TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Anete Holter 155200IAPB

AUTODE TUVASTAMINE JÄRJESTIKULISTEST PUNKTIPILVEDEST VOXELNETI BAASIL

Bakalaureusetöö

Juhendaja: Martin Rebane
Msc

Tallinn 2020

Autorideklaratsioon

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

Autor: Anete Holter

03.04.2020

Annotatsioon

Lõputöö eesmärk oli uurida üht punktipilve töötleva ning kahte järjestikulist punktipilve töötleva tehisnärvivõrgu mudeli üldistusvõimet autode tuvastamisel. Antud töös võeti aluseks lõputöö juhendaja Martin Rebase implementatsioon Apple Inc teadlaste poolt välja töötatud masinõppealgoritmist VoxelNet. Samuti oli vaatluse all, kuidas tehisnärvivõrgu struktuuri suurus mõjutab närvivõrgu üldistusvõimet.

Praktilise osa raames muudeti VoxelNeti struktuuri, et andmetöötlus toimuks järjestikuliste punktipilvede põhjal, mistõttu suurendati märgatavalt VoxelNeti struktuuri. Mudelite võrdluse läbiviimiseks treeniti kahte punktipilve töötlevat mudelit samade parameetritega, millega oli treenitud üht punktipilve töötlevat mudelit. Pärast treenimist viidi läbi valideerimine ning valideerimisandmete analüüs.

Lõputöö raames loodud närvivõrk on võimeline tuvastama kahelt järjestikuliselt punktipilvelt autosid. Valideerimisandmete võrdlusest selgus, et 150 epohhiga ei jõua niivõrd mahukas tehisnärvivõrk kohaneda treeningandmetega sama kiiresti kui teeb seda võrreldav väiksem närvivõrk.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 4 peatükki, 21 joonist, 3 tabelit.

Abstract

Car Detection from Consecutive Point Clouds Based on VoxelNet

The purpose of this thesis was to investigate the ability of predicting cars from point clouds using a machine learning model that processes two sequential point clouds simultaneously, and a model that processes just one point cloud at a time. The work is based on the machine learning algorithm VoxelNet by the researchers of Apple Inc. It is also examined how the size of the artificial neural network affects the ability of generalization.

As part of the practical part of this thesis, the structure of VoxelNet was changed so the model would process two sequential point clouds at a time. This addition significantly increased the size of the structure of VoxelNet. As a part of the thesis a neural network of the new structure was trained. After training, validation and analysis of validation were performed.

The neural network trained as a part of this thesis is able to detect and localize cars from two consecutive point clouds. A comparison of the two models' validation datas showed that the model trained as a part of this thesis is not able to adapt to training data as fast as the smaller neural network does.

The thesis is in Estonian and contains 33 pages of text, 4 chapters, 21 figures, 3 tables.

Lühendite ja mõistete sõnastik

Ahenduskiht	Tunnuskaartide mõõtmete kahandamise kiht, mille väljundina säiluvad kõige olulisemad tunnused (*pooling layer*)
Aktiveerimisfunktsioon	Matemaatiline funktsioon, mis määrab neuronite väljundid (*activation function*)
Hüperparameetrid	Tehisnärvivõrgu treenimisel valitud parameetrid, mille abil kohaneb mudel andmestikuga (*hyperparameters*)
Ground truth	Tõesed andmed, mille põhjal toimub närvivõrgu treenimine ning valideerimine
IOU	Mudeli poolt ennustatud piirkasti ning *ground truth* piirkasti ühisosa (*intersect over union*)
Kaal	Tehisnärvivõrgu neuronitevaheline seos (*weight*)
Keskmine täpsus	Tehisnärvivõrgu üldistusvõimet kirjeldav meetrika, mis võtab aluseks täpsuse ja saagise (*average precision*)
Kaofunktsioon	Tehisnärvivõrgu treenimisel arvutatav funktsioon, mis näitab, kuidas on mudel kohanenud treeningandmetega (*loss function*)
Kernel	Sisendpilti töötlev maatriks
Konvolutsiooniline närvivõrk	Tehisnärvivõrk, mida kasutatakse visuaalset andmetöötlust nõudvate probleemide korral (*convolutional neural network*)
LiDAR	Kõrgresolutsiooniline laserskanneerimisseade, mis väljastab punktipilvi
Neuron	Tehisnärvivõrgu vähim osa, milles toimub andmetöötlus
Peidetud kihid	Tehisnärvivõrgu kihid, mis asuvad sisendkihi ja väljundkihi vahel (*hidden layers*)
Piirikast	Asukohta ennustavate närvivõrkude ennustusel põhinev kast, mis piiritleb ära objekti asukoha (*bounding box*)
Pärilevi	Neuronite väljundite arvutamine, mille tulemusena teeb mudel ennustuse ehk annab väljundi (*forward propagation*)
Sisendkiht	Tehisnärvivõrgu kiht, mille kaudu antakse sisendandmed mudelisse (*input layer*)
Softmax-funktsioon	Funkstioon, mis annab igale võimalikule väljundile tõenäosusskoori (*softmax-function*)
Stohhastiline gradientlaskumine	Närvivõrgu optimiseerimisalgoritm, mille miniploki suuruseks on 1 (*stochastic gradient descent*)

Tagasilevi	Tehisnärvivõrgus gradientide arvutamine ning kaalude ja vabaliikmete uuendamine närvivõrgu õppimise eesmärgil (*back propagtion*)
Tensor	Matemaatiline objekt, mis sisaldab N dimensiooni. Näiteks vektor
Treenimisandmestik	Andmestik, mida kasutatakse närvivõrgu treenimise läbiviimiseks (*train set*)
Valideerimisandmestik	Andmestik, mida kasutatakse närvivõrgu valideerimiseks ehk üldistusvõime hindamiseks (*validation set*)
Voksel	Vähim kolmemõõtmeline element (*voxel*)
VoxelNet	Tehisnärvivõrgu arhitektuur, mille mudelid ennustavad objekte punktipilvedest
Väljundkiht	Närvivõrgu mudeli kiht, mis väljastab ennustuse (*output layer*)

Sisukord

1 Sissejuhatus 11
2 Probleemi taust ja meetodid 12
2.1 Masinõpe ja tehisnärvivõrgud 13
2.2 Konvolutsiooniline närvivõrk 15
2.3 Asukohta ennustav närvivõrk 16
2.4 VoxelNet 16
2.4.1 Tunnuseid õppiv närvivõrk VoxelNetis 18
2.4.2 Konvolutsioonilised vahekihid VoxelNetis 20
2.4.3 Asukohta ennustav närvivõrk VoxelNetis 20
2.5 Punktipilved 20
2.6 KITTI andmestik 23
3 Tehniline lahendus 24
3.1 Andmete laadimine 24
3.2 Treeningandmestiku täiendamine ja töötlus 25
3.3 Närvivõrgu treenimine 26
3.3.1 Treeningplatvorm 28
3.3.2 Treenimisprotsess 29
3.4 Treenitud mudeli valideerimine 30
3.5 Mudeli hindamise meetrika 32
3.6 Mudelite võrdlus 33
3.7 Järeldused ja võimalikud edasiarendused 40
4 Kokkuvõte 42
Kasutatud kirjandus 44

Jooniste loetelu

Joonis 1: Tüüpiline kahekihilise tehisnärvivõrgu struktuur [8]. 14
Joonis 2: VoxelNet kasutab töötlemata punktipilvi ja genereerib kolmemõõtmelise tuvastuse kasutades üht *end-to-end* treenitavat võrku [5]. 17
Joonis 3: VoxelNeti arhitektuur [5]. 17
Joonis 4: *VFE*-kiht 1 (*VFE Layer-1*) [5]. 19
Joonis 5: Kärbitud punktipilv KITTI andmestikust. Stseen treeningandmestikust 000011. 22
Joonis 6: Lainurkkaameraga jäädvustatud foto KITTI andmestikust. Stseen treeningandmestikust 000011. 22
Joonis 7. Autori illustratsioon modifitseeritud VoxelNeti arhitektuurist. 25
Joonis 8: Kaofunktsiooni väärtuse kahanemine treeningprotsessi vältel. 29
Joonis 9: Viimase kümne epohhi kaofunktsiooni graafik 30
Joonis 10: Treenitud mudeli tehtud ennustused. Stseen 000311 valideerimisandmestikust. 36
Joonis 11: Juhendaja poolt treenitud mudeli tehtud ennustused. Stseen 000311 valideerimisandmestikust. 36
Joonis 12: Treenitud mudeli ennustused. Stseen 000692 valideerimisandmestikust. 36
Joonis 13: Treenitud mudeli poolt auto tuvastamine inimeste seast. Stseen valideerimisandmestikust 002892 37
Joonis 14: Treenitud mudeli poolt auto tuvastamine inimeste seast. Stseen valideerimisandmestikust 003487 37
Joonis 15: Treenitud mudeli poolt õigesti tuvastatud vastutulev auto. Stseen valideerimisandmestikust 002585 38
Joonis 16: Treenitud mudeli poolt kõrvaloleval rajal oleva auto tuvastus. Stseen valideerimisandmestikust 000134 38
Joonis 17: Treenitud mudeli poolt vastutuleva auto tuvastamine. Stseen valideerimisandmestikust 002251 38
Joonis 18: Juhendaja poolt treenitud mudeli ennustus vastutulevast autost. Stseen valideerimisandmestikust 002251 39

Joonis 19: Heki tuvastamine autona. Stseen valideerimisandmestikust 000885. 39
Joonis 20: Treenitud mudeli poolt aia tuvastamine autona. Stseen valideerimisandmestikust 000969 39
Joonis 21: Juhendaja poolt treenitud mudeli märgitud piirkastid. Stseen valideerimisandmestikust 000969 40

Tabelite loetelu

Tabel 1: Valideerimise raskusastmed [20]. 34
Tabel 2: Valideerimistulemused treeningandmestikul. AP meetrikate võrdlemine. 34
Tabel 3: Valideerimistulemused valideerimisandmestikul. AP meetrikate võrdlemine. 35

1 Sissejuhatus

Viimase aastakümne üheks suurimaks uuenduseks autotööstuses on isesõitvate autode arendus. Aastal 2016 ennustati *The Guardian*'is avaldatud artiklis, et aastaks 2020 jõuavad kasutusele autonoomsed sõidukid, mille võimekust kasutatakse siiski osaliselt [1]. Sõiduki iseseisva liikumise eelduseks on ümbritseva keskkonna tajumine ning sealt ammutatud infot töödeldes otsuste tegemine.

Selleks, et ilma kasutaja panuseta liikluses hakkama saada, tuginevad autonoomsed sõidukid riist- ja tarkvarale. Sõidukid on varustatud erinevate sensoritega, mis koguvad töötamiseks vajalikku infot ümbruse kohta [2]. Üheks oluliseks sensoriks on LiDAR e. laserskanneerimisseade, mis talletab infot punktipilvede näol. Punktipilvedel on kaugussuhted täpsemalt hinnatavad ning seetõttu suudab sõiduki automaatika edukalt analüüsida oma asetust lähedalasuvate objektide suhtes. Selleks, et isesõitev auto suudaks autonoomselt liigelda, peab ta ära tundma ja lokaliseerima end ümbritsevad objektid. Selle lahendamiseks kasutatakse masinõpet, sealhulgas sügavaid tehisnärvivõrke.

Isesõitvad sõidukid analüüsivad pidevalt ümbritsevat keskkonda, see tähendab, et järjestikulisi punktipilvi, mida läbitöötada on mitmeid. Seetõttu peaks sõiduk korraga suutma analüüsida rohkem kui üht punktipilve e. töötlema sariandmeid, sealt ära tundma ja lokaliseerima erinevaid objekte.

