graafikul näha, kuidas nii treening, kui ka kontroll andmete puhul jaotusid päris- ja võlts naeratustega piltide arvud võrdselt. Treening andmete puhul on 80 päris naeratusega pilt ning 80 võlts naeratusega pilti. Kontroll andmete puhul on 20 päris naeratusega pilti ning 20 võlts naeratusega pilti.




Joonis 4.3 – Andmete jaotumine peale meetodit train_test_split. Visualiseeritud funktsiooniga andmete_jagamise_visualiseerimine.

Funktsioon *ennustuste_töötlemine* teeb sarnast tööd nagu funktsioon *piltide_töötlemine*, kuid siinkohal võetakse pildid "Ennustuste" kaustast. Piltide pealt otsitakse üles elemendid, mis salvestatakse eraldi kaustadesse. Pildid ei ole eelnevalt kategoriseeritud, kuna nende piltide peal käideldakse treenitud mudeleid, mis annavad piltidele lõplikud tulemused – kas pildi peal on tegu võlts naeratusega või päris naeratusega. Võrdväärse tulemuste mõõtmiseks on lisatud ennustuste kausta 10 päris ja 10 võlts pilti.

## 4.2 Mudelite treenimine

Programmi Mudelid.py ülesanne on eelnevalt ehitatud andmeraamistike põhjal treenida välja mudelid. Protsess algab funktsioonist *mudeli_ehitamine*, mis konstrueerib baasmudeli, kasutades eeltreenitud konvolutsioonilise võrgu arhitektuuri VGG16.

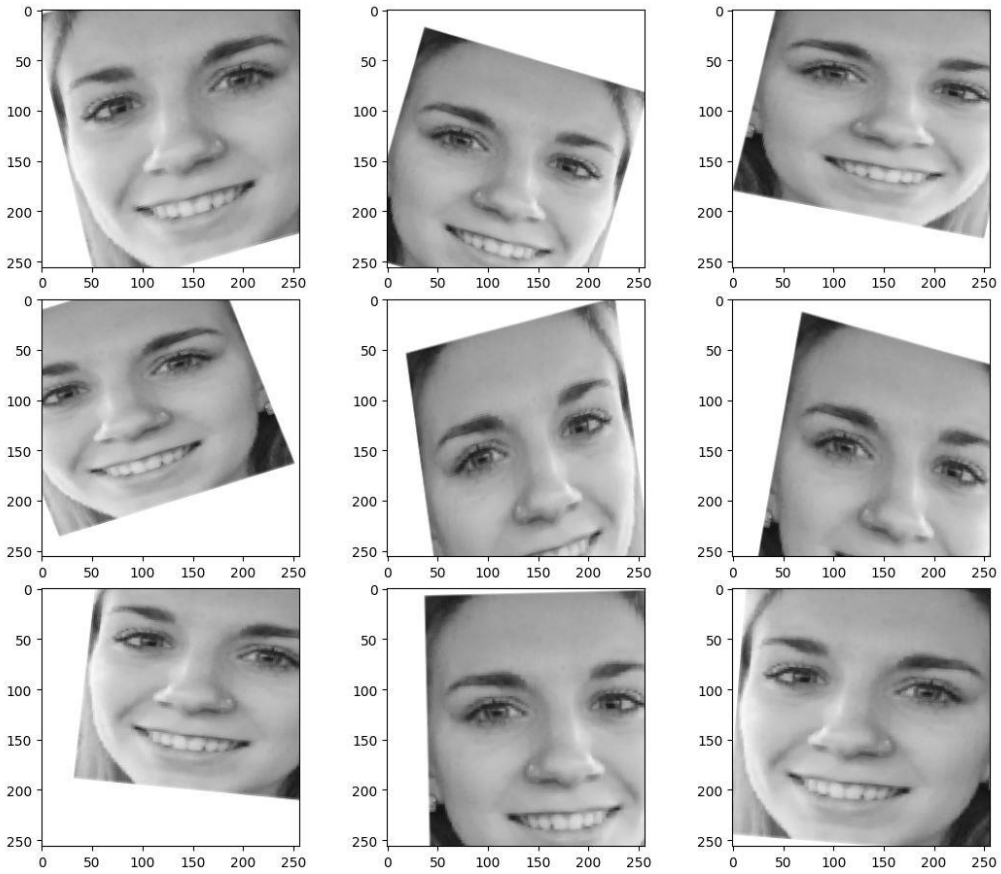
Mudel kompileeritakse kasutades optimeerijat SGD, millele on lisatud juurde õpisamm 1e-4 (0.0001) ning inertsitegur 0,9. Mudeli kahjufunktsiooniks seatakse *binary_crossentropy*. Mudel hakkab oma edukust mõõtma mõõtmisstandardi *accuracy* pealt. Kui kõik parameetrid on kokku kompileeritud, siis see salvestatakse kokku treenimata mudeliks. Joonis 4.4 näitab mudeli arhitektuuri ja seal sees olevaid kihte:

```
1.  Model: "VGG16"
2.  _________________________________________________________________
3.   Layer (type)                 Output Shape              Param #
4.  =================================================================
5.   input_1 (InputLayer)        [(None, 300, 300, 3)]     0
6.
7.   block1_conv1 (Conv2D)       (None, 300, 300, 64)      1792
8.
9.   block1_conv2 (Conv2D)       (None, 300, 300, 64)      36928
10.
11.  block1_pool (MaxPooling2D)  (None, 150, 150, 64)      0
12.
13.  block2_conv1 (Conv2D)       (None, 150, 150, 128)     73856
14.
15.  block2_conv2 (Conv2D)       (None, 150, 150, 128)     147584
16.
17.  block2_pool (MaxPooling2D)  (None, 75, 75, 128)       0
18.
19.  block3_conv1 (Conv2D)       (None, 75, 75, 256)       295168
20.
21.  block3_conv2 (Conv2D)       (None, 75, 75, 256)       590080
22.
23.  block3_conv3 (Conv2D)       (None, 75, 75, 256)       590080
24.
25.  block3_pool (MaxPooling2D)  (None, 37, 37, 256)       0
26.
27.  block4_conv1 (Conv2D)       (None, 37, 37, 512)       1180160
28.
29.  block4_conv2 (Conv2D)       (None, 37, 37, 512)       2359808
30.
31.  block4_conv3 (Conv2D)       (None, 37, 37, 512)       2359808
32.
33.  block4_pool (MaxPooling2D)  (None, 18, 18, 512)       0
34.
35.  block5_conv1 (Conv2D)       (None, 18, 18, 512)       2359808
36.
37.  block5_conv2 (Conv2D)       (None, 18, 18, 512)       2359808
38.
39.  block5_conv3 (Conv2D)       (None, 18, 18, 512)       2359808
40.
41.  block5_pool (MaxPooling2D)  (None, 9, 9, 512)         0
42.
43.  global_max_pooling2d (Globa  (None, 512)              0
44.  lMaxPooling2D)
45.
46.  dense (Dense)               (None, 512)               262656
```

```
47.
48.  dropout (Dropout)           (None, 512)               0
49.
50.  flatten (Flatten)           (None, 512)               0
51.
52.  dense_1 (Dense)             (None, 1)                 513
53.
54. =================================================================
55. Total params: 14,977,857
56. Trainable params: 14,977,857
57. Non-trainable params: 0
```

