TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Terje Russka 164016IAPB

PARKIMISKOHTADE HÕIVATUSE TUVASTAMINE LÄBI KAAMERAPILDI

Bakalaureusetöö

Juhendaja: Martin Rebane
MSc

Tallinn 2019

Autorideklaratsioon

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

Autor: Terje Russka

19.04.2019

Annotatsioon

Käesoleva töö eesmärgiks on luua Tallinna Tehnikaülikooli avatud parklatele parkimiskohtade hõivatuse tuvastamissüsteem. Lahendus koosneb reaalajas kaamerast saadud kaadris olevate parkimiskohtade analüüsist ning hõivatuse tõenäosuse leidmisest. Edasiarendusel on võimalik süsteemis kasutatavat eeltreenitud tuvastusmudelit välja vahetada laiemalt isetreenitud mudeliga, mis oleks spetsiaalselt treenitud Tallinna Tehnikaülikooli avatud parklate parkimiskohtade peal, et anda täpsemaid tulemusi. Süsteemi on võimalik integreerida rakendusliidesega, mis suudaks edastada parkimiskohtade hõivatuse staatust näiteks Tallinna Tehnikaülikooli mobiilirakendusele.

Töö käigus katsetati erinevate olemasolevate tehisnärvivõrkude sobivust ja täpsust ning täpseimat tehisnärvivõrku kasutati spetsiaalse treeningandmestiku kogumiseks Tallinna Tehnikaülikooli väliparkla parkimiskohtadest. Kogutud andmesikuga treeniti uus tehisnärvivõrk, eesmärgiga suurendada parkimiskohtade seisundite tuvastuse täpsust. Tulemuseks tõusis parkimiskohtade seisundite tuvastuse täpsus isetreenitud tehisnärvivõrku kastutades 10 protsenti.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 5 peatükki, 27 joonist, 1 tabelit.

Abstract

Parking Space Occupancy Detection Through A Camera Picture

The purpose of this thesis is to create a system that can detect parking space occupancy from real time camera footage, to provide information about free parking spaces to drivers seeking parking.

The open space parking lots at Tallinn University of Technology lack physical parking detectors, thus drivers spend extra time and resources on trying to find an available parking space.

To solves this problem, different types of ready-made neural networks were tested out, with the goal to detect parking space occupancy. The neural network with the highest accuracy was used to gather a dataset consisting of parking spaces in different states from the camera feed provided by Tallinn University of Technology.

The dataset was used to train a convolutional neural network and the final model achieved a 99% accuracy.

The thesis is in Estonian and contains 23 pages of text, 5 chapters, 27 figures, 1 table.

Lühendite ja mõistete sõnastik

Aktivatsioonifunktsioon	Kasutatakse saadud tehisnärvivõrgu väljundi muutmiseks valitud formaati.
CNN	*Convolutional Neural Network,* konvolutsiooniline tehisnärvivõrk, kasutatakse visuaalsete kujutiste analüüsimisel.
COCO	*Common Objects in Context,* suuremahuline objektide tuvastamise, segmenteerimise andmestik.
Epohh	*Epoch*, treeningandmestiku täieliku treeningtsükli läbimise arv.
Filter	Maatriks, mis iseloomustab pildi tunnusjooni.
Gradientlaskumine	*Gradient descent,* optimeerimisalgoritm parima kaalude grupi leidmiseks masinõppes, mis iseloomustaks sisendite ja tulemuste suhet.
LRN	*Local Response Normalization,* normaliseerimisprotsess, mille käigus summutatakse ühtlaselt suured väljundid, ning tõstetakse väljundite tundlikkust.
Maksimaalne ahenduskiht	*Max pooling*, protsess, mille käigus vähendatakse kujundise ruumilist suurust, säilitades tähtsamad omadused.
Plokk	*Batch,* andmete kogus, mida tehisnärvivõrgu treeningu ajal korraga kasutatakse.
ReLU	*Rectified Linear Unit,* funktsioon, mis muudab kõik negatiivsed väärtused nullideks.
RGB	*Red-green-blue,* punane-roheline-sinine
SoftMax	Aktivatsioonifunktsiooni liik, kus kõikide klassifikatsioonide tõenäosuste summa on 1.
Treeningandmed	Tehisnärvivõrgu treenimiseks kasutatav andmestik.
Tunnuskaart	Maatriks filtri ja sisendpildi korrutisest mis näitab filtrile vastava omaduse olemasolu sisendpildil.
Täielikult ühendatud kiht	*Fully connected layer,* muudab sisendmassiivi klassifikatsioonide arvu põhjal väljundiks.
Varjatud kiht	*Hidden layer,* tehisnärvivõrgu kiht, kus toimub informatsiooni töötlemine.
Võrgusõlm	*Node,* tehisnärvivõrgu väikseim element.