Käesoleva töö eesmärgiks on tuvastada järjestikulistest punktipilvedest autosid, võttes aluseks tehisnärvivõrk VoxelNet, mida edasi arendades muudetakse närvivõrgu struktuuri mahukamaks. Lõputöö sihiks on võrrelda erinevusi üht punktipilve ja kahte järjestikulist punktipilve töötlevate närvivõrkude mudelite toimimises ja tuvastustäpsuses võttes aluseks keskmise täpsuse (*average precision*) meetrika. Samuti uuritakse, kuidas mõjub närvivõrgu struktuuri mahukamaks muutmine mudeli üldistusvõimele

2 Probleemi taust ja meetodid

Tänapäeval populaarsust koguv suund on masinõpe, mis kujutab endast algoritmi treenimist andmetega, kuni masinõppemudel suudab eraldada treeningandmetest olulisi tunnuseid ning kasutada õpitud üldistamisoskust reaalsete probleemide käsitlemisel [3]. Samuti on praeguse aja põnevaimaks valdkonnaks autonoomsed sõidukid ning nende arendus. Selleks, et sõiduk suudaks ilma inimjuhtimiseta liigelda, on vajalik, et auto tajuks keskkonda samahästi või isegi paremini kui teeb seda inimene. Sisendandmeid töötleva sõidukiautomaatika osa edukaks toimimiseks on oluline, et see suudaks teha mõistlikke otsuseid. Selleks tuleb sõidukile integreerida masinõppealgoritm, mis treenitakse tuvastama ja lokaliseerima erinevaid objekte, mis satuvad sensorite vaatevälja. See tähendab, on ülimalt oluline, et masinõppealgoritm õpiks tuvastama punktipilvedest teisi autosid, jalakäijaid, teepiirdeid jms. Objektide õige lokaliseerimine on eelduseks masina tee peal püsimisel ning õnnetuste ennetamisel.

Autonoomsed sõidukid koguvad infot neile paigaldatud sensorite abil, näiteks LiDAR'i, mis väljastab punktipilvi, ja stereokaamerate, mis jäädvustavad pilte [4]. LiDAR on üks usaldusväärsemaid keskkonna jäädvustamiseks mõeldud sensoreid, kuna väljastatud punktipilvedes on objektidevahelised kaugussuhted täpsed ning kvaliteetsete andmete kogumist on võimalik läbi viia ka pimedas või halbade ilmastikuolude korral [4].

On arendatud mitmeid algoritme, mis tuvastavad LiDAR'ist väljastatavate punktipilvede ja/või piltide kombinatsioonist objekte, näiteks Multi-view 3D, Pixor jm. Üheks kõige innovaatilisemaks algoritmiks on organisatsiooni Apple Inc. teadlaste Yin Zhou ja Oncel Tuzel'i poolt väljatöötatud algoritm VoxelNet. Selle algoritmi innovatiivsus tuleneb just algoritmi tunnuseid õppivast *VFE*-kihist, mis jagab punktipilvedes kujutatava kolmemõõtmelise ruumi võrdsetesuurustega vokseliteks ehk kolmemõõtmelise ruumi vähimateks osadeks, ning kodeerides iga vokseli eraldi eraldatakse tunnuseid vokselipõhiselt [5]. Antud algoritm on näidanud tähelepanuväärseid tulemusi objektituvastuses, mis on väljatoodud Zhou ja Tuzeli artiklis [5].

Uurimuslikus osas on võetud aluseks lõputöö juhendaja Martin Rebase poolt VoxelNeti implementatsioon, mida edasi arendades suurendatakse tehisnärvivõrgu struktuuri. Lõputöö eesmärk on treenida VoxelNeti, kasutades kahte järjestikust punktipilve, millest üks on ajahetkel *t-1* ja teine ajahetkel *t*. Töö käigus treenitud mudelit võrreldakse juhendaja poolt eeltreenitud mudeliga, mida on treenitud samade parameetrite ning andmestikuga. Seeläbi vaadeldakse, kas on võimalik parandada algoritmi üldistusvõimet ehk kas mudel suudab tuvastada autosid punktipilvedest paremini kui töödelda kahte järjestikust punktipilve samaaegselt. Mudelite võrdlemine annab aimu, kuidas mõjutab tehisnärvivõrgu struktuuri mahukus mudelite üldistuvõimet.

2.1 Masinõpe ja tehisnärvivõrgud

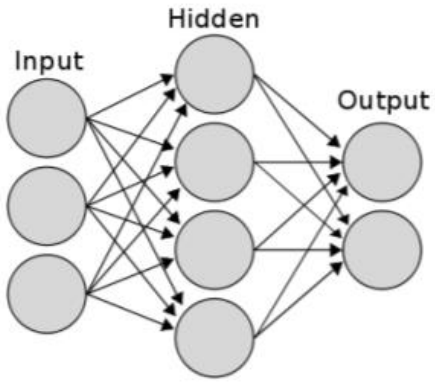
Viimase aastakümne üheks põnevamaks valdkonnaks tehnikas on kujunenud masinõpe, mis on saanud võimalikuks arvutusvõimsuse ja andmemahtude suurenemise ning innovaatiliste masinõppealgoritmide arenduste tõttu [3]. Masinõpe on tehisintellekti vorm, mis võimaldab süsteemil andmetest õppida, tuvastada andmetes mustreid ning teha seda minimaalse inimsekkumisega [6].

Tehisnärvivõrk on masinõppealgoritm, millega imiteeritakse, kuidas inimese aju töötab ning probleemidele läheneb. Tehisnärvivõrgud kasutavad omavahel täielikul ühendatud sõlmi ehk neuroneid, et õppida ja järeldada suhteid vaadeldavatest andmetest [7]. Närvivõrgud on kujundatud simuleerima inimaju tööd, kuna sellisel viisil on võimalik treenida arvuteid mõistma abstraktsioone ja probleeme, mis on halvasti määratletud [7].

On masinõppemudeleid, mida implementeeritakse ja treenitakse eraldi, ning seejärel mudeli erinevaid osi kasutatakse koos *pipeline*'ina e. järjestikku seatud süsteemidena. Sellele vastanduv masinõppemudelit iseloomustab *end-to-end* süvaõpe (*deep learning*). Mõiste *end-to-end* süvaõpe kujutab endast lihtsustatult ühtset tehisnärvivõrku, kuhu antakse sisend ja saadakse väljund ilma vahepealsete etappideta ning kogu mudelit treenitakse ühtselt. Antud lõputöös kasutatav VoxelNeti masinõppealgoritm on samuti *end-to-end* tüüpi algoritm.

Närvivõrgud koosnevad kihtidest: sisendkiht (*input layer*), üks või enam peidetud kihti (*hidden layer*), ja väljundkiht (*output layer*). Kihid koosnevad omavahel täielikult ühendatud sõlmedest ehk neuronitest [3].

Järgnev joonis (Joonis 1) kirjeldab tüüpilist kahekihilist tehisnärvivõrku, kus on kolm sisendit, üks peidetud kiht nelja neuroniga ja viimasena väljundkiht, kus on kaks neuronit. Kolm sisendit tehisnärvivõrgus tähendab, et närvivõrk võtab sisse kolm tunnusjoont ning üritab ennustada proovides üldistada nende joonte järgi. Kaks neuronit väljundkihis tähendab, et võimalikke tulemusi on kaks, millest üks osutub valituks.



Joonis 1: Tüüpiline kahekihilise tehisnärvivõrgu struktuur [8].

Kui tehisnärvivõrk koosneb rohkem kui ühest peidetud kihist, on tegemist sügava närvivõrguga (*deep neural network*) [3]. Süvaõpet on mõistlik kasutada, kui üritatakse leida mustreid struktureerimata andmetest [7]. Struktureerimata andmeteks võivad olla pildid, tekst, heli [3] ja ka sensoritest tulenevad andmed [9]. Sealhulgas on struktureerimata andmeteks ka punktipilved, mida antud lõputöös objektituvastuseks kasutatakse.

Enne närvivõrgu treenimise juurde asumist määratakse neuronite vaheliste seoste tugevused ehk kaalud (*weight*) [3] ja ennustuste nihe tegelikest tulemustest ehk vabaliikmed (*bias*) [3]. Vabaliikmete ja kaalude valik toimub juhuslikult, kuna sellisel juhul toimub närvivõrgu treenimine kiiremini kui nullidega initsialiseerimisel [3].

Närvivõrku võetakse andmed sisse läbi sisendkihi. Sisendkihist liigub info järgmisesse neuronisse edasi tänu pärilevile (*forward propagation*) [3]. Tehisnärvivõrgus toimub infotöötlus kaalude ja vabaliikmete poolt karaktiseeritud neuronite ja nende ühenduste kaudu. Sisendid võetakse neuronisse. Seejärel summeeritakse vastavate kaalude ja sisendite korrutis, hiljem liidetakse juurde vabaliige. Saadud tulemus rakendatakse eelvalitud aktiveerimisfunktsioonile ning selgub, kas neuron aktiveeritakse. Seejärel edastatakse informatsioon järgmistele ühendatud neuronitele kuni jõutakse väljundkihini, millel on üks neuron iga võimaliku tulemuse jaoks [10]. Väljundkihis tekitatakse tulemus vastavatele sisenditele.

Närvivõrkude ja masinõppe põhiliseks eeliseks on nende õppimisvõime ja omadus end täiustada tulemusi iga ennustusega [10]. Õppimine tehisnärvivõrkude mõistes tähendab, et kaalud (*weight*) ja vabaliikmed (*bias*), mis määravad neuronite omavahelised seosed, valitakse järjest täpsemini, kuni tehisnärvivõrgu väljund läheneb sisendi tegelikule väljundile ehk kuni tulemus on õige või piisavalt lähedane sellele [10]. Selle saavutamiseks kasutatakse mitmeid optimeerimisalgoritme, nagu stohhastiline gradientlaskumine, Adam'i algoritm, RMSprop jm.

2.2 Konvolutsiooniline närvivõrk

Konvolutsioonilised närvivõrgud on tehisnärvivõrgud, mida kasutatakse eelkõige pildi- ja videotuvastuses, piltide klassifitseerimisel ja muudel arvutinägemise (*computer vision*) probleemides [3]. Konvolutsioonilised piltide klassifitseerijad saavad sisendiks pildi, töötlevad seda ning liigitavad selle etteantud kategooriasse [3]. Konvolutsioonilistes närvivõrkudes läbib iga sisend mitmeid konvolutsioonillisi kihte filtritega ehk kerneleid, ahenduskihte (*pooling layer*), täielikult ühendatud kihi (*fully connected layer*) ja rakendavad *softmax*-funktsiooni, et klassifitseerida objekti [3]. Objekti klassifitseerimisel väljastatakse igale võimalikule väljundile tõenäosus [3]. Konvolutsiooniliste närvivõrkude eesmärgiks on vähendada sisendpilti kujule, mida oleks lihtsam töödelda, ilma et edukaks klassifitseerimiseks vajaminevad pildi olulised tunnused kaduma läheks [11].

Konvolutsioonilised kihid eraldavad sisendpildist tunnuseid (*feature*) rakendades sisendile järjestikuliselt kerneleid ning seeläbi saades tunnuste kaardi (*feature map*) [3].

Tunnuskaardi dimensioonide vähendamiseks kasutatakse ahenduskihte, et vähendada algoritmi tööks vajaliku arvutusvõimsuse suurust [3]. Ahenduse eesmärgiks on eraldada kõige dominantsemad tunnused [11]. Ahendamiseks on kaks viisi: *max-pooling* ja *average pooling*. *Max pooling'*u korral tagastatakse kerneli all oleva pildi osa kohta maksimaalne väärtus ning *average pooling*'u korral keskmine väärtus [11].

Pärast ahenduskihti saadetakse sisend pärilevi närvivõrku ning tagasilevi rakendatakse pärast igat treeningiteratsiooni [11]. Klassifitseerimiseks kasutatakse *softmax* klassifitseerimist, mis tagastab tõenäosused igale võimalikule väljundile. Kõigi väljastatud tõenäosuste summa peab olema 1.

2.3 Asukohta ennustav närvivõrk

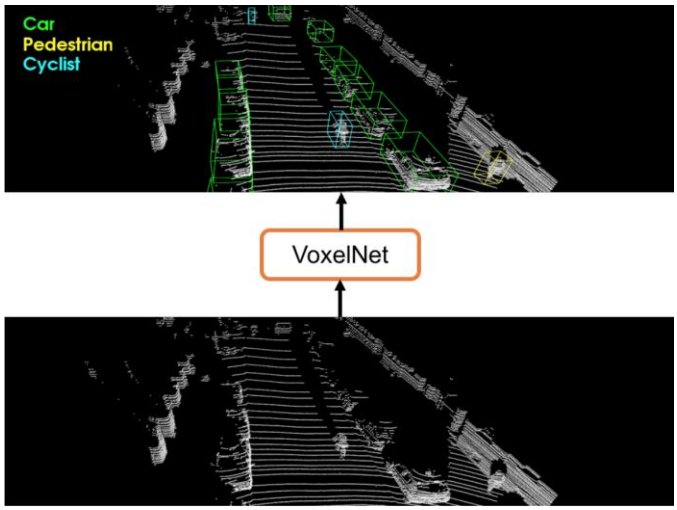
Asukohta ennustav närvivõrk (*region proposal network*) on algoritm objektide asukohtade ennustamiseks [3]. *RPN* võtab sisendiks mistahes suurusega pildi ja väljastab kogumi, mis koosneb neljakandilistest objekti asukoha ennustustest ehk ankurpunktidest (*anchor point*) [3] ning nendega seotud objektiivse hinnangu objekti kuuluvuse kohta [12]. Asukoha ennustuste genereerimiseks libistatakse aken üle konvolutsioonilise tunnuskaardi, mis on viimase konvolutsioonilise kihi väljundiks.