Joonis 4.4 – VGG16 mudeli ja juurde lisatud kihtide arhitektuur

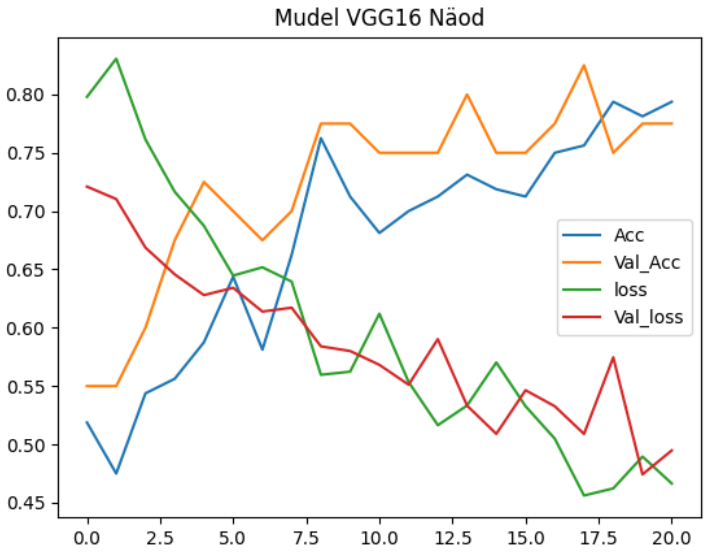
Funktsioon *generaatorite_ehitamine* valmistab eelnevalt ehitatud andmeraamistike põhjal "piltide andmegeneraatorid" (ImageDataGenerator), mille ülesanne on jooksutada piltide andmed masinõppe mudelisse. Selleks, et vältida ületreenimist on siin etapis kasutatud andmete rikastamist (*Data Augmentations*). Andmete rikastamise protsessi eesmärk on kunstlikult andmete koguse suurendamine, viies läbi olemasolevate piltide peal lihtsaid muudatusi, nagu näiteks keeramine, suuruse muutmine, ümber pööramine jne. Joonisel 4.5 on visualiseeritud pildi peal muudatuste läbi viimine kasutades funktsiooni *pildi_augmentatsiooni_kuvamine*.



Joonis 4.5 – Funktsiooni generaatorite_ehitamine andmete rikastamise visualiseerimine funktsiooni pildi_augmentatsiooni_kuvamine abil

Pannes kokku salvestatud treenimata mudel ja piltide andmete generaatorid, viiakse läbi mudeli treenimise faas. Teine meetod ületreenimise ennetamiseks on kasutada teegi Keras meetod *EarlyStopping*, mis lõpetab mudeli treenimise enneaegselt, peale seda, kui on märgatud mudeli treenimise protsessi aeglustumist. Meetod takistab mudeli üleliigset treenimist. Peale treenimise faasi mudel salvestatakse. Sama treenimise protsess viiakse läbi kõigi kolme elemendi jaoks eraldi. Lõpptulemusena saadakse kolm eraldi mudelit nägude, silmade ja suude jaoks.

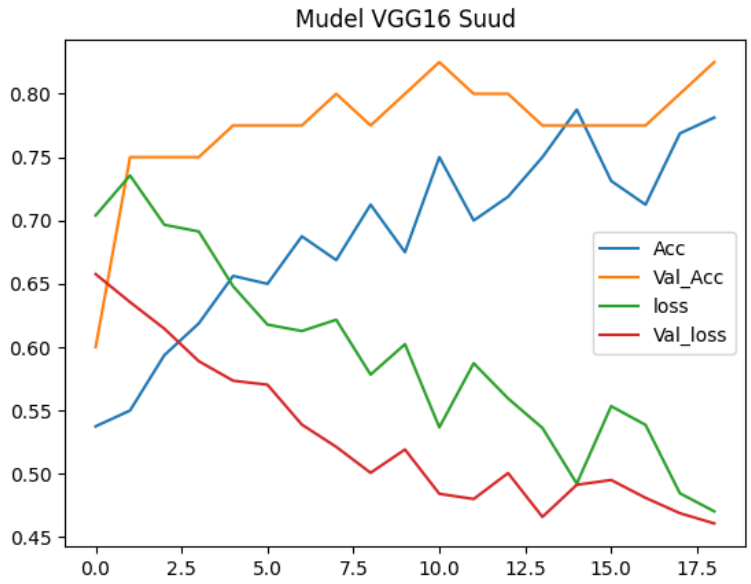
1. Nägude mudel




Joonis 4.6 – Nägude mudeli treenimise graafik

Nägude mudeli treenimise tulemuseks sai - loss: 0.4677 - accuracy: 0.7905 - val_loss: 0.4973 - val_accuracy: 0.7750

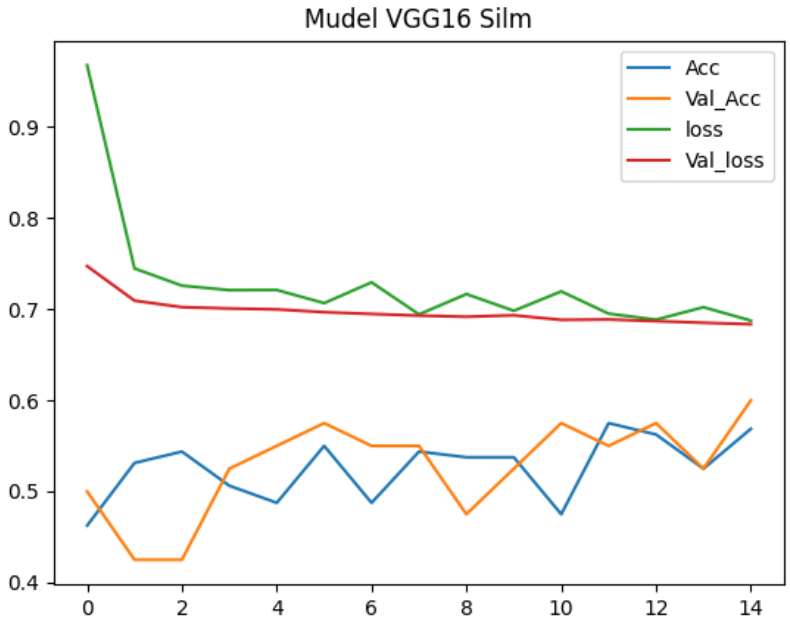
2. Suude mudel




Joonis 4.7 - Suude mudeli treenimise graafik

Suude mudeli treenimise tulemuseks sai - loss: 0.4704 - accuracy: 0.7812 - val_loss: 0.4609 - val_accuracy: 0.8250