Sisukord

Jooniste loetelu

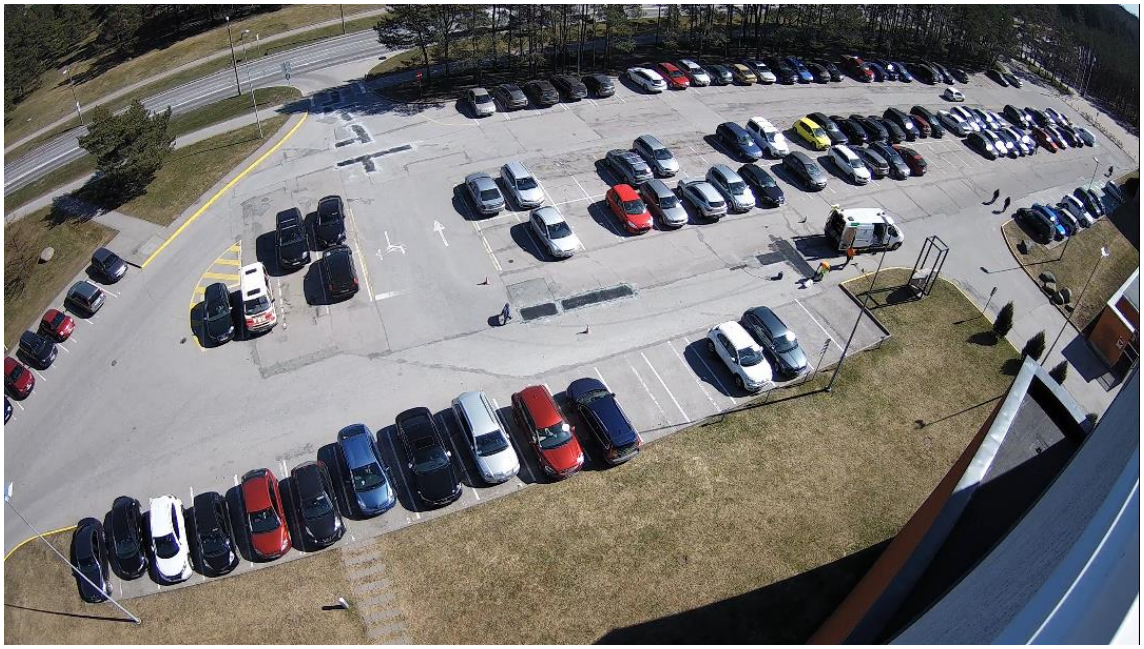
Tabelite loetelu

1 Sissejuhatus

Tallinna Tehnikaülikooli avalikud parklad on kõik väliparklad, kus puuduvad loendusandurid ning kuhu maapealsete sensorite paigaldamine ja hooldamine oleks liiga kulukas. Seetõttu pole autojuhile teada, kus asetsevad parklas olevad vabad parkimiskohad ning autojuht peab vaba parkimiskoha leidmiseks kulutama rohkem aega ja ressurssi.

Töö eesmärgiks on luua süsteem, mis võtab reaalajas kaameraga parkimisplatsi osast kaadreid ning prognoosib kaadris olevatele parkimiskohtadele vaba- ja hõivatud tõenäosuse.

Süsteemi loomiseks kasutatakse Tallinna Tehnikaülikooli poolt üles seadistatud kaamerat, mis asub energeetika hoone katusel ning millele on kooli sisevõrgus ligipääs reaalajas kaadritele resolutsiooniga 2688x1520 pikslit.



Joonis 1. Kaader Tallinna Tehnikaülikooli väliparkla kaamerast.

Süsteemi peab olema suuteline tuvastama erinevates ilmastikuoludes konfigureeritud parkimiskohtade seisundeid kõrge täpsusega, eriti tähtis on just hõivatud

parkimiskohtade seisundi tuvastus, kuna kasutaja seisukohtalt on valeinformatsioon vabast parkimiskohast suurema negatiivse mõjuga, kui valeinformatsioon hõivatud parkimiskohast.

Süsteemi loomiseks uuritakse olemasolevaid objektituvastuse tehisnärvivõrke ning analüüsitakse nende sobivust ja täpsust probleemi lahendamiseks. Parima sobivusega tehisnärvivõrku kasutatakse süsteemi algse prototüübina ning treeningandmestiku kogumiseks.

Saadud treeningandmestik kontrollitakse käsitsi üle, andes valesti tuvastatud parkimiskohtade seisunditele õiged väärtused. Eesmärgiks on treenida spetsiaalne tehisnärvivõrk, mis suudab olemasolevast tehisnärvivõrgust kõrgema täpsusega parkimiskohtade seisundeid tuvastada, võttes arvesse parkimisplatsi ja kaamera omadusi (kaamera kaugus ja nurk, puude varjud). Mõlemate tehisnärvivõrkude täpsuseid analüüsitakse uue kogutud andmestiku peal.

2 Analüüs ja testimine

Selles peatükis uuritakse võimalikke lahendusi ning nende lahenduste sobivust antud probleemile.

2.1 Parkimissensorid

Parkimisplatside haldamiseks on võimalik kasutusele võtta erinevaid füüsilisi parkimissensoreid, mida saab installeerida mistahes konfiguratsioonis. Parkimissensorid võivad olla paigaldatud näiteks maa-aluste-, maapealsete- või laesensoritena. Kõik sensorid on ühenduses parkimise juhtimissüsteemiga, ning suudavad reaalajas edastada vastava parkimiskoha seisundit, kasutades kahesuunalist raadiosidemehhanismi [1].

2.1.1 Maa-alused sensorid

Maa-alused sensorid paigaldatakse maa puurimise käigus, olles ära mõõdetud ja märgitud parkimiskoha keskpunkti [2]. Nende toiteallikad kestavad umbes 7 aastat ning töötavad stabiilselt karmides keskkondades ja temperatuuri kõikumisel [2].

Joonis 2. Maa-alune parkimissensor [1].

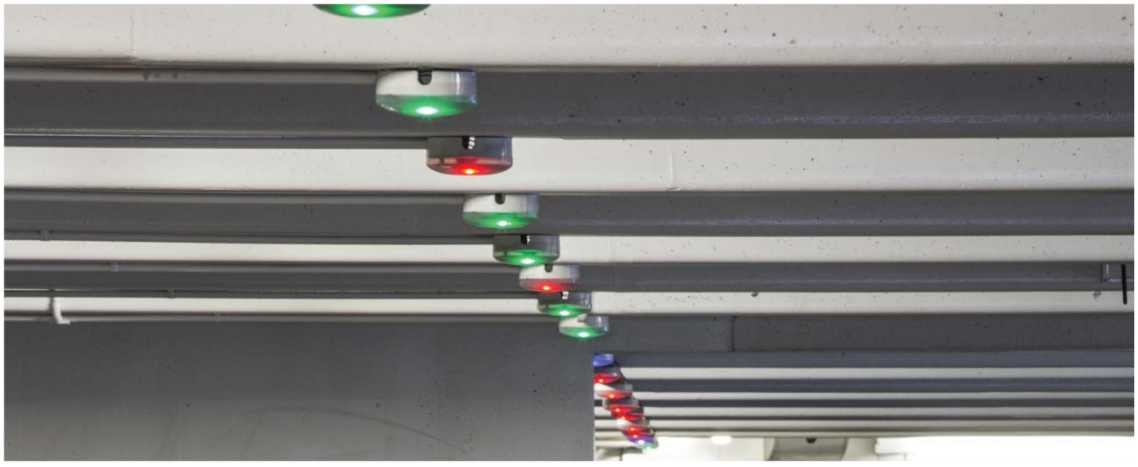
2.1.2 Maapealsed sensorid

Maapealseid sensoreid on võimalik paigaldada näiteks õhukestele pindadele, mis katavad kaableid või torustikku ning asukohtadele, kus maapinna puurimine ei pruugi olla võimaluseks [1].