RPN'e saab treenida *end-to-end* viisil kasutades tagasilevi ja stohhastilist gradientlaskumist [12]. Iga miniplokk koosneb vaid ühest pildist ning positiivsetest kui ka negatiivsetest ankurkastide näidetest [12].

2.4 VoxelNet

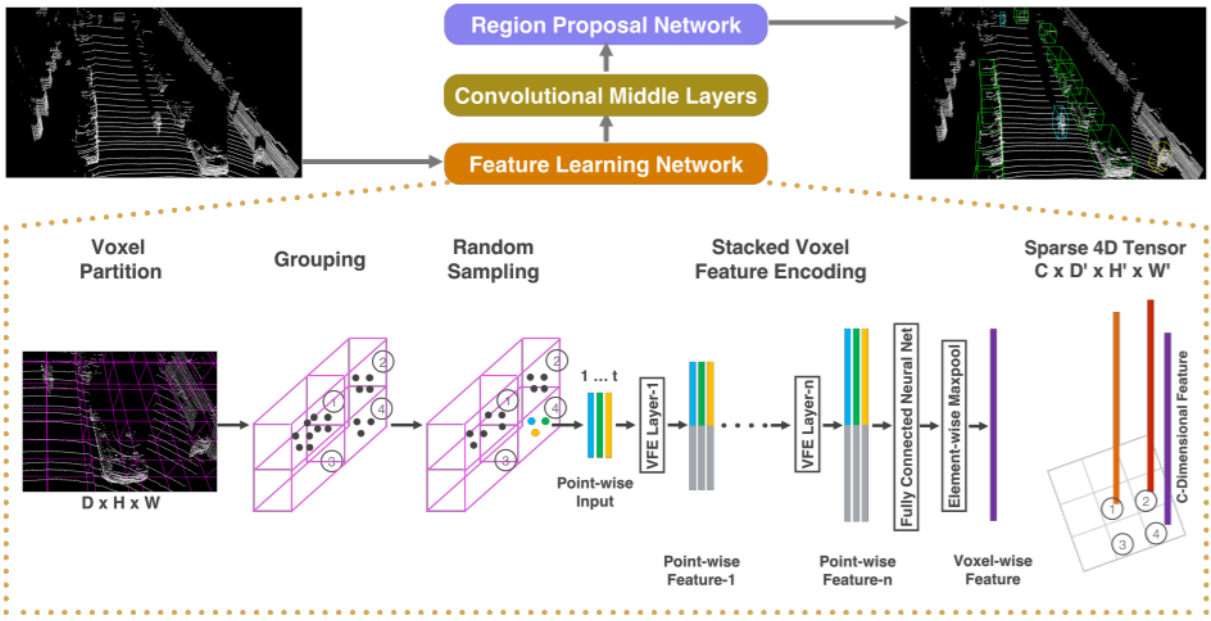
VoxelNet on *end-to-end* tehisnärvivõrk objektide tuvastamiseks ja lokaliseerimiseks kolmemõõtmelistest punktipilvedest [5]. VoxelNeti teeb eriliseks see, et algoritm õpib eristatavaid tunnuseid punktipilvedest ja samaaegselt ennustab täpseid kolmemõõtmelisi piirikaste (bounding box) [5].

Järgneval joonisel (Joonis 2) on toodud näide töötlemata punktipilvest, mis on VoxelNetile antud sisendiks ning tulemuseks on piirikastidega märgistatud objektid antud punktipilves.



Joonis 2: VoxelNet kasutab töötlemata punktipilvi ja genereerib kolmemõõtmelise tuvastuse kasutades üht *end-to-end* treenitavat võrku [5].

Alloleval joonisel (Joonis 3) on kujutatud VoxelNeti arhitektuur, mis koosneb kolmest funktsionaalsest plokist: tunnuseid õppiv närvivõrk (*feature learning network)*, konvolutsioonilised vahekihid (*convolutional middle layers*) ja objekti asukoha ennustav närvivõrk (*region proposal network*) ehk *RPN* [5].



Joonis 3: VoxelNeti arhitektuur [5].

VoxelNet jagab punktipilve võrdselt paigutatud kolmemõõtmelisteks vokseliteks (*voxel*) ja teisendab läbi voksli tunnuseid kodeeriva (*voxel feature encoding*) kihi iga voksli punktid ühtseks tunnuseks. Sellisel viisil on punktipilved kodeeritud deskriptiivse

mahulise esitusena, mis võimaldab hiljem *RPN*'is genereerida tuvastusi ja lokaliseerimisi [5].

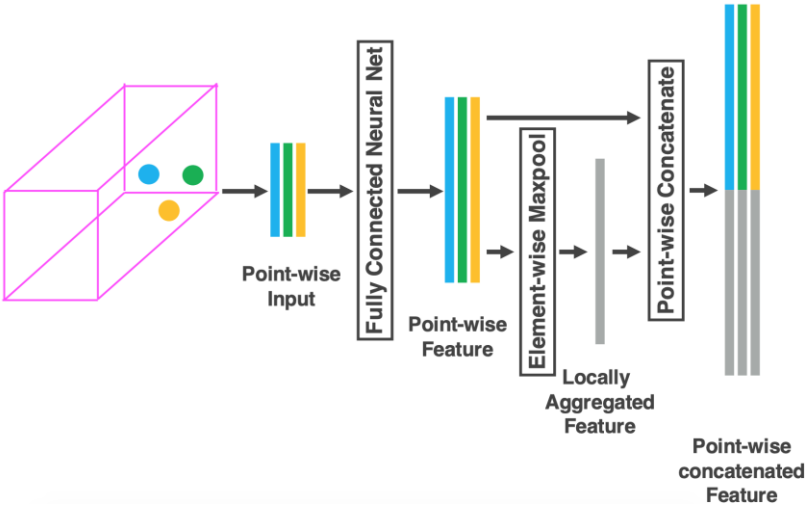
Voksel või mahuline piksel (*volume pixel*) on vähim eristatav kolmemõõtmeline element kolmemõõtmelises objektis. See on mahuline element, mis väljendab mingit kindlat väärtust kolmemõõtmelise ruumi võrgustikus, kuid nad ei sisalda infot oma asukoha kohta kolmemõõtmelises ruumis. Voksli asukoht on kirjeldatav selle positsioonist lähedalasuvate vokselite suhtes [13].

2.4.1 Tunnuseid õppiv närvivõrk VoxelNetis

VoxelNeti esimene andmetöötlusega tegelev plokk on tunnuseid õppiv närvivõrk, milles kasutatakse tunnuste eraldamiseks närvivõrku ja väljastatakse neljamõõtmeliste tensorite kogum [5]. Selline viis võimaldab töödelda mahukaid punktipilvi, mille tunnuste konvolutsiooniliselt eraldamine oleks ebaotstarbekas, kuna nõuaks liialt arvutusvõimsust [5].

Tunnuseid õppivasse kihti antakse sisse punktipilv, kus jagatakse kolmeõõtmeline ruum võrdseteks vokseliteks [5]. Seejärel grupeeritakse punktid vastavalt sellele, millises vokselis need asuvad [5]. Tänu erinevatele faktoritele on LiDAR'i genereeritud punktipilv hõre ja punktide tihedus on kohati ruumis erinev. Seetõttu pärast grupeerimist on erinevates vokselites erinev arv punkte [5]. VoxelNeti arhitektuuri kirjeldaval joonisel (Joonis 1) tunnuseid õppiva närvivõrgu plokis on näha grupeerimisosas, et esimesel vokselil (*Voxel*-1) on märgatavalt rohkem punkte kui teisel (*Voxel-2*) või neljandal (*Voxel-4*) vokselil ning kolmandal vokselil (*Voxel*-3) puuduvad punktid üldse [5].

Kõrglahutusega LiDAR punktipilved koosnevad umbes sajatuhandest punktist. Selliste punktipilvede kõikide punktide töötlemine pole mitte ainult arvutusvõimsuselt kulukas, vaid ka punktide asetsemise tihedus võib soovimatult mõjutada ennustuste tulemust [5]. Selle probleemi vältimiseks ahendatakse andmeid võttes juhuslik valim. Selleks valiti vokselitest välja juhuslik arv T punkte vokselitest, kus on rohkem kui T arv punkte [5]. Selline ahendusstrateegia vähendab arvutusteks vajaminevat ressurssi, vähendab vokselites olevate punktide arvu ebavõrdsust ja lisab treenimisele variatsiooni [5].



Joonis 4: *VFE*-kiht 1 (*VFE Layer-1*) [5].

Järgnevalt on kasutusel innovatiivne tunnuste kodeerimise ehk *VFE* (*Voxel Feature Encoding*) kihtide jada, mis on väljaarendatud VoxelNeti loojate Tuzel'i ja Zhou poolt [5]. *VFE*-kihi struktuuri kirjeldab eelolev joonis (Joonis 4). *VFE*-kihi sisendiks on mitte-tühi voksel, mille punktide asetuste järgi arvutatakse kohalik keskväärtus [5]. Seejärel lisatakse igale punktile juurde tema nihe arvutatud keskväärtusest ning punkt töödeldakse tunnusruumi läbi täielikult ühendatud närvivõrgu (*fully connected neural net*), mis koosneb lineaarsest kihist (*linear layer*), ploki normaliseerimise (*batch normalization*) ehk *BN*-kihist ja mittenegatiivse lineaarfunktsiooni (*rectified linear unit*) ehk *ReLU*-kihist [5]. Kõik mitte-tühjad vokselid on kodeeritud samamoodi ja omavad samu parameetreid täielikult ühendatud võrgus [5]. Täielikult ühendatud võrgust väljastatakse iga punkti kohta punktipõhine tunnus [5]. Pärast punktipõhiste tunnuste (*point-wise feature*) kujutuste saamist kasutatakse elemendipõhist *max-pooling*'ut üle kõikide vokseliga seotud tunnuste ning saadakse lokaalselt agregeeritud tunnus (*locally aggregated feature*) [5]. *Max-pooling* tähendab, et tunnustest valitakse välja väärtus, mis on maksimaalne ehk parim [3]. Seejärel koondatakse kõik punktipõhised tunnused ja lokaalselt agregeeritud tunnus. See tähendab, et *VFE*-kihi väljundiks on punktipõhiste tunnuste kogum, kus igal punkti kohta on ka lokaalselt agregeeritud tunnus [5]. Tänu sellele *VFE*-kihtide kuhjamine e. *SVFE* (*stacked voxel feature encoder)* kodeerib punktisuhteid vokselisiseselt ja võimaldab lõpptunnusel õppida kirjeldatavat informatsiooni objekti kuju kohta [5].

Ainult mitte-tühjade vokselite töötlemisel saadakse vokselitunnuste kogumik, kus iga tunnus on unikaalse seosega vokselisiseste ruumikoordinaatidega. Saadud vokselipõhine kogum on esitatav hõreda neljadimensioonilise tensorina [5]. Selline esitusviis vähendab hulgaliselt algoritmi arvutusvõimsuse vajadust, seda eriti tagasilevil (*backpropagation*), ja on murranguline omadus VoxelNeti võimekas implementatsioonis [5].

2.4.2 Konvolutsioonilised vahekihid VoxelNetis

Keskmised konvolutsioonilised kihid töötlevad tunnuseid õppiva närvivõrgu väljundeid ehk tensoreid ja väljundiks on tunnuskaart (*feature map*) [5]. Iga konvolutsiooniline vahekiht rakendab vastavas järjekorras kolmemõõtmelist konvolutsiooni, ploki normaliseerimise ehk *BN*-kihti ja mittenegatiivset lineaarfunktsiooni ehk *ReLU*-kihti [5]. Konvolutsioonilised vahekihid koondavad vokselipõhiseid tunnuseid, millega lisatakse objekti kuju kohta konteksti [5].