3. Silmade mudel




Joonis 4.8 - Silmade mudeli treenimise graafik

Silmade mudeli treenimise tulemuseks sai - loss: 0.6875 - accuracy: 0.5688 - val_loss: 0.6836 - val_accuracy: 0.6000

Keskmine treeningandmestiku täpsus (Acc) kolme mudeli vahel on 0.7135 ja keskmine valideerimisandmestiku täpsus (Val_Acc) on 0.7333

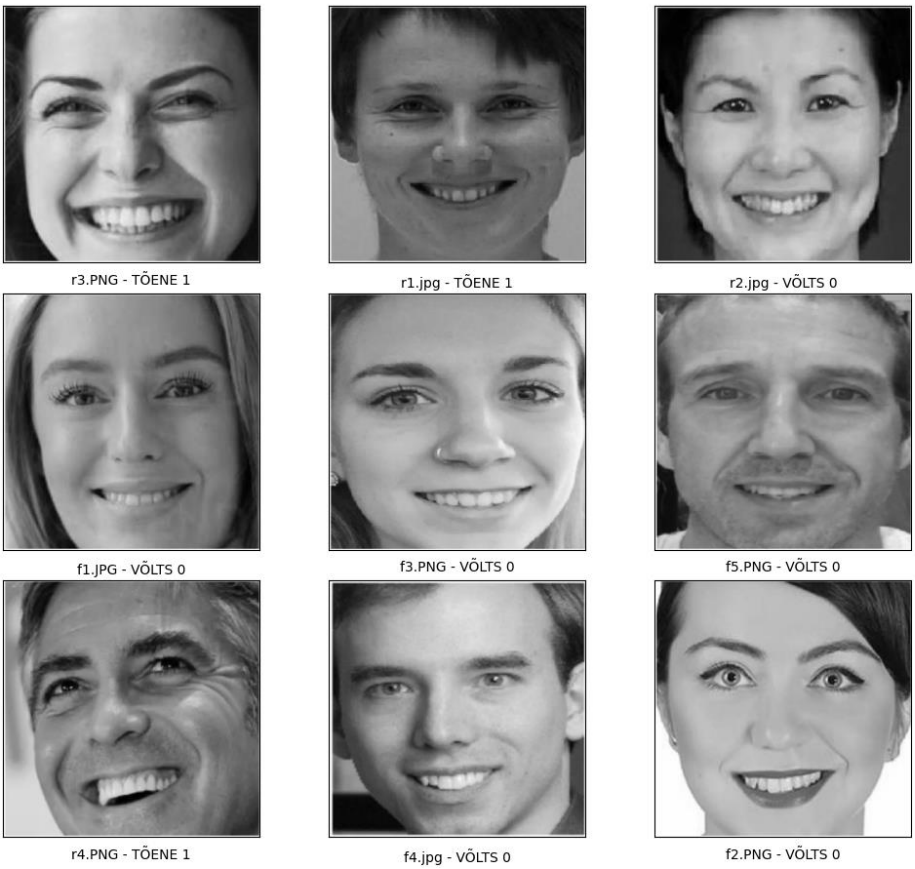
## 4.3 Tulemuste ennustamine

Programm *Tulemused.py* kasutab failis *Andmed.py* funktsiooniga *ennustuste_töötlemine* kausta "Ennustused" salvestatud elementide pilte ning failis *Mudelid.py* salvestatud mudeleid. Alustuseks tekitab programm funktsiooniga *piltide_elementide_hankimine* ennustuste kaustas olevatest elementide piltidest andmeraamistikud.

Seejärel ehitatakse, sarnaselt failis *Mudel.py*, andmete generaatorid iga elemendi kohta. Funktsioonis *ennustan_pildid* laetakse salvestatud mudelid ning kasutades teegi Keras meetodit *predict*, antakse igale elemendile vastava mudeliga tulemus. Iga elemendi andmeraamistikus oleva pildi juurde lisatakse väärtus *tulemus*, mis näitab kas mudeli arvates on tegu võlts või päris naeratuse juurde käiva elemendiga. Tulemus väljendatakse arvuna vahemikus 0 kuni 1. Kui mudeli antud väärtus on suurem kui >=0.5, siis on tulemuseks 1 ehk "päris", kui väärtus on alla <0.5, siis on tulemuseks 0 ehk "võlts".

Funktsioon *lõplik_ennustus* loeb iga elemendi andmeraamistikku lisatud tulemused kokku ning annab lõpliku tulemuse, kas pildi peal on tegu võlts naeratusega või päris naeratusega. Kolme elemendi tulemus liidetakse kokku ning kui summa väärtus on üle kahe, siis loetakse lõplikku tulemust tõeseks ehk "päris". Lõplik tulemus oleneb erinevatele elementidele antud hinnangutest, kui näiteks elemendid suu ja silm said tulemusteks väärtuse 1 ehk "päris", aga element nägu sai tulemuseks väärtuse 0 ehk "võlts", siis loetakse lõplikku tulemust ikkagist päriseks, kuna suurem osa elemente said tulemuseks "päris". Lõplik tulemus oleneb tulemuste enamusest. Näiteks kui ennustamise käigus on näo ja suu tulemuseks 1 ehk "päris", aga silma tulemuseks on 0 ehk "võlts, siis lõplikuks tulemuseks tuleb 1 ehk "päris", kuna kolmest elemendist kaks said tulemuseks 1 ehk "päris.