Joonis 3. Maapealsed parkimissensorid [1].

2.1.3 Laesensorid

Hea nähtavuse ja värvikoodiga LED laesensoreid võimaldavad siseparklates autojuhtidel näha parkimisplatsil mitme rea parkimiskohtade hetkeseisu [1]. Iga laesensor, kombinatsioonis maalause sensoriga, jälgib eraldi vastavat parkimiskohta, tuvastades sõiduki saabumist, kohalolekut ja väljumist [3].



Joonis 4. Erinevate värvikoodidega LED laesensorid [1].

2.1.4 Paigaldamisprotsess

Parkimissensorite paigaldamine nõuab palju tööjõudu ja rahalist ressurssi, seega pole tegu kõige efektiivsema lahendusega. Maa-aluste parkimissensorite paigaldamine väliparklatele, kaasa arvatud Tallinna Tehnikaülikooli peaparklale, koosneks mitmest etapist.

Kõigepealt on vaja üles seada võrguühendus haldussüsteemi ja iga parkimissensori vahel, ning kontrollida kas raadiosagedussignaalid on piisavalt tugevad, et sensorid suudaksid parkimisinformatsiooni edastada [2]. Seejärel tuleb ära kaardistada parkimisplats ja määrata igale parkimiskohale vastav parkimissensor, mis installeeritakse teedeehituse spetsialistide poolt [2].

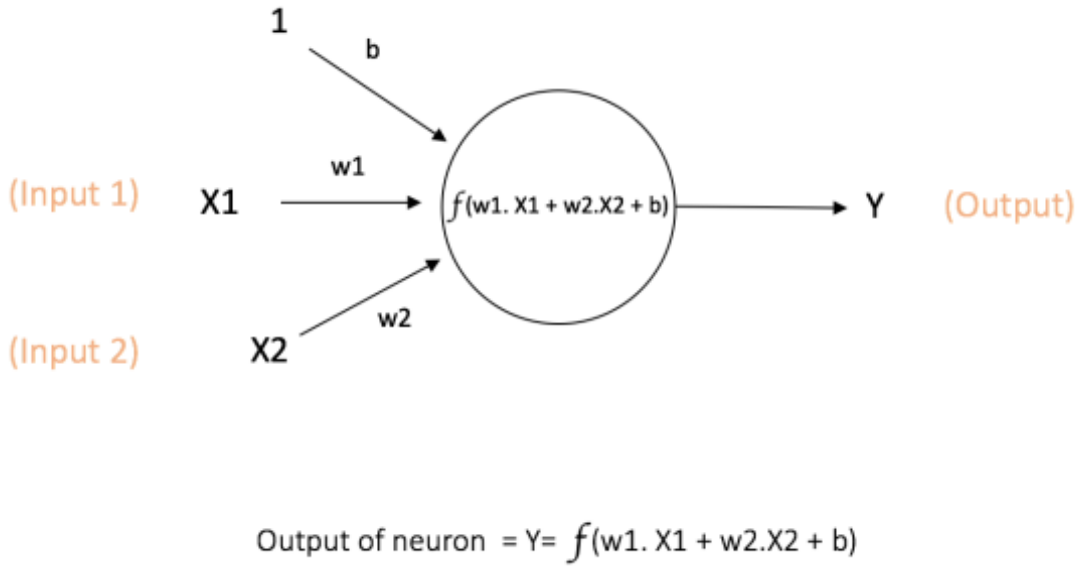
Joonis 5. Maa-aluse parkimissensori paigaldamisprotsess [2].

2.2 Tehisnärvivõrgud

Tehisnärvivõrgud pakuvad kiiret ja odavat lahendust erinevatele probleemidele ning on üks võimalikest tehnoloogiatest, mida saab lahendusena kasutusele võtta. Tehisnärvivõrk elimineeriks vajaduse investeerida parkimissensorite süsteemi paigaldusse ja hooldusesse. Selle asemel saab kasutada parkimisplatsist reaalajas videokaadreid, mida saab töödelda tehisnärvivõrguga ning tulemuseks tuvastada parkimiskohtade vastavad seisundid. Süvaõppe tehnikad on muutnud tehisintellekti maailma, ning tehisnärvivõrgud on tänaseks suutelised inimestest paremini lahendama näo - ja objektide tuvastusülesandeid [4].

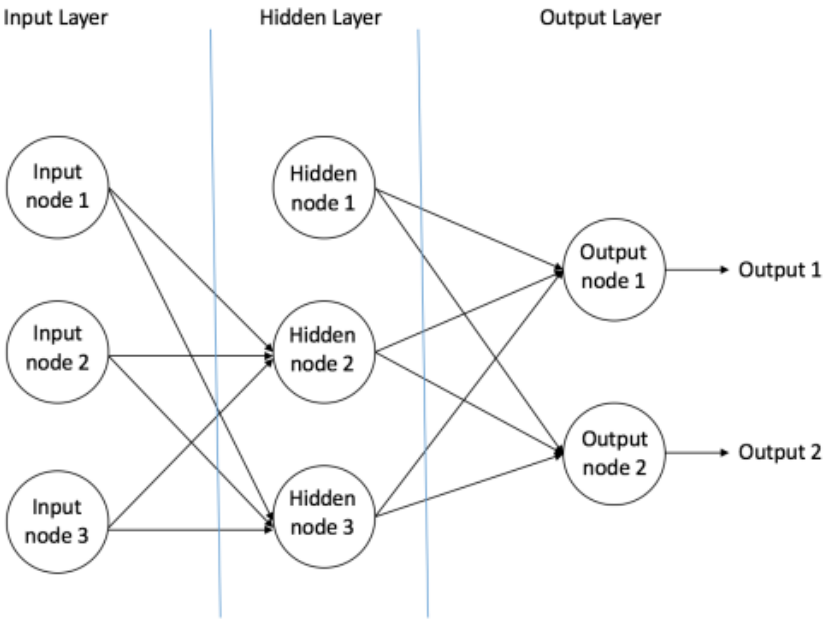
Tehisnärvivõrk koosneb mitmetest omavahe ühendatud neuronitest, mida nimetatakse ka sõlmedeks [5]. Üks sõlm koosneb sisenditest, väljundist ning sõlme sees olevatest kaaludest ja aktivatsioonifunktsioonist [5]. Sõlmed saavad sisendit teistelt sõlmedelt või välistest allikatest [5]. Igale sisendile määratakse kaal vastavalt selle sisendi suhtelisele

tähtsusele teiste sisendite suhtes [5]. Sõlme väljundiks on aktivatsioonifunktsiooni tulemus kaalutud sisendite summast [5].