2.4.3 Asukohta ennustav närvivõrk VoxelNetis

VoxelNeti asukohta ennustaval närvivõrgul (*region proposal network*) ehk *RPN*'il on sisendiks konvolutsiooniliste vahekihtide väljund [5]. *RPN* koosneb VoxelNetis kolmest täielikult konvolutsioonilisest kihtidest koosnevast plokist [5]. Iga ploki esimeses osas ahendatakse sisendtunnuskaardi dimensioone sammuga 2 ja seejärel rakendatakse konvolutsioone sammuga 1 [5]. Pärast igat konvolutsioonikihti, rakendatakse ploki normaliseerimise ja *ReLU*-operatsioonid [5]. Seejärel laiendatakse iga ploki väljundit, mis hiljem liidetakse kõrgresolutsiooniliseks tunnuskaardiks [5]. Tunnuskaart kaardistatakse soovitud õpiväljunditeks: tõenäosusskoorikaardiga ja regressioonikaardiga [5].

2.5 Punktipilved

Punktipilved on kolmemõõtmelised andmed, mida saadakse enamasti fotogrammeetria või ümbruskonna tajumise kaudu. Fotogrammeetria kasutab mitmeid erinevate nurkade alt jäädvustatud piltidekombinatsioone, mida töödeldes on võimalik genereerida punktipilv [14]. Hoopis moodsam viis on ümbruskonna tajumine erinevate satelliitide või sõidukitele paigaldatud sensorite, näiteks LiDAR'i kaudu. Tänapäeval rakendatakse LiDAR'i jäädvustusi ehk punktipilvi erinevates eluvaldkondades. Näiteks geoinfosüsteemide arenduses salvestades maapinnavorme täpsemini kui kunagi varem.

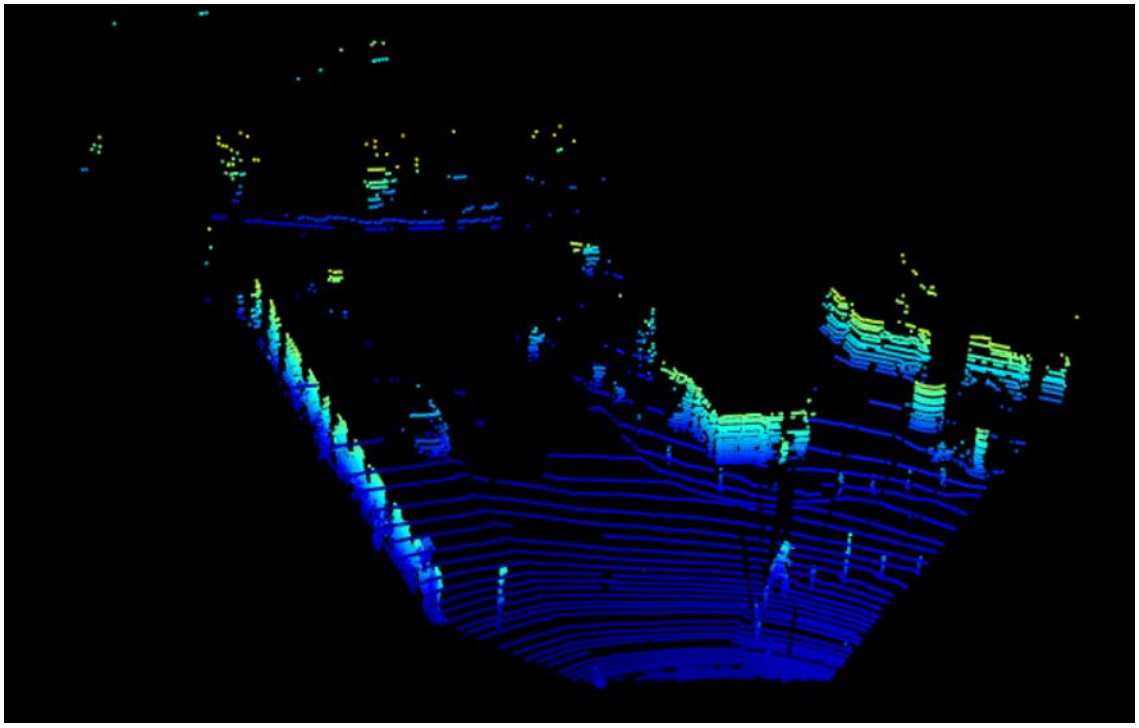
Samuti jäädvustatakse LiDAR'itega kolmemõõtmelisi keskkondi erinevatest linnadest. Punktipilvede töötlemine on kasutusel autonoomsetes sõidukites, 3D modelleerimises, 3D printimises, meditsiinis, arhitektuuris ja ka virtuaalreaalsuse loomes [15].

LiDAR on laserskanneerimisseade, mille tööpõhimõte sarnaneb radarile, kuid raadiolainete asemel on kasutusel infrapuna valgusimpulsid: saadetakse välja valgusimpulsse ning arvutatakse täpselt välja nende teekonna pikkus seadmest objekti pinnani kasutades valguse levimiskiirust. LiDAR jäädvustab miljoneid täpseid kaugusmõõtmisi igal sekundil, millest on võimalik genereerida kolmemõõtmeline andmestik ümbritsevast keskkonnast. LiDAR'ite eeliseks on selle täpsus ja võimekus kaardistada keskkonda täpselt ka pimedas [4]. LiDAR seadmed salvestavad andmed punktipilvedena.

Punktipilv, mida LiDAR sensorid jäädvustavad, on punktide kogum ruumis, mis kujutab kolmemõõtmelist kujundit või tunnust. Igal punktipilves oleval punktil on kolm koordinaati, mis väljendavad punkti asukohta punktipilves. Paljude punktide koosasetsemise mõjul hakkavad visuaalselt välja joonistuma kujundid või tunnused. Punktipilved sisaldavad infot ainult kujutatava objekti välispinnalt. [16] . Punktipilvi peetakse üheks lihtsamaks kolmemõõtmeliste objektide kujutamisviisiks, kuna need koosnevad ainult töötlemata koordinaatidest kolmemõõtmelises ruumis. [17]

Järgnevatel piltidel on kujutletud näidet punktipilvest (Joonis 5) ning samast stseenist jäädvustustatud foto (Joonis 6) näol. Punktipilvest on näha, kus asetsevad jalakäija ning tagapool olev auto. Ka fotolt on näha needsamad objektid, kuid nende kaugussuhted pole visuaalselt täpselt hinnatavad. Punktipilvedel on objektidevahelised kaugused täpsemini hinnatavad, kuna säilinud on punktidevahelised kaugussuhted kolmemõõtmelises ruumis. Antud punktipilve visualiseerimiseks on kasutatud veebipõhist visualiseerimistööriista *Online LIDAR point cloud viewer*[1].

[1] http://lidarview.com/



Joonis 5: Kärbitud punktipilv KITTI andmestikust. Stseen treeningandmestikust 000011.

Joonis 6: Lainurkkaameraga jäädvustatud foto KITTI andmestikust. Stseen treeningandmestikust 000011.

Punktipilvede esitamine jaguneb kaheks: ASCII- ja binaarformaadid [18]. ASCII-formaatides on punktipilv kirjeldatud kasutades teksti [18]. Kõige rohkem ASCII-formaatidest kasutatakse XYZ, OBJ, PTX ja ASC. Binaarsüsteemid salvestavad infot otse binaarkoodis ning arvutuslikult on tõhus esitada punktipilvi binaarformaadis. Enimkasutatavad binaarformaadid on BIN, FLS (*Faro*), PCD (*Point Cloud Library*) ja LAS [18]. Tekstipõhised formaadid on mahukamad ja sisaldavad palju taustinformatsiooni (*metadata*). Binaarformaadis olevad failid on kompaktsemad, sisaldavad rohkem infot ja nende töötlemine ja visualiseerimine on vähem ajakulukas,

kuna nende töötlemine on võimalik osade kaupa [18]. KITTI andmestikus olevad punktipilved on salvestatud binaarformaadis [19] ning kasutatav VoxelNeti implementatsioon võtab sisendiks binaarformaadis punktipilvi.

2.6 KITTI andmestik

Hästi toimiva masinõppealgoritmi eelduseks on selle treenimisprotsess. Treenimisjärgus on ülimalt oluline piisava koguse treeningandmete töötlemine närvivõrgu poolt, et algoritm suudaks neist üldistada. Väheste treeningandmete korral võib tekkida algoritmil alasobitumine, mistõttu ei suuda algoritm õppida ära olulisi tunnuseid, mida edukaks tuvastamiseks ja lokaliseerimiseks tarvis on. On mitmeid võimalikke andmestikke, mida kasutada tehisnärvivõrgu treenimiseks, näiteks nuScenes, KITTI, Oxford Robotcar jm. Andreas Geiger'i *et al* poolt on väljatöötatud KITTI andmestik, mis koosneb LiDAR'i poolt genereeritud punktipilvedest ja kõrgresolutsiooniga kaamerate piltidest [20]. KITTI andmestikku kogutud andmed on mitmekesised: päriselulised liiklussituatsioonid varieeruvad kiirteede, maakohtade ja ka linnasiseste stseenidega, kus on nii liikuvaid ja ka staatilisi objekte [20]. Andmestik koosneb ka objektide märgistest (*label*), mis on olulised treeningprotsessiks, ning punktipilved on salvestatud Velodyne kaameraga binaarformaati [20].

Igale etalonile genereeritakse õige väljund (*ground truth*). Genereeritud punktipilved kajastavad ümbruskonna kohta infot 360 kraadi ulatuses. Kaamerate salvestatud pildid kajastavad vaid 120 kraadi ulatuses. Kuna KITTI andmestikus on märgised vaid kaamera vaatevälja ulatuvate objektide jaoks, on vajalik punktipilvede töötlemine, kus lõigatakse punktipilvest ära kogu selline osa, mis ei kajastu kaamera vaateväljas.

Lõputöö käigus loodav masinõppe mudel treenitakse ning valideeritakse KITTI andmestikul põhineval andmestikul, mida on VoxelNeti masinõppealgoritmi jaoks kohandatud ning töödeldud juhendaja Martin Rebase poolt. Praktilise osa raames täiustatakse seda andmestikku, lisades sinna eelneva kaadri iga punktipilve kohta.

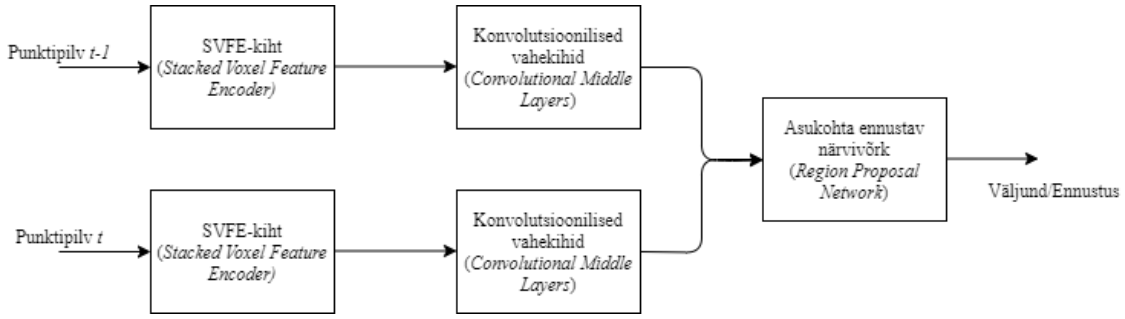
3 Tehniline lahendus

Tehnilise lahenduse teostuseks võeti aluseks antud lõputöö juhendaja Martin Rebase implementatsioon VoxelNetist ning täiustati seda, et oleks võimalik töödelda kaht järjestikust punktipilve samaaegselt. Implementatsiooni täiustamine hõlmab VoxelNeti SVFE-kihi ja konvolutsiooniliste vahekihtide kahekordistamist, mis muudab VoxelNeti arhitektuuri küllaltki mahukaks. Treenitud mudeli üldistusvõime täpsuse võrdlemiseks võeti aluseks lõputöö juhendaja poolt treenitud üht punktipilve töötlev mudel, mida on eelnevalt treenitud samade parameetrite ja andmestikuga.

3.1 Andmete laadimine

Zhou ja Tuzel'i poolt välja töötatud VoxelNet on ülesehitatud sellele, et andmestik laetakse sisse ning korraga on töötluses vaid üks punktipilv. Ka Martin Rebase implementatsioon VoxelNetist on sarnase ülesehitusega. Käesoleva lõputöö praktilises osas modifitseeriti lõputöö juhendaja VoxelNeti lahendust töötlema kaht järjestikust punktipilve.