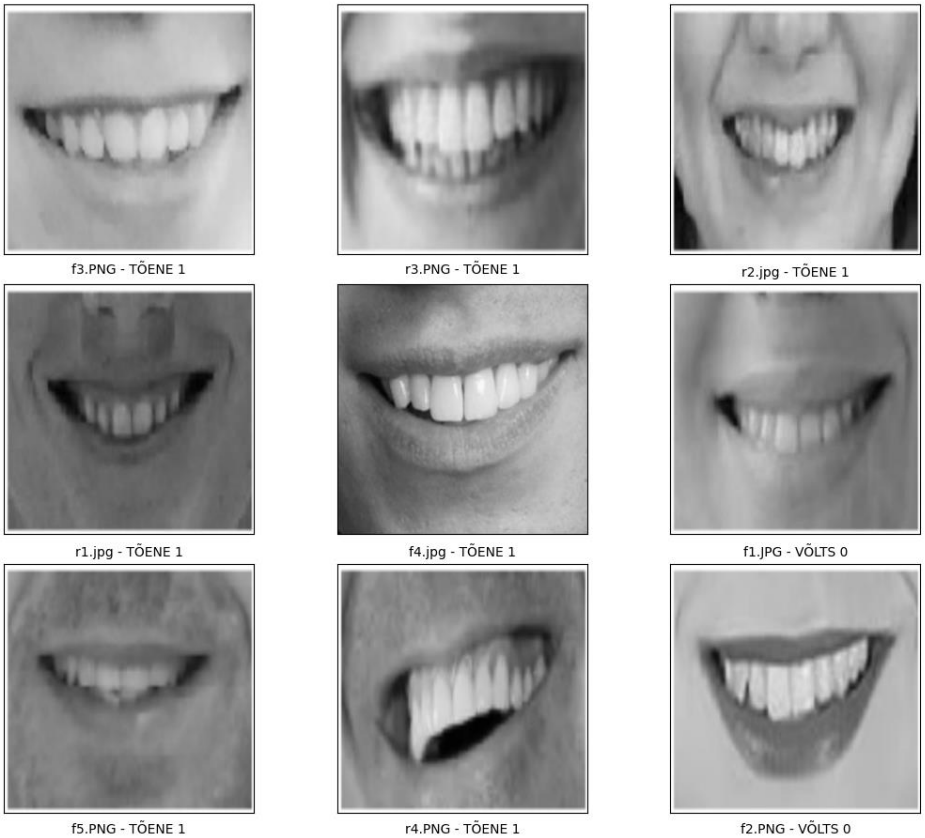
Funktsioon *ennustuste_kuvamine* väljastab akna, kus kuvatakse pilt ning temale antud tulemus, kasutades selleks teeki *matplotlib.pyplot*. Lisaks kirjutab funktsioon välja protsendid, mis väljendavad kui tõenäoliselt on pilt tõene või võlts. Funktsiooni tööd on näha joonistel 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 ja 4.16, kus piltide alla on kirjutatud juurde temale antud tulemus ning programm kirjutab välja ennustuse protentuaalse väärtuse, mis mudel pildile kaasa andis.




Joonis 4.9 – Näo piltide ennustuste kuvamine kasutades funktsiooni ennustuste_kuvamine

```
1. Nägu :  r3.PNG  - pilt on 0.74% TÕENE ja 0.26% VÕLTS
2. Nägu :  r1.jpg  - pilt on 0.88% TÕENE ja 0.12% VÕLTS
3. Nägu :  r2.jpg  - pilt on 0.05% TÕENE ja 0.95% VÕLTS
4. Nägu :  f1.JPG  - pilt on 0.12% TÕENE ja 0.88% VÕLTS
5. Nägu :  f3.PNG  - pilt on 0.47% TÕENE ja 0.53% VÕLTS
6. Nägu :  f5.PNG  - pilt on 0.12% TÕENE ja 0.88% VÕLTS
7. Nägu :  r4.PNG  - pilt on 0.99% TÕENE ja 0.01% VÕLTS
8. Nägu :  f4.jpg  - pilt on 0.16% TÕENE ja 0.84% VÕLTS
9. Nägu :  f2.PNG  - pilt on 0.08% TÕENE ja 0.92% VÕLTS
```

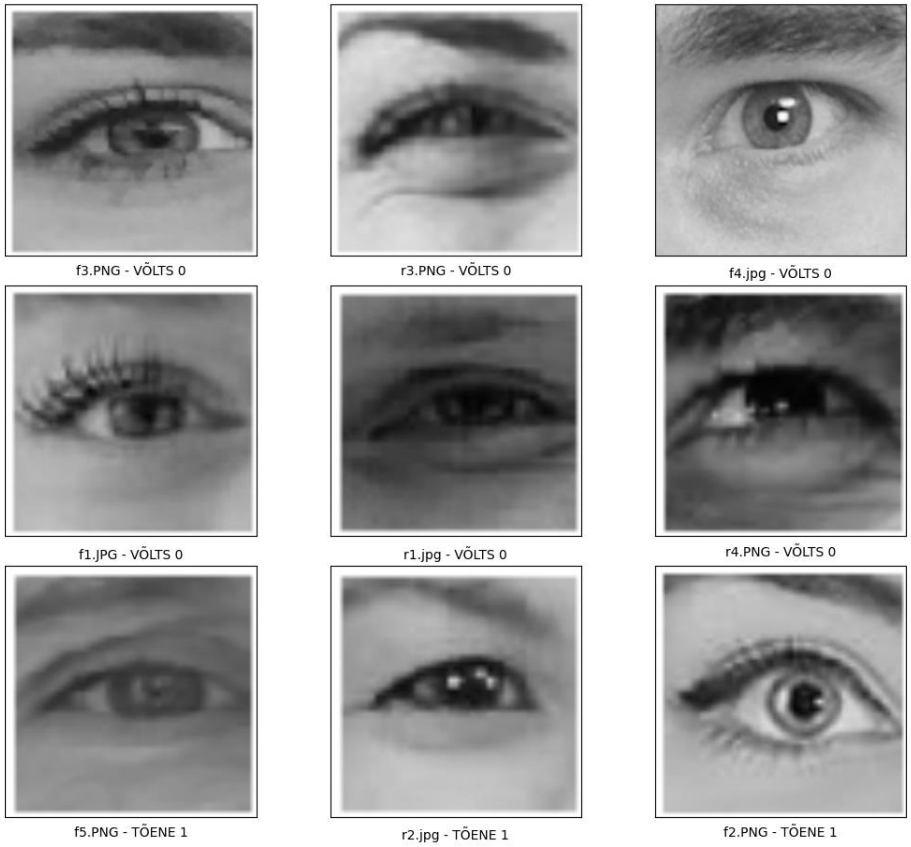
Joonis 4.10 – funktsiooni *ennustuste_kuvamine* väljund näo piltide protsentuaalsed väärtused