Joonis 6. Tehisnärvivõrgu sõlm [6].

Sõlmed jagunevad erinevateks kihtideks, mis annavad meile tervikliku tehisnärvivõrgu. Esimeseks kihiks on sisendkiht, kus arvutusi ei tehta ning informatsioon antakse edasi järgmisele kihile [5]. Järgnevaid kihte nimetatakse varjatud kihtideks, kus toimub vahepealne informatsiooni töötlemine või arvutamine [5]. Viimases kihis ehk väljundkihis kasutatakse aktivatsioonifunktsiooni, mis annab meile soovitud formaadis väljundi [5].



Joonis 7. Kolme kihiline tehisnärvivõrk [6].

Tehisnärvivõrke on võimalik treenida kahel peamisel viisil: järelevalvega ja järelevalveta. Järelevalvega treenimine nõuab kõigepealt treeningandmete kogumist ning õiget klassifitseerimist, näiteks ära sorteeritud piltide kogum vabadest ja hõivatud parkimiskohtadest [7]. Treeningu alustamisel määratakse sõlmede kaalud juhuslikult või kasutatakse algoritme, mis võtavad arvesse ka tehisnärvivõrgu suuruse, näiteks Xavier-i algoritm [8]. Treeningu eesmärgiks on leida parim grupp kaalude väärtusi, mille tulemusel väljundi täpsus tõuseks [7].

2.2.1 COCO andmestikuga eeltreenitud tehisnärvivõrk

COCO[1] (*Common Objects in Context*), pakub ulatuslikku andmestikku objektide tuvastamise ja segmenteerimise jaoks. Andmestik koosneb 1,5 miljonist objekti eksemplarist ning kõik objektid on omakorda jagatud vastavatesse kategooriatesse. Seega on võimalik tuvastada ühe kaadri pealt erinevaid objekte ja nende asukohti.

[1] http://cocodataset.org/#home

Joonis 8. Näidis COCO andmestikus olevast pildist ning seal tuvastatud autodest[1].

Käesolevale tööle lahendust pakkuva süsteemi loomisel katsetati parkimisplatsil objektide tuvastust Tensorflow poolt COCO andmestiku põhjal eeltreenitud kõige mahukamat ja jõudluselt suurimat mudelit nimega faster_rcnn_nas[2], mis kasutab objektide tuvastamiseks Faster R-CNN[3] (*Faster Region-based Convolutional Network*) meetodit ning NAS (*Neural architecture search*) tehnikat.

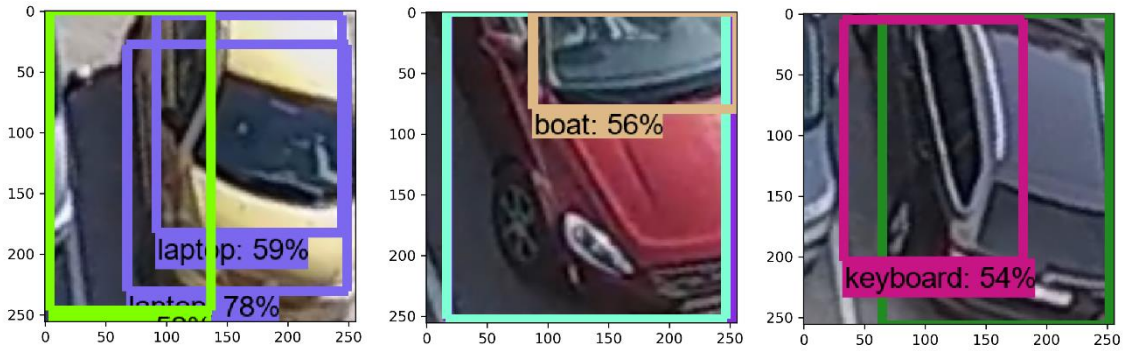
Mudelit katsetati üksikute väljalõigatud parkimiskohtade peal kasutades Tensorflow Object Detection[4] näiteprojekti, eeldades et kui mudel sõiduki objekti pildil ei tuvasta, siis on parkimiskoht vaba. Andes sisendiks hõivatud parkimiskohast pildi, ei suutnud mudel tuvastatud objekte sõidukina klassifitseerida.

[1] http://cocodataset.org/#explore

[2] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md#coco-trained-models

[3] https://arxiv.org/abs/1506.01497

[4] https://github.com/tensorflow/models/tree/master/research/object_detection#tensorflow-object-detection-api



Joonis 9. Hõivatud parkimiskohtade objekttuvastuse tulemused.

Kehva tuvastuse põhjuseks on mudeli mittespetsialiseeritud ülesehitus ning treeningprotsessi käigus on suur hulk ressursist suunatud võimalikult paljude objektide tuvastusele, mitte ühele kindlale objektile, et saavutada kõrge täpsus. Samuti mängivad rolli treeningandmestiku omadused. Näiteks COCO andmestik koosneb hea resolutsiooniga maapealse vaatenurgaga piltidest, mis ei lähe kokku antud töös kasutatud kaugelt jälgitavate, pealtvaates parkimiskohta piltidega.

2.2.2 CNRPark+EXT andmestikuga eeltreenitud tehisnärvivõrk

CNRPark+EXT[1] andmestik on loodud just parkimiskohtade visuaalse hõivatuse tuvastamiseks. Andmestik koosneb 144,965 hõivatud või mittehõivatud parkimiskohast erinevatel aegadel ja ilmastikuoludes.