Selleks, et töötlus toimuks mitme punktipilve põhjal, muudeti esmalt VoxelNeti struktuuri. Esialgu kahekordistati sisendit, mistõttu saadeti järjestikuliste pilvede asemel mudelisse sama punktipilv, et vigade silumine (*debugging*) oleks implementeerides võimalikult lihtne. Duplikeeritud andmed saadeti sisse nii *SVFE*-kihti (*Stacked Voxel Feature Encoding*), konvolutsioonilistesse vahekihtidesse ning seejärel liideti konvolutsiooniliste vahekihtide väljundid kokku ning saadeti *RPN*-kihti üks sisend. Pärast konvolutsioonilisi vahekihte oli oluline kahe erineva punktipilve tulemused liita, kuna kahe erineva *RPN*-kihi kasutamine oleks olnud liiga suure RAM-mälu kasutuse tõttu hetkel võimatu. Pärast *RPN*-kihti väljastatakse üks väljund, mis on ühe treeningploki tulemusteks. Järgneval pildil (Joonis 7. Autori illustratsioon modifitseeritud VoxelNeti arhitektuurist.) on kirjeldatud uuendatud VoxelNeti arhitektuur, kus on kahekordistatud *SVFE*-kihti ning konvolutsioonilisi vahekihte ning kokkuliidetud tulemus on asukohta ennustava närvivõrgu sisendiks.



Joonis 7. Autori illustratsioon modifitseeritud VoxelNeti arhitektuurist.

Pärast algoritmi töö korrektsuses veendumist muudeti treenimisprotsessi, kus toimus andmete sisselaadimine ning treenimine. Treenimisprotsess hõlmab epohhide läbimist, tehisnärvivõrgu mudelisse sisendandmete andmisega, kaofunktsiooni arvutamisega. Seetõttu oli vaja seda modifitseerida, et andmeid laetakse ühestes miniplokkides, kuna korraga töödeldavate punktipilvede laadimine muutub muidu mälukasutuselt liiga mahukaks. Uuendatud struktuuriga VoxelNeti implementatsioon on saadaval Tallinna Tehnikaülikooli GitLabi repositooriumis[1].

3.2 Treeningandmestiku täiendamine ja töötlus

Närvivõrgu mudeli treenimisel on üheks olulisemaks aspektiks treeningandmestik, mille põhjal treenitav mudel õpib. Antud lõputöö raames loodava mudeli treenimiseks pakkus lõputöö juhendaja välja KITTI andmestikul põhineva VoxelNetile kohandatud treeningandmestiku. Kuna VoxelNeti esialgne lahendus töötles igast stseenist vaid üht punktipilve, siis oli ka välja pakutud andmestikus ühe stseeni kohta üks punktipilv. Selleks, et oleks võimalik kasutada antud andmestikku, oli tarvis igale stseenis olevale punktipilvele ajahetkel *t* leida eelnev punktipilv ajahetkel *t-1*.

Kuna algses treeningandmestikus on lisaks punktipilvedele ka sellele vastav kaamerapilt, siis loodi programmikood, mis itereeris üle treeningandmestiku piltide ajahetkel *t*, otsides KITTI töötlemata andmestikust (*raw data*) identset pilti kasutades pildi räsimist e. *hash*-funktsiooni.

[1] https://gitlab.cs.ttu.ee/anholt/iapb

Kahe pildi võrdlemiseks võeti mõlemast pildist räsifunktsiooni väärtus kasutades *dHash*'i. Piltide võrdlemiseks kasutatud räsifunktsioon *dHash* annab väga vähe vale-positiivseid tulemusi [21]. Räsifunktsioon *dHash* on praktiliselt identne *aHash* funktsioonile, kuid annab palju paremaid tulemusi, kuna dHash arvutab pikslitevahelisi gradiente. Alternatiivselt võrdleb funktsioon *aHash* iga piksli väärtust pildi keskmise väärtusega ning annab väärtuse vastavalt sellele, kui erinev on pikslivääртus pildi keskmisest väärtusest [21].

Töötlemata andmed laeti alla KITTI andmestiku loojate ametlikult lehelt[1]. Töötlemata andmestik hõlmas must-valgeid ja värvilisi kaamerapilte, töötlemata punktipilvi ja kalibreerimisinfot. Kuna olemasolevad pildid ajahetkel *t* on värvifotod, itereeriti üle vaid vastavatest piltidest, mis olid samuti värvilised. Andmestiku mahukuse tõttu oleks olnud ebaotstarbekas kogu andmestikust korduvalt üle itereerida. Vähendatud andmestik, mida puuduvate andmete otsimiseks kasutati, oli mahult 66.7 Gb.

Identse pildi ajahetkel *t* leidmisel võeti järjenumbri alusel eelmine pilt ja punktipilv ajahetkel *t-1* ja lisati need vastavatesse eelvalitud kaustadesse. Kuigi pilte VoxelNeti lahendus treenimisel ei kasuta, on piltide leidmine valideerimiseks oluline, kuna piltidele joonistatakse piirkastid. Mudeli töö hindamiseks on oluline mudel ka treeningandmestikul valideerida, et tuvastada mudeli töös võimalikke probleeme.

Treeningandmete leidmisel oli tarvis leitud punktipilved kärpida (*crop*), et punktipilvedes kajastuks vaid selline osa, mis on ka kaamerapildile jäädvustunud.

Lõplik treenimisandmestik, mida lõputöö raames loodava mudeli treenimiseks kasutati, koosnes 3713 stseenist, mis hõlmas enda alla iga stseeni kohta 2 punktipilve ajahetkedel *t-1* ja *t*, kaamerapilti ajahetkest *t*, kalibreerimisandmeid. Kasutatud treenimisandmestik oli mahult 4.74 Gb.

3.3 Närvivõrgu treenimine

Tehisnärvivõrgud toimivad küll ise õppides, kuid selleks, et õpe toimuks efektiivselt, on vaja määrata hüperparameetrid (*hyperparameters*). Hüperparameetrid on tehisnärvivõrgu

[1] http://www.cvlibs.net/datasets/kitti/raw_data.php

parameetrid, mis määravad selle struktuuri, ja muutujad, mis määravad, kuidas närvivõrku treenitakse [22]. Mudeli hüperparameetrid määratakse enne treenimisprotsessi algust [22]. Õigete hüperparameetrite valik mängib võtmerolli masinõppealgoritmi edukuses [22].

Käesoleva lõputöö käigus treenitud närvivõrgu parameetrite valik lähtus juhendaja eelnevalt väljapakutud mudelist, kuna sellisel juhul on tulemused võrreldavad. Mudelit treeniti tuvastama ainult autosid, kuna eeltreenitud mudel oli samuti treenitud tuvastama ja lokaliseerima ainult autosid.

Õpisamm on parameeter, mis määrab, kui kiiresti algoritm kohaneb andmetega. Kui õpisamm on liiga väike, ei pruugi mudel õppida andmetes olevaid olulisi mustreid ning treenimisprotsess vajab palju aega ja treeningandmeid [3]. Vastandlikult liiga suure õpisammu korral kohandub algoritm liiga kiiresti tehes üldistusi vaid kõige viimaste treenindandmete põhjal ja algoritm ei pruugi koonduda miinimumis [3]. Esialgseks õpisammuks kujunes 0.001, mida hakkas vastavalt treeningprotsessile muutma Adam optimeerimisalgoritm.

Optimeerimisalgoritmiks valiti Adam algoritm, mis kuulub adaptiivseks õpisammu optimeerimisalgoritmiks, mis on väljatöötatud konkreetselt tehisnärvivõrkude treenimiseks [23]. Adam algoritmi kasutatakse klassikalise stohhastilise gradientlaskumise asemel, kuna uuendab õpisammu treenimise ajal [23].

Mini-plokkide suurus määrab, kui suur on treeningandmete arv ühes epohhis [3]. Suurem mini-ploki suurus teeb algoritmi aeglasemaks, kuna arvutuslikult on tehete arv suurem [22]. Väiksemad mini-plokkide suurused tähendavad, et kaofunktsioonis on rohkem kõikumisi, mis takistab algoritmi peatumist lokaalses miinimumis [22]. Antud lõputöös on ploki suuruseks 1, kuna paralleelselt mitme treeningandme töötlemine nõuab liialt palju arvutusvõimsust.

Epohhide arvu valimiseks pööratakse tähelepanu valideerimisveale [22]. Intuitiivne võte on treenida seni, kuni valideerimisviga langeb, ning siis kasutada varast lõpetamist (*early stopping*) [22]. Varase lõpetamise ideeks on lõpetada treenimisprotsess, kui kaofunktsioon pole langenud üle 10-20 epohhi [22]. Kuna käesolevas lõputöös oli oluline võrrelda juba treenitud mudelit, treeniti loodavat mudelit samuti 150 epohhi.

Tehisnärvivõrkude toimimiseks on oluline, et need õpiksid väljundit ennustama. Mudeli peidetud ühikute ehk neuronite arv iseloomustab mudeli õppimisvõimet [22]. Lihtsamate ülesannete jaoks on vajalik väiksem arv peidetud ühikuid, kuid mida mahukam on algoritmi funktsioon, seda suuremat õppimisvõimet on sellel tarvis. Seega, tähendab see keerulised ülesanded nõuavad suuremat peidetud ühikute arvu [22]. Lihtsa probleemi korral liiga paljude ühikute kasutamine põhjustab ülesobituse (*overfitting*) [22], mis tähendab seda, et algoritm jätab treeningandmed meelde ning seetõttu ei oska hiljem, näiteks valideerimisel, õpitud tunnuste põhjal üldistada [3].

Tehisnärvivõrgu peidetud kihtide arvu valikul näeb märgatavaid erinevusi töös kahekihilise ning kolmekihilise närvivõrgu korral [22]. Rohkemate kihtide lisamine edendab ainult konvolutsiooniliste närvivõrkude üldistamisvõimekust [22]. Antud lõputöö raames tehisnärvivõrgu kihtide ega neuronite arvu ei muudetud.

3.3.1 Treeningplatvorm

Käesoleva lõputöö käigus treenitud mudeli treenimisprotsess plaaniti läbi viia kasutades pilveteenuseid, mis võimaldavad kasutada *GPU*'sid ehk graafikaprotsessoreid. *GPU*'de kasutamine treenimisel *CPU*'de e. tavapäraste protsessorite asemel kiirendab oluliselt treenimise kulgu [24].

Mudeli treenimine pidi aset leidma *Google Cloud Platvormi*'s ehk *GCP*'s, mis võimaldab mugavalt ülesse seada virtuaalmasinat ning kasutada 300 *USD* väärtuses krediiti treenimiskulude katteks. Pärast korduvaid ebaõnnestunud katseid luua virtuaalmasin soovitud *GPU*'ga otsustati alustada treenimist mujal pilveteenuses. *GCP*'s polnud tol hetkel piisavalt vabu vahendeid soovitud virtuaalmasina loomiseks.

Närvivõrgu treenimist alustati pilveteenuses *Vast.ai*, kuid renditud virtuaalmasina serveri asukoha tõttu oli üles- ja allalaadimiskiirused liiga madalad ning seetõttu otsustati treenimine migreerida mujale. Pärast viit epohhi lõpetati *Vast.ai*'s treenimine ning prooviti uuesti *GCP*'s luua virtuaalmasin, mis osutus sel korral edukaks. Ülejäänud treenimine viidi läbi *Google Cloud Platform*'is.

Treenimist jätkati esialgu planeeritud *Google Cloud Platform*'ist renditud virtuaalmasinal, mis võimaldas kasutada *GPU*'d. *GPU*'ks valiti üks *NVIDIA Tesla T4*

graafikaprotsessor. Treenimiseks kasutati tehisnärvivõrgu õppimisraamistikku *Pytorch* ja paralleelse andmetöötluse platvormi *CUDA 10.1*.