Joonis 4.11 – Suu piltide ennustuste kuvamine kasutades funktsiooni ennustuste_kuvamine

```
1.  Suu :  f3.PNG  - pilt on 0.96% TÕENE ja 0.04% VÕLTS
2.  Suu :  r3.PNG  - pilt on 0.59% TÕENE ja 0.41% VÕLTS
3.  Suu :  r2.jpg  - pilt on 0.96% TÕENE ja 0.04% VÕLTS
4.  Suu :  r1.jpg  - pilt on 0.72% TÕENE ja 0.28% VÕLTS
5.  Suu :  f4.jpg  - pilt on 0.99% TÕENE ja 0.01% VÕLTS
6.  Suu :  f1.JPG  - pilt on 0.42% TÕENE ja 0.58% VÕLTS
7.  Suu :  f5.PNG  - pilt on 0.72% TÕENE ja 0.28% VÕLTS
8.  Suu :  r4.PNG  - pilt on 0.90% TÕENE ja 0.10% VÕLTS
9.  Suu :  f2.PNG  - pilt on 0.22% TÕENE ja 0.78% VÕLTS
```

Joonis 4.12 – funktsiooni ennustuste_kuvamine väljund suu piltide protsentuaalsed väärtused

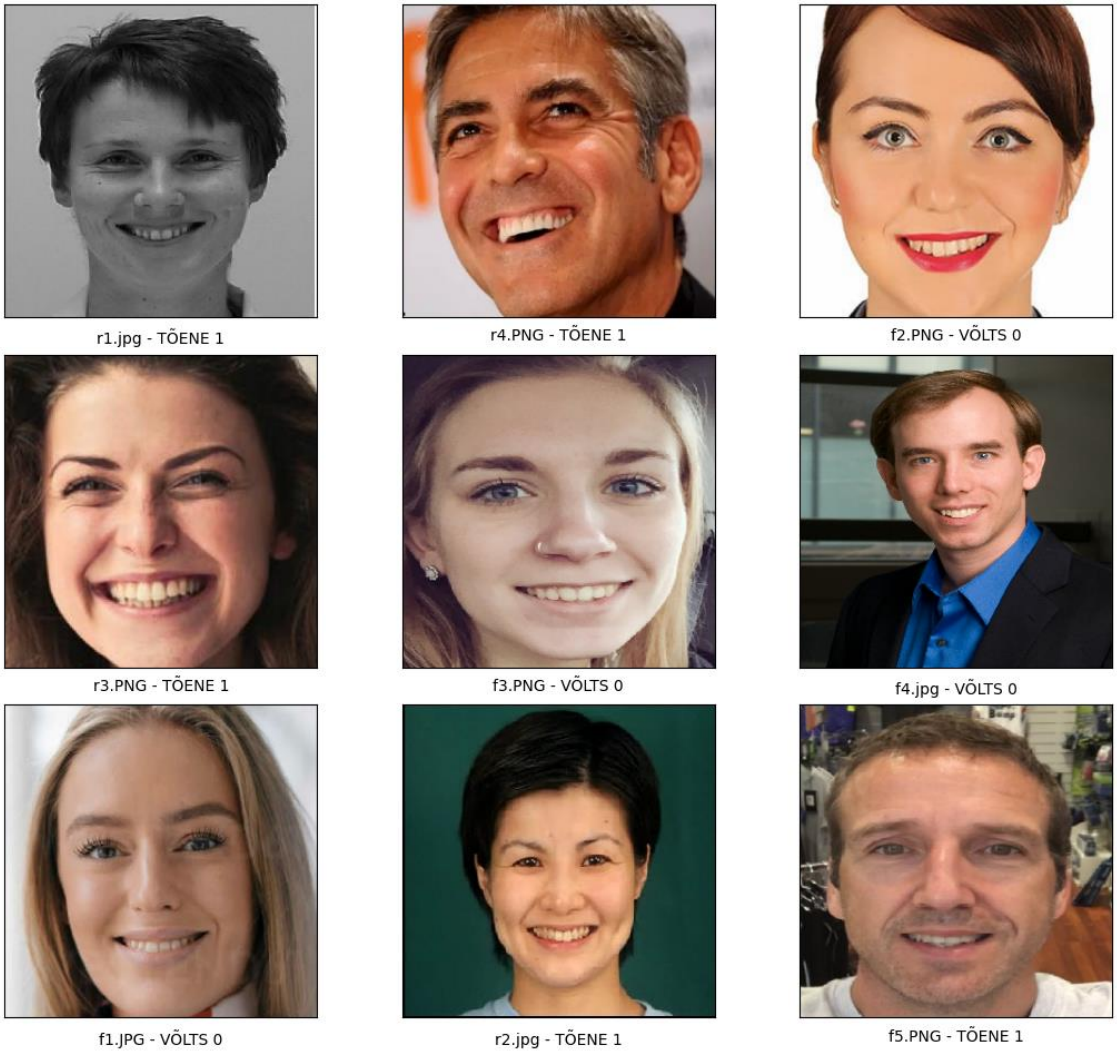



Joonis 4.13 – Silma piltide ennustuste kuvamine kasutades funktsiooni ennustuste_kuvamine

```
1. Silm :  f3.PNG  - pilt on 0.47% TÕENE ja 0.53% VÕLTS
2. Silm :  r3.PNG  - pilt on 0.45% TÕENE ja 0.55% VÕLTS
3. Silm :  f4.jpg  - pilt on 0.48% TÕENE ja 0.52% VÕLTS
4. Silm :  f1.JPG  - pilt on 0.36% TÕENE ja 0.64% VÕLTS
5. Silm :  r1.jpg  - pilt on 0.45% TÕENE ja 0.55% VÕLTS
6. Silm :  r4.PNG  - pilt on 0.31% TÕENE ja 0.69% VÕLTS
7. Silm :  f5.PNG  - pilt on 0.51% TÕENE ja 0.49% VÕLTS
8. Silm :  r2.jpg  - pilt on 0.78% TÕENE ja 0.22% VÕLTS
9. Silm :  f2.PNG  - pilt on 0.71% TÕENE ja 0.29% VÕLTS
```

Joonis 4.14 – funktsiooni ennustuste_kuvamine väljund silma piltide protsentuaalsed väärtused

Funktsiooni *ennustuste_kuvamine* lõpliku piltide admeraamistike peal väljastab akna, kus kuvatakse algseid ennustatavaid pilte ning neile antud lõplikke tulemusi.




Joonis 4.15 – Silma piltide ennustuste kuvamine kasutades funktsiooni ennustuste_kuvamine

```
1.  r1.jpg  - pilt on TÕENE - 1  Nägu 1, Silm 0, Suu 1
2.  r4.PNG  - pilt on TÕENE - 1  Nägu 1, Silm 0, Suu 1
3.  f2.PNG  - pilt on VÕLTS - 0  Nägu 0, Silm 1, Suu 0
4.  r3.PNG  - pilt on TÕENE - 1  Nägu 1, Silm 0, Suu 1
5.  f3.PNG  - pilt on VÕLTS - 0  Nägu 0, Silm 0, Suu 1
6.  f4.jpg  - pilt on VÕLTS - 0  Nägu 0, Silm 0, Suu 1
7.  f1.JPG  - pilt on VÕLTS - 0  Nägu 0, Silm 0, Suu 0
8.  r2.jpg  - pilt on TÕENE - 1  Nägu 0, Silm 1, Suu 1
9.  f5.PNG  - pilt on TÕENE - 1  Nägu 0, Silm 1, Suu 1
```

Joonis 4.16 – funktsiooni ennustuste_kuvamine väljund algsete piltide protsentuaalsed väärtused

Funktsioon *salvestan_tabelisse* salvestab tulemustega koos andmeraamistiku .csv tabelina *Excel*-i faili.