[1] http://cnrpark.it/

Joonis 10. Näiteid vabadest ja hõivatud parkimiskohtadest CNRPark+EXT andmestikus [9].

CNRPark+EXT pakub ka erinevaid BVLC (*The Berkeley Vision and Learning Center*) Caffe[1] raamistikuga eeltreenitud tehisnärvivõrkude mudeleid. Selleks, et projekti käigus neid mudeleid kasutada saaks, tulevad need esmalt konverteerida Caffe mudelitest Tensorflow mudeliteks. Konverteerimiseks modifitseeris autor juhendi[2] järgi caffe-tensorflow[3] projekti, mis suudaks genereerida eraldiseisvad mudelid.

Süsteemi loomise jaoks valiti mudeli nimega mAlexNet-on-PKLot2Days, mis on loodud mAlexNet (mini AlexNet) arhitektuuriga [10]. Konverteeritud mAlexNet-on-PKLot2Days mudeli sisendiks on üks 256x256 piksli suurune pilt ühest parkimisplatsist ning mudeli väljund koosneb vaba ja hõivatud tõenäosusest, mille kogusumma on 1.

2.2.3 Isetreenitud tehisnärvivõrk

Isetreenitud tehisnärvivõrgu eesmärgiks on tõsta klassifitseerimistäpsust, kasutades treenimiseks sarnases keskkonnas kogud andmeid. Kuna mAlexNet-on-PKLot2Days mudel oli treenitud Itaalias kogutud andmete peal, siis Eesti kliimat arvestades langeks talveperioodidel klassifitseerimistäpsus märkimisväärselt.

[1] https://caffe.berkeleyvision.org/

[2] https://ndres.me/post/convert-caffe-to-tensorflow/

[3] https://github.com/ethereon/caffe-tensorflow

Isetreenitud tehisnärvivõrk annaks võimaluse täiendada treeningandmete hulka, lisades aastaaegadega seonduvad olukorrad ja erijuhtumid, näiteks talvel kokku aetud lumehanged, mis võivad samuti parkimiskoha hõivata.

3 Tehniline lahendus

Selles peatükis kirjeldatakse süsteemi ülesehitust. Süsteem on üles ehitatud Anaconda[1] platvormil loodud virtuaalses keskkonnas, mis on konfigureeritud kasutama programmeerimiskeeleks Pythonit[2]. Keskkonda on installeeritud OpenCV[3] teek, mida kasutatakse reaalajas parkimisplatsi kaadrite kättesaamiseks ning nende kaadrite töötlemiseks. Parkimiskohtade koordinaatide rotatsiooni ning muude arvutuste jaoks kasutatakse installeeritud Numpy[4] teeki. Erinevate treenitud tehisnärvivõrkude kasutamiseks kasutatakse Tensorflow[5] teeki ning tehisnärvivõrgu treenimiseks kasutatakse Jupyter[6] keskkonda koos Keras[7] teegiga.

Süsteemi sisendiks on videokaader parkimisplatsist ning andmed kaadril olevate parkimiskohtade koordinaatidest. Koordinaatide põhjal lõigatakse kaadrist välja kõik parkimiskohad ning kontrollitakse, et parkimiskoha pildisuurus oleks 256x256 pikslit, mis on vajalik nii mAlexNet-on-PKLot2Days mudeli kui ka isetreenitud tehisnärvivõrgu mudeli jaoks.

3.1 Konvolutsiooniline tehisnärvivõrk

Kuna eesmärgiks on tuvastada pildi järgi kas parkimiskoht on vaba või hõivatud, siis selle probleemi lahendamiseks on mAlexNet-on-PKLot2Days mudel treenitud mAlexNet konvolutsioonilise tehisnärvivõrguna [9]. Edasi vaadeldakse antud mAlexNet-on-PKLot2Days mudeli struktuuri ja töötamise tervet protsessi, alustades sisendväärtustest ning lõpetades väljundi aktivatsioonifunktsiooniga.

---

[1] https://www.anaconda.com/

[2] https://www.python.org/

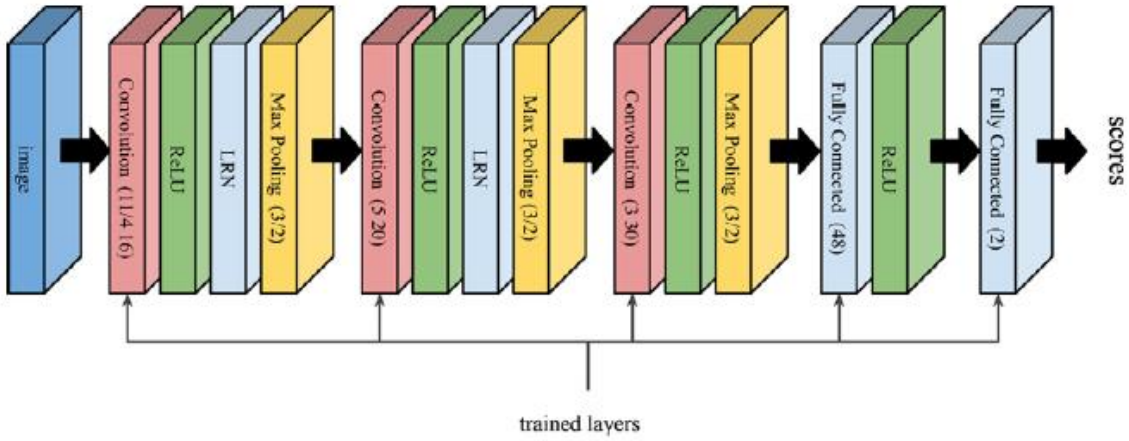
[3] https://opencv.org/

[4] https://www.numpy.org/

[5] https://www.tensorflow.org/

[6] https://jupyter.org/

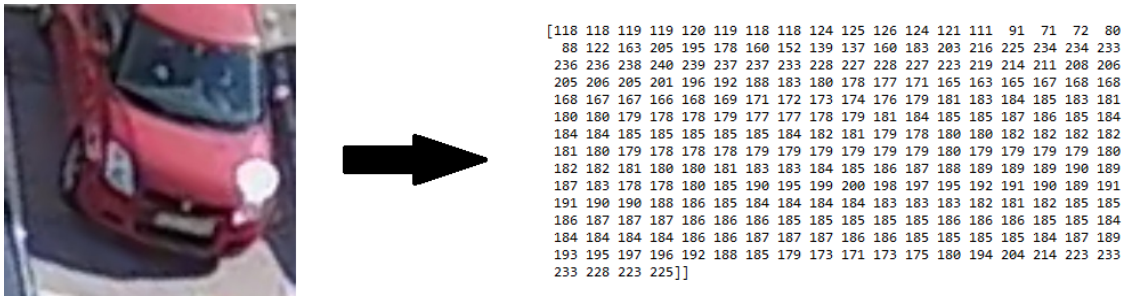
[7] https://keras.io/



Joonis 11. mAlexNet-on-PKLot2Days mudeli struktuur [9].