3.3.2 Treenimisprotsess

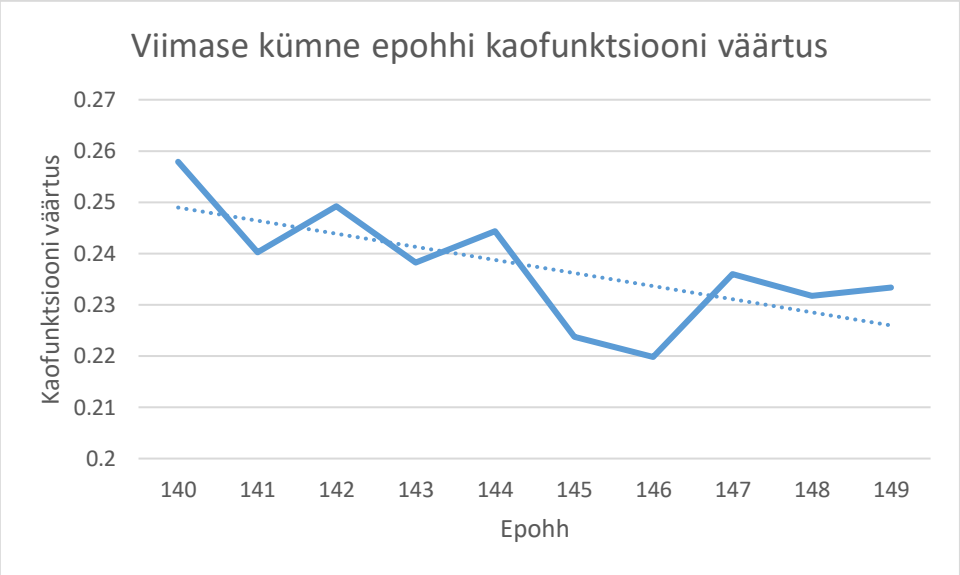
Treenimine viidi läbi *Google Cloud Platvorm*'i keskkonnas, kus treenimise kulgu monitooriti läbi *Visdom* visualiseerimise tööriista. *Visdom*'is kajastus mudeli kaofunktsiooni graafik, mis andis aimu, kui kiiresti kohaneb mudel treeningandmetega.

Kuna ühe epohhi läbimine oli väga ajakulukas just ühe treeningandme ehk stseeni mahu tõttu, prooviti leida mooduseid, kuidas vähendada ühe epohhi treenimiseks kuluvat aega. Selleks muudeti andmetelaadimise osa selliselt, et stseeni töötlemisel laetakse järgmise stseeni andmed paralleelselt. Sellisel moel muutus algoritmi töö kiiremaks, kuna andmete laadimisele aega ei kulunud. Esialgne andmete laadimise korral kulus ühe epohhi läbimiseks keskmiselt 4 tundi ja 22 minutit, kuid uuendatud moodus vähendas keskmist ajakulu poole tunni võrra ehk epohh läbiti 3 tunni ja 52 minutiga. Esialgse laadimisega läbiti keskmiselt 5.5 epohhi päevas, mis tähendab, et 150 epohhi läbimiseks kuluv aeg küündib 27 päeva ning 6 tunnini. Uuendatud versioon läbis keskmiselt 6.2 epohhi päevas ning ennustatavalt läbitakse 150 epohhi 24 päeva ja 4 tundi möödudes. Uuendus tõi ajalist võitu üle 3 päeva, mis on ka rahaline võit, kuna ühe päeva treenimise maksumus *GCP* pilveteenuses kujunes keskmiselt 12-13 €.

Joonis 8: Kaofunktsiooni väärtuse kahanemine treeningprotsessi vältel.

Eeloleval graafikul (Joonis 8) on toodud välja mudeli kogu treenimise kaofunktsiooni graafik. Graafikult on näha, et 150 epohhi läbiti selges langustrendis. Treenitud mudeli

kaofunktsioon jõudis 150 epohhiga kahaneda tulemuseni ~0.2334. Samalajal jõudis juhendaja poolt treenitud mudeli kaofunktsioon kahaneda väärtuseni ~0.05, mis on väga palju väiksem antud töös treenitud mudeli kaofunktsiooni väärtuse suhtes.



Joonis 9: Viimase kümne epohhi kaofunktsiooni graafik

Kuna treenitud mudeli kaofunktsiooni väärtus viimases kümnes epohhis oli ikka langev (Joonis 9), on selge, et mudeli kaofunktsioon pole minimeeritud ning üldistusvõime parandamine on võimalik edasi treenides ehk lisanduvate epohhide läbimisel. Joonis 9 on kuvatud punktiirjoonega ka trendijoon, mis on selges languses. Antud lõputöö raames treenimine peatati, kuna eesmärgiks on võrrelda samade parameetritega treenitud mudeleid ning 150 epohhi on piisav, et tuua välja mudelite erinevused üldistusvõimes.

3.4 Treenitud mudeli valideerimine

Masinõppe mudelite ainsaks eesmärgiks on üldistada andmete põhjal ning teha seda hästi [25]. Üldistamine on mudeli võime teha arukat otsust andmete põhjal, mida pole varem õpitud [25]. Tehisnärvivõrgu mudeli üldistusvõime hindamiseks on vajalik treenitud masinõppe mudeli valideerimine. Valideerimisel võeti arvesse ainult autode tuvastamise ja lokaliseerimise võimekust, kuna treenimisprotsessil keskenduti ainult autode tuvastamisele.

Valideerimine viiakse läbi valideerimisandmestikul, mis kujutab endast andmeid, mida tehisnärvivõrk pole varem treenimisel töödelnud. Mudeli võimekuse hindamisel on samuti oluline valideerida treeningandmestikul, kuna sellisel juhul on võimalik avastada mudeli treenimisel tekkinud probleeme, näiteks ülesobitumine (*overfitting*) või alasobitumine (*underfitting*) [3].

Kui mudel suudab teha täpseid ennustusi treeningandmestikul, kuid valideerimisandmestikul on ennustuse täpsus väga palju halvem, on tegemist ülesobitumisega [3]. Ülesobitumise korral treenitakse tehisnärvivõrk sobituma treeningandmetega liiga hästi ning valideerimisel ei suuda närvivõrk oluliste tunnuste põhjal üldistada. See tähendab, et närvivõrk õpib kõik treeningandmed selgeks ja valideerimisandmestiku korral teeb ettearvamatuid ennustusi. Üheks ülesobituse tunnuseks on see, kui kaofunktsioon on stabiilselt madal, kuid valideerimisel pole tehisnärvivõrk edukas. Ülesobitumise vältimiseks on erinevaid meetodeid: treeningandmete suurendamine, regulariseerimine, varane lõpetamine (*early stopping*), väljajätu meetod (*drop out*), andmete rikastamine (*data augmentation*) ja mudelistruktuuri lihtsustamine [3] [25] [22].

Alasobitumise korral ei suuda mudel treeningandmetest ära õppida olulisi tunnuseid ning samuti ei oska uute andmete põhjal üldistada ja teeb ettearvamatuid ennustusi [25]. Alasobitumise parandamiseks võib suurendada mudeli keerukust, näiteks peidetud kihtide arvu (*hidden layers*) ja peidetud ühikute (*hidden units*) arvu [26]. Alasobitunud mudeli üldistusvõimekuse suurendamiseks võib samuti mudelit edasi treenida, kuni kaofunktsiooni väärtus on minimeeritud [3].

Töö käigus treenitud mudeli valideerimiseks kasutatud valideerimisandmestik koosneb 3472 stseenist. Iga valideerimiseks kasutatud stseen koosneb kahest järjestikulisest punktipilvest ajahetkedel *t-1* ja *t*, kaamerapildist ajahetkel *t*, kuhu joonistuvad ennustuste põhjal piirikastid, ja märgistest, mille kaudu arvutatakse statistikat üldistusvõime hindamiseks. Treenimis- ja valideerimisandmestiku jaotus järgib Xiaozhi Chen *et al* poolt väljapakutud andmejaotust [27], et tulemused oleksid võrreldavad. Ennustuse tegemisel luuakse uued pildifailid vastavalt 3D piirikastidele ning 2D piirikastidele, kuhu joonistatakse piirikastid mudeli töö visuaalseks hindamiseks. Asukohta ennustavate tehisnärvivõrkude mudelite valideerimisel mõõdetakse närvivõrgu poolt ennustatud piirkaste (*bounding box*) *ground truth* piirikastide kattuvust ehk IOU'd. Valideerimisel

kasutatud IOU läviks (*threshold*) valiti 0.1. IOU lävis määrab selle, kui palju peab ennustatud piirikast kattuma *ground truth* piirikastiga, et ennustus oleks tõene positiivne. IOU lävis valiti vastavalt lõputöö juhendaja poolt treenitud ja valideeritud mudeli põhjal, kuna sellisel viisil on võimalik võrrelda kahte treenitud mudelit.

3.5 Mudeli hindamise meetrika

Tehisnärvivõrgu mudeli hindamiseks kasutatakse erinevaid viise ja meetrikaid. Antud lõputöö raames loodud mudeli hindamiseks kasutatakse keskmise täpsuse (*average precision*) ehk AP meetrikat, mida kasutatakse objektituvastavate mudelite jõudluse hindamiseks.

Antud lõputöös treenitud mudeli AP väärtused leiti kasutades *eval_kitti*[1] tarkvara, mis kasutas AP meetrika arvutamiseks 11-punktilist interpoleerimist. 11-punktiline interpoleerimine keskmistab täpsust üheteistkümne võrdselt jaotatud saagise väärtustel. AP annab väga hea ülevaate, kui hästi tehisnärvivõrgu mudel toimib võrreldes teiste mudelitega, mis on treenitud sama andmestikuga. Valideerimisel leitud AP väärtused arvutatakse *eval_kitti* algoritmis masinõppe mudeli ennustuste täpsustega (*precision*) ja saagistega (*recall*) valemite põhjal [28]:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{interp(r)} \quad (1)$$

ning

$$\rho_{interp} = \max_{\tilde{r}:\tilde{r} \geq r} \rho(\tilde{r}), \quad (2)$$

kus $\rho(\tilde{r})$ on mõõdetud täpsus saagise $\tilde{r}$ korral.

Täpsus annab infot, kui palju positiivsetest ennustustest on korrektselt ennustatud ning selle arvutamiseks kasutatakse valemit [29]:

$$täpsus = \frac{tõesed\ positiivsed}{tõesed\ positiivsed + väärad\ positiivsed} \quad (3)$$

[1] https://github.com/cguindel/eval_kitti

Saagis on meetrika, mis annab aimu, kui palju tegelikest positiivsetest ennustati õigesti. Saagist arvutatakse järgneva valemiga [29]:

$$saagis = \frac{tõesed\ positiivsed}{tõesed\ positiivsed + väärad\ negatiivsed} \quad (4)$$

Tehisnärvivõrgu mudeli üldistusvõime hindamiseks on oluline analüüsida mõlemaid näitajaid: nii täpsust kui ka saagist. Tavapäraselt täpsuse parandamisel halveneb saagis ja vastupidi [29].

3.6 Mudelite võrdlus

Antud lõputöö eesmärk oli võrrelda kahte erinevat VoxelNeti arhitektuuriga mudelit. Üheks neist oli korraga üht punktipilve töötlev mudel ning teiseks on antud töö praktilises osas loodud mudel, mis töötleb kaht järjestikulist punktipilve samaaegselt. Võrreldavad mudelid on arhitektuuri dimensoonidelt üpris erinevad ning seetõttu on oodata ka erinevat võimekust autode tuvastamises ja lokaliseerimises.

Võrreldavatest mudelitest kumbki pole treenitud kaofunktsiooni minimeerimiseni ehk parima võimaliku tulemuseni just mudelite treenimise aja ning kulukuse tõttu. Treenimiseks määratud 150 epohhi on piisav treenimise pikkus, et mudelitevahelisest erinevustest järeldusi teha.

Mudeleid valideeriti kolmel erineval viisil, milleks oli 3D, 2D ning linnulennuline valideerimine ehk BEV (*bird's eye view*). 3D-valideerimisel hinnatakse mudeli kahemõõtmelist ennustust ehk piirkasti kahemõõtmelises ruumis; ning ka kolmemõõtmelist ennustust ehk kolmemõõtmelist piirkasti [20]. 2D-valideerimisel hinnatakse vaid mudeli kahemõõtmelist ennustust [20]. Linnulennulises valideerimisel hinnatakse mudeli nii kahemõõtmelist ennustust, kui ka kolmemõõtmelist ennustust, kuid ennustus ei pea sisaldama informatsiooni objekti asukoha kohta kõrgusteljel [20]. Igal viisil oli kolm raskusastet: lihtne, keskmine ja raske.

Raskustasemete kategooriate kriteeriumid on vaadeldavad järgnevast tabelist (Tabel 1):

Tabel 1: Valideerimise raskusastmed [20].