# 5 TULEMUSTE ANALÜÜS

Eksimismaatriks (*confusion matrix*) on viis kuidas mõõta mudeli täpsust, võrreldes ennustatavaid tulemusi tegelike tulemustega. Maatriks on 2 x 2 tabel mis põhineb nelja väärtuse esinemise sagedusel ja nendevahelistel suhetel. Elemendid on: True Negative (TN), True Positive (TP), False Negative (FN) ja False Positive (FP). Ennustamiste täpsuse (*accuracy*) saab arvutada kui liita TP + TN ja jagada see kõikide mõõtmiste arvuga. Täpsus näitab kui palju mudel ennustas korrektselt. Eksimismäära (*error rate*) saab kui liita FP + FN ja jagada see kõikide mõõtmiste arvuga. Eksimismäär näitab kui kui palju mudel ennustas valesti. *Sensitivity (recall)* ehk tabavus näitab kui mitu korda positiivsetest tulemustest ennustati korrektselt. Tabavust saab arvutada jagades TP tegelike positiivsete mõõtmistega (päris naeratus). Lisaks on mudeli täpsust võimalik hinnata kasutades F-skoori. F-skoori saab arvutada korrutades TP kahega ja jagades selle kaks korda TP + FP + FN. F-skoori väärtus jääb 0 ja 1 vahele, ning mida kõrgem see on, seda kõrgemaks hinnatakse mudeli täpsust [33].

Ennustuste kaustas on 20 pilti, millest 10 on päris ja 10 on võlts. Funktsiooni *salvestan_tabelisse* tagajärjel salvestatakse piltide tulemused .csv faili.

Table 5.1 – Tulemuste andmestik

| | nimi | nägu | suu | silm | tulemus |
|---|---|---|---|---|---|
| 0 | f1.JPG | 1 | 1 | 0 | 1 |
| 1 | f10.jpg | 1 | 1 | 0 | 1 |
| 2 | f2.PNG | 1 | 0 | 0 | 0 |
| 3 | f3.PNG | 1 | 0 | 0 | 0 |
| 4 | f4.jpg | 1 | 0 | 0 | 0 |
| 5 | f5.PNG | 1 | 1 | 0 | 1 |
| 6 | f6.JPG | 0 | 0 | 1 | 0 |
| 7 | f7.JPG | 0 | 1 | 1 | 1 |
| 8 | f8.JPG | 0 | 0 | 1 | 0 |
| 9 | f9.jpg | 0 | 1 | 0 | 0 |
| 10 | r1.jpg | 0 | 1 | 1 | 1 |
| 11 | r10.jpg | 0 | 1 | 1 | 1 |
| 12 | r2.jpg | 0 | 1 | 1 | 1 |
| 13 | r3.PNG | 0 | 1 | 1 | 1 |
| 14 | r4.PNG | 0 | 0 | 1 | 0 |
| 15 | r5.JPG | 0 | 0 | 0 | 0 |
| 16 | r6.jpeg | 1 | 1 | 1 | 1 |
| 17 | r7.jpg | 1 | 0 | 1 | 1 |
| 18 | r8.jpg | 1 | 1 | 1 | 1 |
| 19 | r9.jpg | 0 | 0 | 0 | 0 |

Esimese korra tulemuste andmestik on toodud välja Tabelis 1. Tabelis olevatest kahekümnest pildist esimesed kümme on võltsid. Vaadeldes tabelis olevaid tulemusi, on

näha, et võlts piltidest ennustati 4 tükki päriseks (*False Positive*) ning 6 (*True Negative*) tükki võltsiks. Päris piltidest ennustati 7 (*True Positive*) tükki päriseks ja 3 (*False Negative*) tükki võltsiks. Esimese tabeli tulemuste eksimismaatriks on toodud välja tabelis 2.

Table 5.2 - Esimese tabeli tulemuste eksimismaatriks

| | Ennustus Võlts | Ennustus Päris | |
|---|---|---|---|
| Tegelik Võlts | 6 (TN) | 4 (FP) | 10 |
| Tegelik Päris | 3 (FN) | 7 (TP) | 10 |
| | 9 | 11 | |

Kasutades funktsiooni *eksimismaatriks*, andes talle sisendiks tulemuste andmestiku, arvutab funktsioon välja mudeli täpsust iseloomustavad väärtused. Funktsioon võtab arvesse piltide nimed (nime ees olev täht: r – päris, f – võlts) nende kategoriseerimisel. Tabelis 2 välja toodud andmete põhjal on väärtused järgnevad: täpsus – 0.65, eksimismäär – 0.35, tabavus 0.7 ja F-skoor 0.666.

Kuna vähese andmestiku tõttu on risk, et väikesegi kõrvalkalde puhul võivad tulemused erineda suuresti, siis jooksutati programmi 5 korda, et leida lõplike tulemuste keskmised. Järgnevad tabelid on nende 5 korra eksimismaatriksite kokku liidetud eksimismaatriksid. Kuna programmi jooksutati eraldiseisva instantsina, ei mõjuta tulemused üksteist ning andsid iseseisvad tulemused.

Table 5.3 - Näo tulemuste eksimismaatriks

| | Ennustus Võlts | Ennustus Päris | |
|---|---|---|---|
| Tegelik Võlts | 25 (TN) | 25 (FP) | 50 |
| Tegelik Päris | 30 (FN) | 20 (TP) | 50 |
| | 55 | 45 | |

Täpsus – 0.45, eksimisäär – 0.55, tabavus 0.4 ja F-skoor 0.421051.