3.1.1 Sisend

Tehisnärvivõrgu sisendiks on 256x256 piksli suurune RGB pilt parkimiskohast, mida saab esitada pikslivääruste massiivina 256x256x3, kus 3 viitab RGB väärtustele. Igale massiivis olevale indeksile antakse väärtus 0-225, mis iseloomustab pikslite intensiivsust selles punktis. [11]



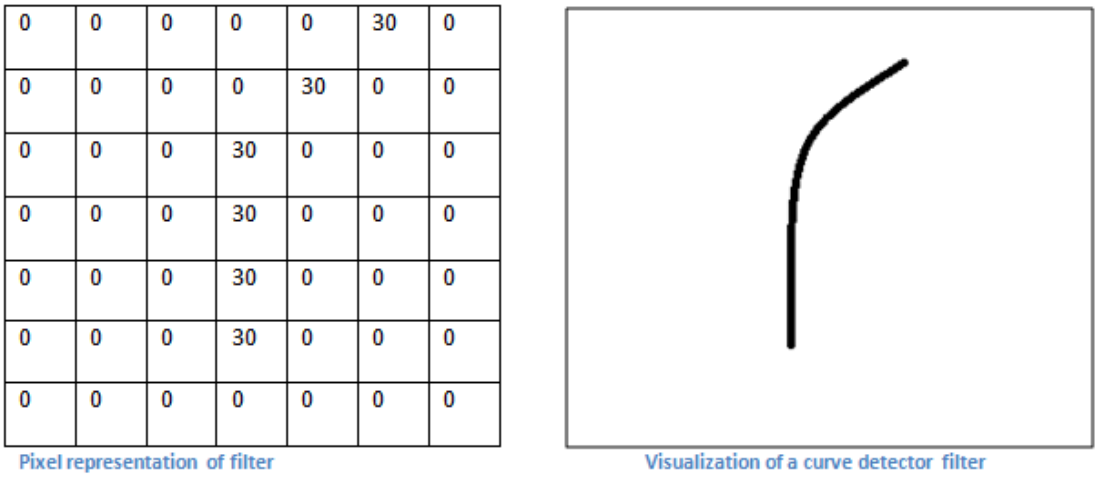
Joonis 12. Sisendpilt pikslivääruste massiivina.

3.1.2 Konvolutsiooniline kiht

Konvolutsiooniliste tehisnärvivõrkude, samas ka mAlexNet-on-PKLot2Days mudeli, esimeseks varjatud kihiks on konvolutsiooniline kiht. Selle asemel, et tehisnärvivõrgu sisendiks anda terve pildi massiivi, tükeldatakse see väiksemateks kattuvateks osadeks. Konvolutsioon suudab säilitada pikslite ruumilise suhte, õppides ära tundma pildi omadusi läbi väiksemate sisendite [11].

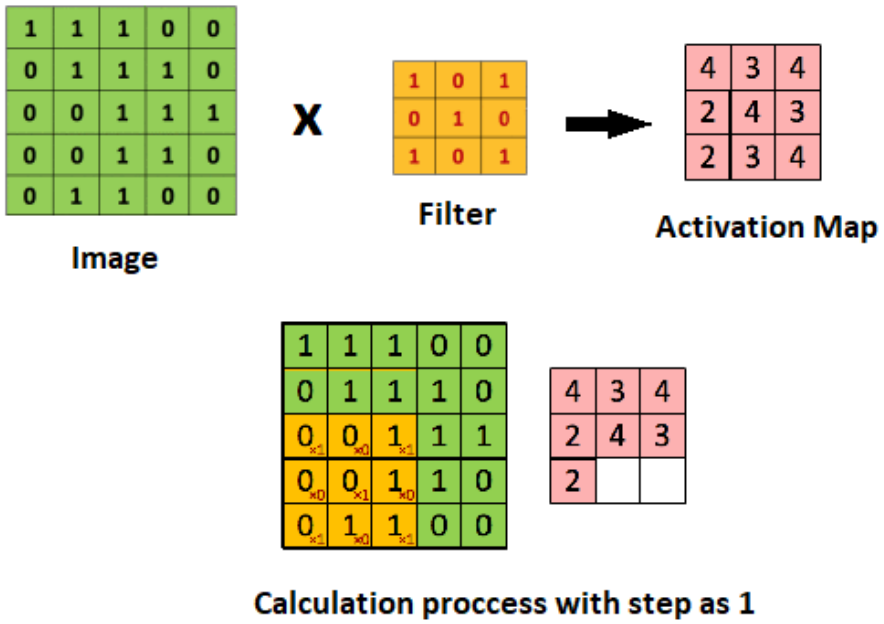
Konvolutsioonilises kihis luuakse kõigepealt etteantud arv kerneleid ehk filtreid, mis töötavad pildi tunnusjoonte detektoritena. mAlexNet-on-PKLot2Days mudelis on 16 erinevat filtrit massiividega 11x11x3. Pildi tunnusjoonteks võivad olla näiteks värvid,

sirgjooned või kõverjooned. Filtrite väärtused ja kaalud leitakse tehisnärvivõrgu treenimise käigus. Mida rohkem filtreid meil on, seda paremini suudab tehisnärvivõrk piltide omadusi ja mustreid tuvastada.



Joonis 13. Filter, mis tuvastab kõverjoont [11].