	Minimaalne piirikasti kõrgus	**Maksimaalne nähtavus**	**Maksimaalne kärbitus**
Lihtne	40 px	Täielikult nähtav	15%
Keskmine	25 px	Osaliselt mittenähtav	30%
Raske	25 px	Raskesti nähtav	50%

Minimaalne piirkasti kõrgus tähendab, et mudeli poolt ennustatud piirkast peab olema vähemalt etteantud kõrgusega. Maksimaalne nähtavus määrab, kui hästi on ennustatav objekt punktipilves ja pildil visuaalselt nähtav. Maksimaalne kärbitus määrab, kui suur osa ennustatavast objektist võib olla kaadrist väljaspool, et seda võetaks arvesse valideerimisel [20]. Näiteks lihtsas valideerimiskategooris võib olla tuvastatav objekt maksimaalselt 15% kaadrist väljas, et objekti võetaks arvesse mudeli valideerimisel täpsuse ja saagise arvutamisel.

Järgnevates tabelites on välja toodud valideerimiste tulemused treeningandmestiku (Tabel 2) ja valideerimisandmestiku (Tabel 3) põhjal. Võrreldava mudeli all on mõeldud lõputöö juhendaja Martin Rebase poolt treenitud mudeli valideerimisandmed ning treenitud mudeli all antud lõputöö raames treenitud mudeli valideerimisandmed.

Tabel 2: Valideerimistulemused treeningandmestikul. AP meetrikate võrdlemine.

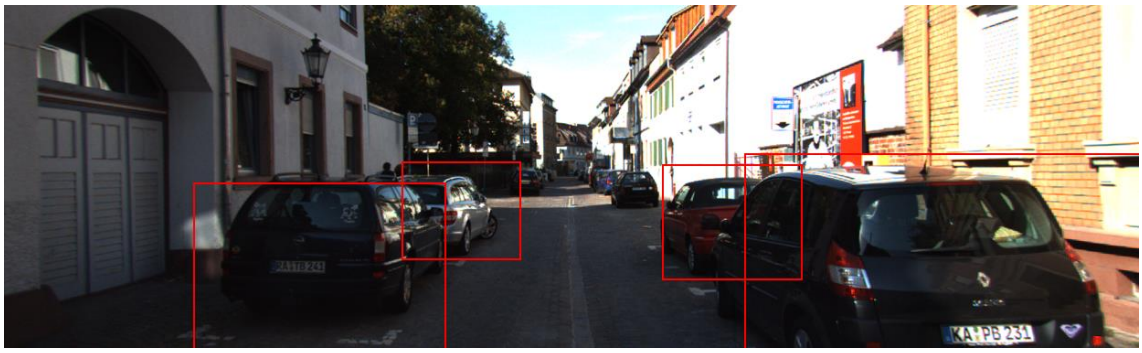
	Võrreldav mudel Lihtne/keskmine/raske	**Treenitud mudel** Lihtne/keskmine/raske
3D AP	62.48% / 57.74% / 54.69%	47.20% / 25.84% / 21.14%
2D AP	51.71% / 45.83% / 46.06%	48.17%/ 26.49% / 21.70%
BEV AP	62.53% / 57.84% / 54.79%	47.37% / 25.93% / 21.21%

Tabel 3: Valideerimistulemused valideerimisandmestikul. AP meetrikate võrdlemine.

	Võrreldav mudel Lihtne/keskmine/raske	**Treenitud mudel** Lihtne/keskmine/raske
3D AP	57.93% / 58.02% / 52.53%	32.76% / 19.91% / 15.77%
2D AP	39.54% / 41.14% / 41.80%	36.28% / 20.87% / 18.64%
BEV AP	58.07% / 58.29% / 52.81%	32.98% / 20.02% / 15.84%

Tulemustest on näha, et lõputöö raames treenitud mudeli üldistusvõime ei küündi juhendaja poolt treenitud mudeli tulemusteni. Juhendaja poolt treenitud mudeli kaofunktsioon oli ka 150 epohhi lõpuks tunduvalt madalam. Treenitud mudeli kaofunktsioon 150 epohhi läbimisel oli ~0.2334 ning juhendaja poolt treenitud mudeli oma ~0.05. Treenimisandmestiku põhjal teostatud valideerimine osutus mõlema mudeli puhul edukamaks kui valideerimisandmestiku põhjal. Selline nähtus on mudelite treenimisel tavapärane. Kõige sarnasema tulemuse võrreldava mudeliga saavutas treenitud mudel valideerimisandmestiku põhjal 2D-valideerimise lihtsal raskusastmel, kus tulemusteks on 39.54% võrreldaval mudelil ning 36.28% treenitud mudelil.

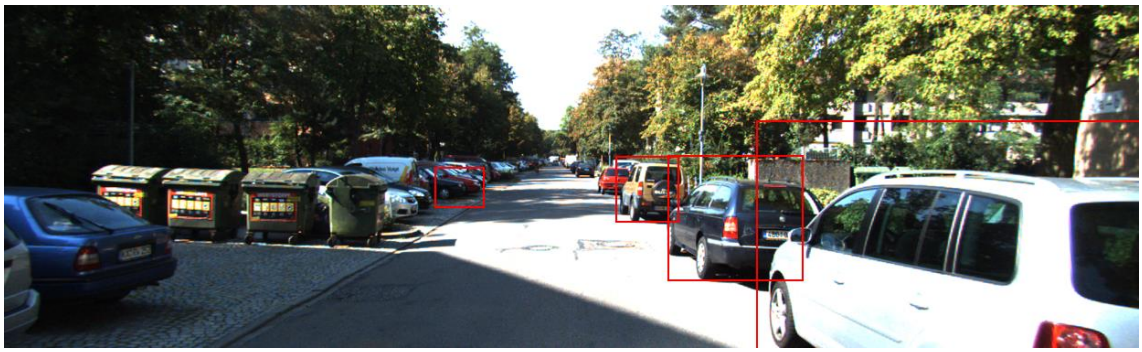
Mudeli valideerimisel genereeritud piirkastide piltide näitel saab tuua välja treenitud mudeli üldistusvõime mustreid. Selgus, et treenitud mudel suudab tuvastada ja lokaliseerida pigem autosid, mis on sõidetava teega piki (Joonis 10). Samuti on toodud välja juhendaja poolt treenitud mudeli ennustus sama stseeni kohta (Joonis 11), kust on näha, et mudel on teinud palju rohkem ennustusi, kuid paljud tehtud ennustused on ebatäpsed. Vastanduvalt on treenitud mudeli ennustusi pigem vähem, kuid on täpsemad (Joonis 10), mis tähendab vähem väärate positiivsete arvu.



Joonis 10: Treenitud mudeli tehtud ennustused. Stseen 000311 valideerimisandmestikust.

Joonis 11: Juhendaja poolt treenitud mudeli tehtud ennustused. Stseen 000311 valideerimisandmestikust.

Sõiduteega ristiolevaid parkivaid autosid tuvastas treenitud mudel vähem (Joonis 12). Joonis 12 on võimalik näha, et treenitud mudel pole suutnud tuvastada roheliste prügikastide ees olevaid autosid ning prügikastide taga olevatest autodest tuvastati vaid üks. Vastandlikult samal joonisel (Joonis 12) paremal pool teeserva pikuti pargitud autod tuvastati edukalt.



Joonis 12: Treenitud mudeli ennustused. Stseen 000692 valideerimisandmestikust.

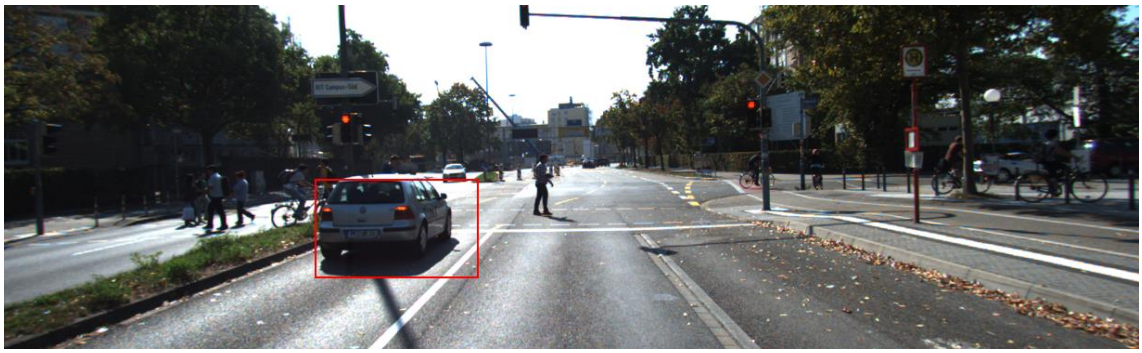
Samuti leidus seaduspärasus, kus mudel suutis selgelt eristada autosid inimeste seast (Joonis 13, Joonis 14). Joonis 14 olev piirkast ei ümbritse tuvastatud autot, kuid valideerimisel loetakse selline tuvastus õigeks. Selleks, et mudel suudaks täpsemini piirikaste määrata, oleks tarvis mudelit täiendavalt treenida.

Joonis 13: Treenitud mudeli poolt auto tuvastamine inimeste seast. Stseen valideerimisandmestikust 002892.

Joonis 14: Treenitud mudeli poolt auto tuvastamine inimeste seast. Stseen valideerimisandmestikust 003487.

Mudel tegi rohkem täpseid ennustusi liikuvate autode tuvastamisel, eelkõige just samal sõidurajal või kõrvaloleval rajal samas suunas sõitvate autode puhul (Joonis 16), kui ka lähedalasuvaid vastutulevaid sõidukeid (Joonis 17, Joonis 15).

Joonis 15: Treenitud mudeli poolt õigesti tuvastatud vastutulev auto. Stseen valideerimisandmestikust 002585.



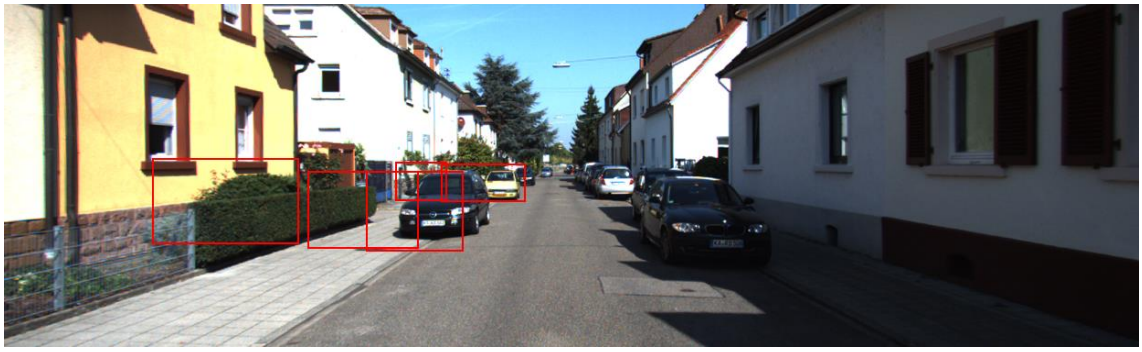
Joonis 16: Treenitud mudeli poolt kõrvaloleval rajal oleva auto tuvastus. Stseen valideerimisandmestikust 000134.

Juhendaja poolt treenitud mudeliga võrreldes oli vastutulevate autode lokaliseerimine võrdselt hea (Joonis 17, Joonis 18).

Joonis 17: Treenitud mudeli poolt vastutuleva auto tuvastamine. Stseen valideerimisandmestikust 002251.

Joonis 18: Juhendaja poolt treenitud mudeli ennustus vastutulevast autost. Stseen valideerimisandmestikust 002251.

Treenitud mudeli ennustusi vaadeldes tehti pigem vähem ennustusi, kuid kohati täpsemaid võrreldes juhendaja mudeli ennustustega. Samuti oli tulemusi, kus ennustus oli tehtud valesti (Joonis 19, Joonis 20). Ilmnes muster, kus treenitud mudel tuvastas autosid kohtades, kus tegelikult oli aed või majasein. Tihti oli piirikaste, kus ristuva tee äärne aed tuvastati autona (Joonis 20). Lõputöö raames treenitud ja juhendaja poolt treenitud mudeli sarnasuseks võib tuua, et mõlemad lokaliseerisid autosid kohtades, kus oli majasein või teepiire (Joonis 20, Joonis 21).



Joonis 19: Treenitud mudeli poolt heki tuvastamine autona. Stseen valideerimisandmestikust 000885.