Table 5.4 - Suu tulemuste eksimismaatriks

| | Ennustus Võlts | Ennustus Päris | |
|---|---|---|---|
| Tegelik Võlts | 24 (TN) | 26 (FP) | 50 |
| Tegelik Päris | 21 (FN) | 29 (TP) | 50 |
| | 45 | 55 | |

Täpsus – 0.53, eksimismäär – 0.47, tabavus 0.58 ja F-skoor 0.552381

Table 5.5 - Silmade tulemuste eksimismaatriks

| | Ennustus Võlts | Ennustus Päris | |
|---|---|---|---|
| Tegelik Võlts | 27 (TN) | 23 (FP) | 50 |
| Tegelik Päris | 18 (FN) | 32 (TP) | 50 |
| | 45 | 55 | |

Täpsus – 0.59, eksimismäär – 0.41, tabavus 0.64 ja F-skoor 0.609524

Nägude, suude ja silmade tabelite keskmised arvutused on täpsus – 0.52333, eksimismäär 0.47667, tabavus 0.54 ja F-skoor 0.52765. Võrreldes mudelite treenimisel tulemusteks saadud täpsustega on eksimismaatriskite abil välja arvutatud väärtused madalamad. See võib viidata mudelite ületreenimisele, kus mudelite treenimise käigus on tulemused kõrgemad kui mudelite tegelikud sooritused.

Table 5.6 – Lõplike tulemuste eksimismaatriks

| | Ennustus Võlts | Ennustus Päris | |
|---|---|---|---|
| Tegelik Võlts | 24 (TN) | 26 (FP) | 50 |
| Tegelik Päris | 22 (FN) | 28 (TP) | 50 |
| | 46 | 54 | |

Täpsus – 0.52, eksimismäär – 0.48, tabavus 0.56 ja F-skoor 0.538462.

Mudeli eeldatav tegelik täpsus eksimismaatriksi põhjal on 0.52 ehk 52%, mis on võrreldes mudelite treenimise käigus tulemusteks saandud täpsusestest väiksem. Mudeli täpsus on oodatust väiksem. Mudelid näitasid treenimise faasis paljulubavaid tulemusi jõudes täpsuse parameetritega 80% kanti, kuid tegelikkuses ei saavuta mudel soovitud tulemust. Põhjuseks võib olla andmebaaside piltide halb kvaliteet, kust mudel ei loe piisavalt väikeseid nüansse välja, et tuvastada erinevusi võlts- ja pärisnaeratuse vahel. Samas ei ole võimalik mudeleid ka väikese andmebaasi koguse tõttu liigselt treenida.

# KOKKUVÕTE

Töös anti ülevaade Duchenne'i naeratusest, masinõppest, konvolutsioonilisest närvivõrgust, erinevatest olemasolevatest eel-treenitud mudelitest ja masinõppe mudeli treenimisest. Tulemuste analüüsimise käigus toodi välja viis kuidas kontrollida mudeli täpsust.

Töö eesmärgiks oli koostada algne andmestik piltidest, mis kujutavad võlts- ja pärisnaeratusi ning treenida nende peal masinõppe mudelit, mis oskaks piltide peal olevaid naeratusi eristada. Töö baseerus suuresti kirjutatud programmidele, mis on saadaval sissejuhatuse all välja toodud lingil. Töö käigus tutvuti masinõppega, kuidas toimivad tehisnärvivõrgud ning tutvustati viisi, kuidas närvivõrke õpetada. Lähemalt tutvustati konvolutsioonilist närvivõrku ning selle unikaalsust. Täpsemalt käidi üle, mis kihtidest konvolutsiooniline võrk on ehitatud ning mis on nende kihtide ülesanded. Töö käigus tutvustati Tensorflow tarkvara ning Kerase teeki, eelkõige just sealseid populaarseid eel-treenitud mudeleid ning nende omapärasid. Selgitati, miks ehitati töö eesmärgi täitmiseks omapärane mudel ning mis rolli täidavad erinevad parameetrid mudeli treenimise juures.

Peatükis Programmi arhitektuur käidi üle milline on ehitatud programm ning milliseid ülesandeid funktsioonid süsteemi sees täidavad. Programm on suuresti jaotatud kolmeks osaks. Iga pildi peal tehakse tööd kolm elementiga: nägu, silm ja suu. Iga elemendi põhjal on treenitud isiklik mudel, mille ülesanne on õppida eristama elemente kas võlts - või pärisnaeratusega kokku käivaks. Programm visualiseerib mudelite tööd ning toob välja iga elemendi tulemuse (1 ehk "päris või 0 ehk "võlts). Lõplik tulemus pildi suhtes otsustatakse tulemuste enamuse põhjal. Kui pildi kaks elementi kolmest on saanud tulemuseks 1 ehk "päris", siis antakse pildile vastav lõplik tulemus. Töö tulemuste analüüsi peatükis tutvustati mudeli täpsuse hindamise protsessi.

Töö tugevaks küljeks on selle unikaalsus. Duchenne'i naeratus spetsiifiliselt on varasemalt masinõppes vähe uuritud ning selle jaoks puudub vastav andmestik, sel põhjusel koostati selle töö raames täiesti uus piltide andmebaas. Lisaks pidi töö autor kirjandusele tuginedes mõtlema välja viisi, kuidas masinõppes adekvaatselt eristada naeratuse liike.

Töö nõrgaks küljeks jäi piltide andmestik. Kuna pole olemas konkreetset Duchenne'i/võlts naeratuse andmestikku, siis polnud võimalik leida adekvaatses koguses pilte. Liigselt väike andmestik võib mudeli treenimisel tekitada ebastabiilsust. Mudelite treenimisel oli näha pidevat kõikumist treenimise graafikute peal ning puudus selge ühtsus tulemuste suhtes. Mudelit oleks saanud rohkem optimeerida väikese

andmestiku peal töötamiseks, kuid kuna piltide peal otsitavad nüansid olid väikesed ning piltide andmestik minimaalne, siis oleks selline optimeerimine varem või hiljem jäänud stoppama. Kui teemat edasi arendada tuleks alustada rikkalikuma andmestiku kogumisega.

# SUMMARY

The thesis gave an overview of the Duchenne smile, maschine learning, convulational neural network, available pre-trained models and the training of machine learning models. Durng the analysis of the results a methos to measure the models accuracy was presented.

The aim of this work was to construct a databse of pictures that show fake and genuine smiles and to train a machine learning model, that could distinguish the two. This work is largely based of the program written, that can be accessed via the link provided under the introduction. The basics of machine learning was explained, how neural networks work and how to train them. Convulation networks were discussed in depth and what makes them unique. What is the architecture of a convulation network, what kind of layers is it made out of and what is the purpose of those layers. An overview of the Tensorflow library and Keras framework was given and the pre-trained models found within that framework. An explanation was given as the why a unique models was built for this thesis and what of work do the layers added do during the model training.

In the chapter "Programmi arhitektuur" (*Architecture of the program*) an overview of the program was given and what kind of fucntion does the code serve. The program was largely divided into three parts. Every part works on a certain element: face, eyes and mouth. A personal models was trained for each element so that it can distinguish between the fake and real version of them. The program visualizes the work done within the models and the predictions given to each picture. For the final results, a majority rule was proposed. If two elements of a picture received a result of 1, as in "real", out of the three elements, then the final result of the picture is "real". In the chapter reviewing the results, a way to analyse the results of the models accuracy was given.