Iga loodud filter liigub vastavalt antud sammule üle sisendpildi ning arvutab sisendpildi ja filtri väärtuste korrutised hetkepiirkonnas. Kõik sammujärgsed korrutised loovad uue massiivi, mida nimetatakse tunnuskaardiks. [12]



Joonis 14. Tunnuskaardi arvutamise protsess [12].

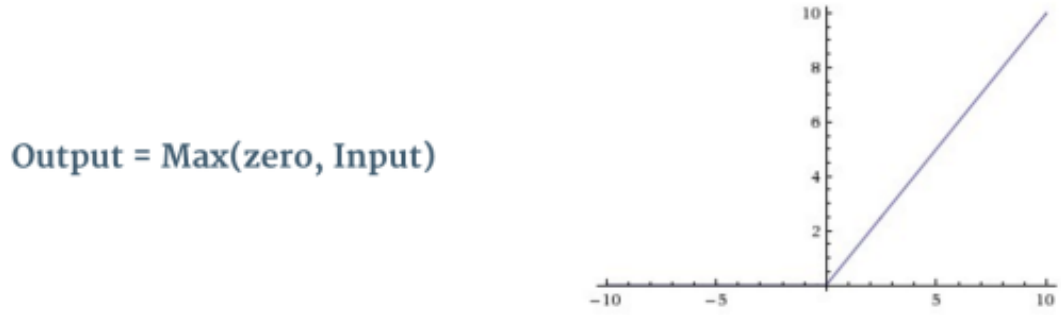
Tunnuskaartide massiivi suurus sõltub filtrite arvust, kuna iga filter loob uue tunnuskaardi ning filtri suurusest ja sammu pikkusest. mAlexNet-on-PKLot2Days

mudeli konvolutsioonilise kihi väljundiks on tunnuskaartide massiiv suurusega 36x36x16, kuna sisendpildi suurus on 256x256, filtri suurus on 11x11, filtri liikumissamm üle sisendpildi on 4 ning filtreid kokku on 16.

Tehisnärvivõrkude esimene konvolutsiooniline kiht suudab tuvastada ainult madala taseme tunnuseid, seega esimese kihi väljud kirjeldab ainult originaalpildi asukohti, kus ilmuvad filtrite poolt leitud madala taseme omadused. Selleks, et tuvastada kõrgema taseme omadusi, peame saadud tunnuskaartide massiivile rakendama järgmise konvolutsioonilise kihi filtrid. Kõrgema taseme filtrid võivad olla näiteks joonte kombinatsioonid, ringid või ruudud. [11]

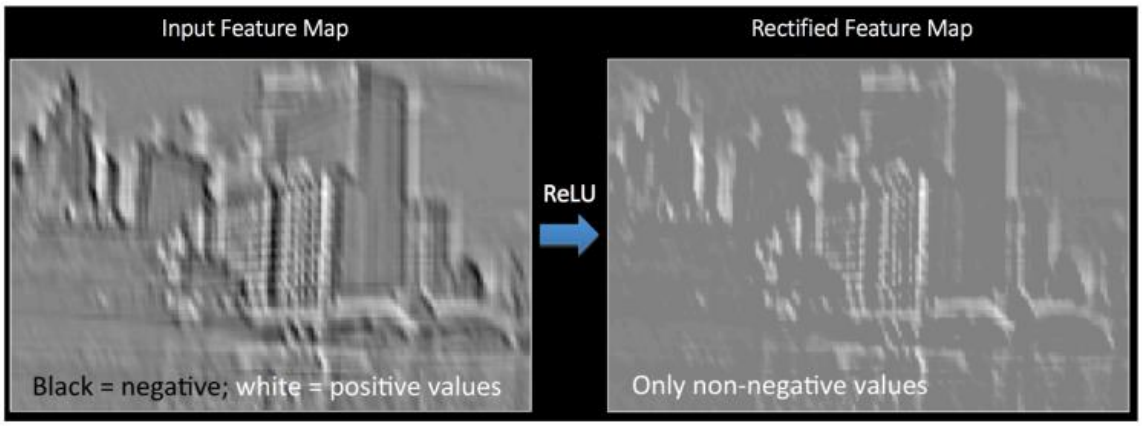
3.1.3 ReLU

Konvolutsiooniliste kihtide vahel kasutatakse ReLU (*Rectified Linear Unit*) operatsiooni, mis rakendatakse sisendi igale väärtusele. ReLU vahetab kõik negatiivsed väärtused nulliga. [12]



Joonis 15. ReLU operatsioon [12].

ReLU operatsiooni eesmärgiks on tõsta sisendi mittelineaarsust, kuna pildid on loomult mittelineaarsed. Mittelineaarsuse all mõeldakse näiteks pikslitevahelisi üleminekuid ja piirjooni. Konvolutsioonilise kihi väljund sisaldab filtrite tulemusel uduseid piirjooni ning lineaarseid üleminekuid tumedatest värvidest heledate värvideni. ReLU kasutamisega saame pildilt eemaldada kõik negatiivsed, tumedad elemendid ning selle tulemusel tõstame pildi mittelineaarsust, muutes piirjooned selgemaks ning värvide üleminekud äkilisemaks. [13]



Joonis 16. ReLU operatsiooni kasutamine pärast konvolutsioonilist kihti [12].

3.1.4 Lokaalse vastuse normaliseerimine

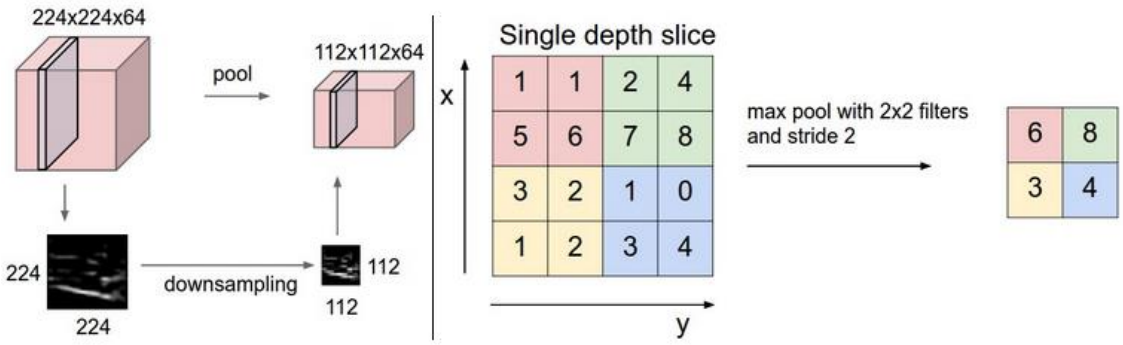
Local Response Normalization (LRN) tuleb kasutusele pärast ReLU neuronite väljundite arvutamist, kuna ReLU ei vaja sisendite normaliseerimist. Kui ReLU väljundid on kõikides kohalikes naabruskondades ühtlaselt suured, siis LRN summutab need väljundid ja kui me normaliseerimise aktiveeritud neuroni kohaliku naabruskonna ümber, siis muutub see neuron veelgi tundlikumaks võrreldes tema naabritega [14].