Joonis 20: Treenitud mudeli poolt aia tuvastamine autona. Stseen valideerimisandmestikust 000969.

Joonis 21: Juhendaja poolt treenitud mudeli märgitud piirkastid. Stseen valideerimisandmestikust 000969.

3.7 Järeldused ja võimalikud edasiarendused

Töö praktilises osas treenitud mudeli valideerimistulemustest kajastub, et närvivõrgu mudel pole andmetega piisavalt hästi kohanenud, et teha sama edukaid ennustusi autode tuvastamiseks ja asukoha määramiseks, kui seda teeb juhendaja poolt treenitud mudel.

Kuna viimaste treeningepohhide kaofunktsioonid olid langustrendis, on võimalik, et närvivõrku edasi treenides oleks võimalik saada ka paremaid tulemusi autode tuvastamisel ja lokaliseerimisel. Samuti võib parandada algoritmi võimekust, muutes mudeli treenimisel kasutatavaid hüperparameetreid. Sealhulgas võib positiivselt mõjuda kihtide lisamine, mis muudab tehisnärvivõrgu sügavamaks, ning mini-ploki arvu suurendamine. Mini-ploki arvu suurendamine polnud antud lõputöö raames võimalik, sest see nõuaks treenimisel liialt suurt arvutusvõimsust.

Kuna muudetud tehisnärvivõrgu struktuur on sedavõrd mahukam, on võimalik, et närvivõrgu iga neuroni osakaal ennustustes on seda väiksem ning seetõttu kulgeb treenimine aeglasemalt. Seetõttu on vajalik treenida rohkem epohhe tuvastamise ja lokaliseerimise parandamiseks. Selleks võib kasutada väljajätu meetodit, mis võimaldab tehisnärvivõrgu treenimisel osade neuronite väljundi jätta arvestamata.

Lõputöö tulemustest järeldus, et VoxelNeti suurendatud arhitektuuriga tehisnärvivõrgu versioon on võimeline tuvastama ja lokaliseerima autosid järjestikulistest punktipilvedest. VoxelNeti arhitektuuri dimensioonide suurendamine nõuab suuremat epohhide arvu ning treeningparameetrite muutmist. Suure arhitektuuriga närvivõrk ei kohane andmetega sama kiiresti kui teeb seda väiksem närvivõrk, seetõttu on oluline

muuta arvestatavate tulemuste saavutamiseks närvivõrgu mudeli struktuuri, parameetreid või treenimise ülesehitust.

Võimalikeks edasiarendusteks on mudeli edasi treenimine katsetades erinevaid hüperparameetrite kombinatsioone. Samuti oleks võimalik implementeerida osa, kus kahe järjestikulise punktipilve töötlusel ennustatakse ka auto liikumise suund ja kiirus.

4 Kokkuvõte

Lõputöös uuriti erinevusi üht punktipilve töötleva ja kahte järjestikulist punktipilve töötleva masinõppemudeli vahel ning nende ennustustäpsust autode tuvastamisel ja lokaliseerimisel. Samuti vaadeldi, kuidas mõjub struktuuri suurendamine närvivõrgu ennustustulemustele.

Närvivõrgu mudeli loomiseks võeti aluseks antud lõputöö juhendaja Martin Rebase implementatsioon VoxelNetist, mida kohandati töötlema kahte punktipilve samaaegselt. Samuti tegeldi treening- ja valideerimisandmestiku kohandamisega loodud VoxelNeti struktuuriga, kuna andmestikest puudusid stseeni töötluseks vajalikud kaadrid. Lõputöö raames viidi läbi mudeli treenimine *Google Cloud Platform* pilveteenuses. Treenimiseks kasutati 3713 ststeeni. 150 epohhi läbinud mudel valideeriti ning hinnati saadud tulemusi. Valideerimine viidi läbi 3472 stseeni põhjal.

Valideerimistulemusi võrreldi juhendaja poolt treenitud mudeli valideerimistulemustega, millest võrreldi AP meetrikat ehk keskmist täpsust. Kõige parema tulemuse valideerimisandmestikul valideerides saavutas lõputöö raames treenitud mudel kahemõõtmelise valideerimise lihtsas versioonis, kus AP tõenäosuseks kujunes 36.28%, mis on juhendaja poolt treenitud mudeli valideerimisel saadud skooriga 39.54% kõige sarnasem. Valideerimisandmete põhjal toodi välja sarnasusi kui ka erinevusi mudelite ennustusmustrites valideerimistel loodud piirikastide piltide näitel. Seaduspärasustest selgus, et juhendaja poolt treenitud mudel teeb rohkem ennustusi, millest osa on ebatäpsed, kuid samade stseenides teeb treenitud mudel vähem ennustusi, mis on täpsemad.

Valideerimisest selgus, et loodud tehisnärvivõrk on võimeline kahelt järjestikuliselt punktipilvelt autosid tuvastama ja lokaliseerima. Praktilise töö tulemusena selgus, et mudeli kaofunktsioon koondub soovitud suunas, mis tähendab, et mudelit kauem treenides on võimalik jõuda praktiliselt rakendavate valideerimistulemusteni. Käesoleva töö eelarve piires polnud võimalik treenida närvivõrgu mudelit miinimumini, kuna

pilveteenuse kasutamine on küllaltki kulukas nii ajaliselt kui ka rahaliselt. See-eest treenitud 150 epohhi on piisav treenimise kestus, kuna mudelitevahelised erinevused kujunesid välja. Antud lõputöö raames treenitud 150 epohhi polnud piisav, et ületada üht punktipilve töötleva mudeli üldistusvõimet.

Kasutatud kirjandus

[1] S. Gibbs, „Self-driving cars: who's building them and how do they work?,“ The Guardian, 26 May 2016. [Võrgumaterjal]. Available: https://www.theguardian.com/technology/2016/may/26/self-driving-cars-whos-building-them-and-how-do-they-work. [Kasutatud 10 4 2020].

[2] T. English, „How do self-driving cars work?,“ Interesting Engineering, [Võrgumaterjal]. Available: https://interestingengineering.com/how-do-self-driving-cars-work. [Kasutatud 10 4 2020].

[3] A. Ng, K. Katanforoosh ja Y. B. Mourri, „Neural Networks and Deep Learning - deeplearning.ai,“ Coursera, [Võrgumaterjal]. Available: https://www.coursera.org/specializations/deep-learning.

[4] Leddar Tech, „Why lidar?,“ Leddar Tech, [Võrgumaterjal]. Available: https://leddartech.com/why-lidar/. [Kasutatud 10 4 2020].

[5] Y. Zhou ja O. Tuzel, „VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection,“ %1 *2018 IEEE/CVF Conference on Computer Vision and Pattern*, Salt Lake City, UT, USA, 2018.

[6] SAS Institute Inc., „Machine Learning,“ [Võrgumaterjal]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html. [Kasutatud 10 4 2020].

[7] D. K. Judith Hurwitz, Machine Learning For Dummies, Hoboken: John Wiley & Sons, Inc., 2018.

[8] H. Wang ja B. Raj, „A Survey: Time Travel in Deep Learning Space: An Introduction to Deep Learning Models and How Deep Learning Models Evolved from Initial Ideas,“ 2015.

[9] C. Taylor, „Structured vs Unstructured data,“ Datamation, 28 3 2018. [Võrgumaterjal]. Available: https://www.datamation.com/big-data/structured-vs-unstructured-data.html. [Kasutatud 10 4 2020].

[10] E. Inzaugarat, „Understanding Neural Networks: What, How and Why?,“ Towards Data Science, 31 10 2018. [Võrgumaterjal]. Available: https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31. [Kasutatud 10 04 2020].

[11] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way,“ Towards Data Science, 15 12 2018. [Võrgumaterjal]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Kasutatud 14 4 2020].

[12] S. Ren, K. He, R. Girshick ja J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, kd. 39, nr 6, pp. 1137-1149, 2017.

[13] Techopedia, „Volume Pixel (Volume Pixel or Voxel),“ Techopedia, 20 1 2013. [Võrgumaterjal]. Available: https://www.techopedia.com/definition/2055/volume-pixel-volume-pixel-or-voxel. [Kasutatud 10 04 2020].

[14] J. S. Aber, I. Marzolff ja J. Ries, „Photogrammetry,“ %1 *Small-Format Aerial Photography*, Elseiver Science, 2010, pp. 23-39.

[15] M. Rouse, „Point Cloud,“ What Is, [Võrgumaterjal]. Available: https://whatis.techtarget.com/definition/point-cloud. [Kasutatud 14 4 2020].

[16] F. Rooms, „Point clouds - 1: What's the point?,“ Brycsis Blog, 2019. [Võrgumaterjal]. Available: https://blog.bricsys.com/point-clouds-whats-the-point/. [Kasutatud 10 4 2020].

[17] X. Liu, Z. Han, Y.-S. Liu ja M. Zwicker, „Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-Based Sequence to Sequence Network,“ %1 *The Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[18] C. Thomson, „What are the most common 3D point cloud formats and how to solve interoperability issues,“ Vercator, [Võrgumaterjal]. Available: https://info.vercator.com/blog/what-are-the-most-common-3d-point-cloud-file-formats-and-how-to-solve-interoperability-issues. [Kasutatud 9 4 2020].

[19] A. Geiger, R. Urtasun ja C. Stiller, „Vision meets robotics: the KITTI dataset,“ *The International Journal of Robotics Research*, kd. 32, pp. 1231-1237, 2013.

[20] A. Geiger, P. Lenz ja R. Urtasun, „Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,“ %1 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[21] D. N. Krawetz, „The Hacker Factor Blog,“ 21 01 2013. [Võrgumaterjal]. Available: http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html. [Kasutatud 2020 4 23].

[22] J. Leonel, „Hyperparameters in Machine/ Deep Learning,“ Medium, 6 4 2019. [Võrgumaterjal]. Available: https://medium.com/@jorgesleonel/hyperparameters-in-machine-deep-learning-ca69ad10b981. [Kasutatud 13 4 2020].

[23] V. Bushaev, „Adam — latest trends in deep learning optimization.,“ Towards Data Science, 22 10 2018. [Võrgumaterjal]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c. [Kasutatud 20 5 2020].

[24] J. Dsouza, „What is a GPU and do you need one in Deep Learning?,“ Towards Data Science, 25 4 2020. [Võrgumaterjal]. Available: https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d. [Kasutatud 17 5 2020].

[25] A. Al-Masri, „What Are Overfitting and Underfitting in Machine Learning?,“ Towards Data Science, 22 6 2019. [Võrgumaterjal]. Available: https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690. [Kasutatud 15 4 2020].

[26] A. Sharma, „Tackling Underfitting and Overfitting Problems in Data Science,“ Analytich India Mag, 27 22 2018. [Võrgumaterjal]. Available: https://analyticsindiamag.com/tackling-underfitting-and-overfitting-problems-in-data-science/. [Kasutatud 14 4 2020].

[27] X. Chen, H. Ma, J. Wan, B. Li ja T. Xia, „Multi-View 3D Object Detection Network for Autonomous Driving,“ *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, pp. 1907-1915, 22 6 2017.

[28] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn ja A. Zisserman, „The Pascal Visual Object Classes (VOC) Challenge,“ *International Journal of Computer Vision,* 2009.

[29] Google Inc, „Classification: Precision and Recall,“ Google Inc, [Võrgumaterjal]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. [Kasutatud 24 5 2020].

[30] C.-F. Wang, „The Vanishing Gradient Problem,“ TowardsDataScience, 8 1 2019. [Võrgumaterjal]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. [Kasutatud 20 7 2020].

[31] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[32] S. Patrikar, „Towards Data Science,“ Batch, Mini Batch & Stochastic Gradient Descent, 2019 10 2019. [Võrgumaterjal]. Available: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a. [Kasutatud 10 04 2020].

[33] D. Kapil, „Stochastic vs Batch Gradient Descent,“ Medium, 7 1 2019. [Võrgumaterjal]. Available: https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1. [Kasutatud 10 4 2020].

[34] Missinglink.ai, „7 Types of Neural Network Activation Functions,“ Missinglink.ai, [Võrgumaterjal]. Available: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. [Kasutatud 10 4 20].

[35] Queriozf.com, „Evaluation Metrics for Ranking problems: Introduction and Examples,“ Queriozf.com, 13 4 2020. [Võrgumaterjal]. Available: https://queirozf.com/entries/evaluation-metrics-for-ranking-problems-introduction-and-examples. [Kasutatud 30 7 2020].