A strong part of the work lies within its uniqueness. As the Duchenne smile has not been thoroughly researches with the machine learning space, the was no exsisting databse for fake smiles and real smiles. For this reason, a new database was created by hand. Also an adequate method to was proposed to distinguish fake smiles from real smiles using machine learning.

Since there was no exsiting database for fake smiles, this was also a weak part of the thesis. An adequate numbers of pictures was not found. There is a risk with using a small databse, that the trained model might be unstable in its accuracy. An fluctuation of the training parameters was observed during the training of the models. The modles could have been more optimized to work specifically with smaller databases, but since

the nuances that were sought on the picutres were small and the database was minimal, the optimization of the models could have only carried as far. To further develop this subject, a better databse is needed.

# KASUTATUD KIRJANDUS


1. "Definition of "JPEG"". Collins English Dictionary. Retrieved 2013-05-23.
2. Gunnery, S., & Ruben, M. (2016). Perceptions of Duchenne and non-Duchenne smiles: A meta-analysis. *Cognition and Emotion, 30*(3), 501-515.
3. Using data science to tell which of these people is lying. (2018, May 22). NewsCenter.
4. Weinberger, Sharon (May 2010). "Airport security: Intent to deceive?". Nature. 465 (7297): 412–415
5. Wang, Q., Xu, Z., Cui, X., Wang, L., & Ouyang, C. (2017). Does a big Duchenne smile really matter on e-commerce websites? An eye-tracking study in China. *Electronic Commerce Research, 17*(4), 609-626.
6. Hartwig, M., & Bond, C. F. Jr. (2011). Why do lie-catchers fail? A lens model meta-analysis of human lie judgments. Psychological Bulletin, 137, 643-659
7. The Polygraph and Lie Detection. National Research Council. 2003. ISBN 978-0-309-26392-4
8. Gunnery, S. D., Hall, J. A., & Ruben, M. A. (2013). The deliberate Duchenne smile: Individual differences in expressive control. Journal of Nonverbal Behavior, 37, 29–41.
9. Krumhuber, E. G., & Manstead, A. S. R. (2009). Can Duchenne smiles be feigned? New evidence on felt and false smiles. Emotion, 9, 807–820.
10. Okubo, M., Kobayashi, A. and Ishikawa, K. (2012) 'A Fake Smile Thwarts Cheater Detection', Journal of Nonverbal Behavior, 36(3), pp. 217–225. doi: 10.1007/s10919-012-0134-9
11. Chelouah, R., & Siarry, P. (2022). *Optimization and Machine Learning*. Newark: John Wiley & Sons, Incorporated.
12. Bourgeois, M. (2015). Guillaume Duchenne de Boulogne (1806-1875). *Annales Médico Psychologiques, 173*(3), 294-295.
13. Bi, Q., Goodman, K., Kaminsky, J., & Lessler, J. (2019). What is Machine Learning? A Primer for the Epidemiologist. *American Journal of Epidemiology, 188*(12), 2222-2239.
14. Warner HR, Toronto AF, Veasey LG, et al. A mathematical approach to medical diagnosis. Application to congenital heart disease. JAMA. 1961;177:177–183.
15. Chowdhury, R., Mahdy, M., Alam, T., Al Quaderi, G., & Arifur Rahman, M. (2020). Predicting the stock price of frontier markets using machine learning and modified Black–Scholes Option pricing model. *Physica A, 555*, 124444.
16. Lv, Q., Zhang, S., & Wang, Y. (2022). Deep Learning Model of Image Classification Using Machine Learning. *Advances in Multimedia, 2022*, Advances in multimedia, 2022, Vol.2022.

17. Shanmugamani, R. (2018). *Deep learning for computer vision* (1st ed.). Birmingham: PACKT Publishing.
18. Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature (London), 521*(7553), 436-444.
19. O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks.
20. Zeiler, M., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. *Computer Vision – ECCV 2014, 8689*(1), 818-833.
21. Coy, H., Hsieh, K., Wu, W., Nagarajan, M., Young, J., Douek, M., . . . Raman, S. (2019). Deep learning and radiomics: The utility of Google TensorFlow™ Inception in classifying clear cell renal cell carcinoma and oncocytoma on multiphasic CT. *Abdominal Imaging, 44*(6), 2009-2020.
22. Moolayil, J. (n.d.). Learn Keras for Deep Neural Networks. Berkeley, CA: Apress.
23. Ying, X. (2019). An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series, 1168(2), 22022.
24. Badillo, S., Banfai, B., Birzele, F., Davydov, I., Hutchinson, L., Kam-Thong, T., . . . Zhang, J. (2020). An Introduction to Machine Learning. Clinical Pharmacology and Therapeutics, 107(4), 871-885.
25. Basaveswara, Sai Kumar. "CNN Architectures, a Deep-dive." *Towards Data Science* 27 (2019). https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049
26. Bisong, E. (2019). Regularization for Deep Learning. In Building Machine Learning and Deep Learning Models on Google Cloud Platform (pp. 415-421). Berkeley, CA: Apress.
27. Jie, H., & Wanda, P. (2020). Runpool: A dynamic pooling layer for convolution neural network. *International Journal of Computational Intelligence Systems, 13*(1), 66-76.
28. Hara, K., Saito, D., & Shouno, H. (2015). Analysis of function of rectified linear unit used in deep learning. *2015 International Joint Conference on Neural Networks (IJCNN), 2015*, 1-8.
29. Lyon, Richard F. (2006). A brief history of 'pixel'
30. Kanan, C., & Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition?. PloS one, 7(1), e29740
31. A Pixel Is Not A Little Square! (And a Voxel is Not a Little Cube) 1 Technical Memo 6 Alvy Ray Smith July 17, 1995
32. Mita, T., Kaneko, T., & Hori, O. (2005). Joint Haar-like features for face detection. *Proceedings of the IEEE International Conference on Computer Vision, 2*, 1619-1626 Vol. 2.

33. Zeng, G. (2020). On the confusion matrix in credit scoring and its analytical properties. *Communications in Statistics. Theory and Methods, 49*(9), 2080-2093.

# LISAD

## Piltide andmestik

Käsitsi kogutude piltide andmestik on leitav GitHub’i lingi alt:

https://github.com/HenriSe/DuchenneNaeratus/tree/main/Pildid

Pilte on kokku 200 tükki ning on jaotatud .rar failidesse järgnevalt:

Elemendid, Pildi_päris, Pildid_võlts