3.1.5 Maksimaalne ahenduskiht

Maksimaalse ahenduskihi (*max pooling*) ülesandeks on järk-järgult vähendada kujutise ruumilist suurust, et vähendada parameetrite ja arvutuste hulka. *Max pooling* võetakse perioodiliselt kasutusele pärast konvolutsioonilisi kihte. [15]

Max pool võtab etteantud suurusega maatriksist suurima väärtuse. Maatriks paigutatakse sarnaselt tunnuskaartide loomisel sisendiks olevale massiivile ning nihutatakse etteantud sammu võrra. [16]

Populaarsemad *max pooling* variatsioonid on kattuvad (*overlapping*) ahenduskihid, kus ahenduskihi maatriksi suuruseks on 3x3 ning liikumissammuks 2 ja tavalised ahenduskihid, kus maatriksi suuruseks on 2x2 ning liikumissamm on samuti 2 [15]. mAlexNet-on-PKLot2Days mudel kasutab kattuvat maksimaalset ahenduskihti.



Joonis 17. Tavaline maksimaalne ahenduskiht [15].

3.1.6 Täielikult ühendatud kiht

Täielikult ühendatud kiht (*Fully Connected Layer*) muudab sisendmassiivi N dimensiooniliseks vektoriks, kus N on etteantud klassifikatsioonide arv. Antud tehisnärvivõrgus on 2 klassifikatsiooni „free“ ja „busy“, ehk parkimiskoha hetkeseisundid. [11]

Täielikult ühendatud kiht vaatab eelmisest kihist saadud tunnuskaarte, mis nüüdseks kujutavad kõrgetaseme omadusi, ning määrab ära millised omadused kattuvad kõige rohkem antud klassifikatsiooniga. Näiteks „hõivatud“ klassifikatsiooni alla kuuluvad kõrgetaseme omadused nagu autorehvid, tuled, kapott jne.

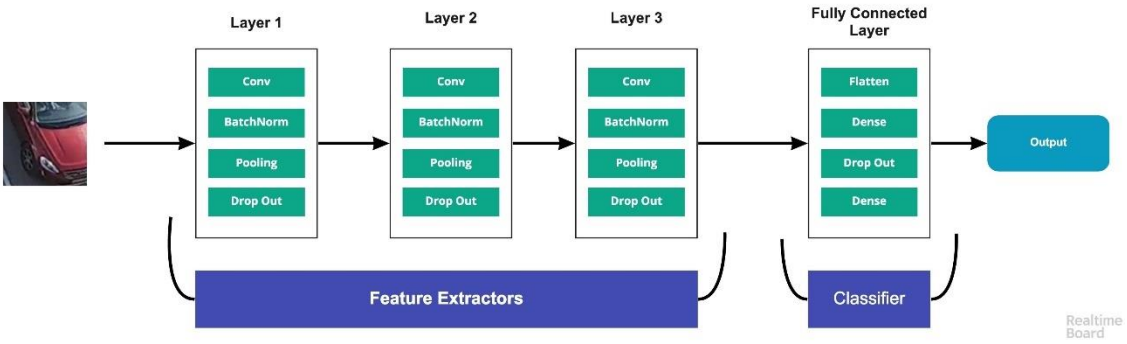
3.1.7 Väljund ja softmax aktivatsioonifunktsioon

Täielikult ühendatud kihist saadud väljund muudetakse vastavalt valitud aktivatsioonifunktsioonile õigesse formaati. mAlexNet-on-PKLot2Days mudel kasutab väljundi formaadi loomiseks softmax aktivatsioonifunktsioon, mis muudab klassifikatsioonide väärtused tõenäosusteks ning kõikide tõenäosuste summa on alati 1. Seega tehisnärvivõrgu väljundiks on alati vektor suurusega 2, kus esimene indeks näitab „free“ klassifikatsiooni tõenäosust ning teine indeks „busy“ klassifikatsiooni tõenäosust.

3.2 Konvolutsioonilise tehisnärvivõrgu treenimine

Selles peatükis kirjeldatakse konvolutsioonilise tehisnärvivõrgu treenimise etappe. Tehisnärvivõrgu loomisel võeti aluseks autori poolt modifitseeritud Kerase

konvolutsioonilise tehisnärvivõrgu loomise projekt[1] ning hiljem konverteeriti Kerase mudel Tensorflow mudeliks, kasutades olemasolevat keras_to_tensorflow[2] projekti.



Joonis 18. Isetreenitava konvolutsioonilise tehisnärvivõrgu struktuur, vt. lisa 2.

3.2.1 Treeningandmete ettevalmistus

Selleks, et üldse oleks võimalik tehisnärvivõrku treenida, on meil vaja treeningandmeid. Treeningandmeteks võivad olla näiteks tekstilõigud, pilt- või helifailid, millele on kaasa antud vastavad märgid, kuidas neid andmeid õigesti kategoriseerida.

Teenimise alustamisel valitakse ära ka treeningandmete ploki (*batch*) suurus. Kuna tehisnärvivõrgu treenimise ajal on ebaefektiivne korraga tervet treeningandmestikku sisendiks anda, siis tuleb see kõigepealt hajutada plokkideks [17]. Ploki suurus piirab tehisnärvivõrgule näidatavate testandmete hulka enne kaalude uuendamist. Sama piirang kehtib ka siis, kui tehisnärvivõrgu treenimine on lõppenud ning soovitakse uusi andmeid töödelda [17].

Ploki suuruse valimisel peab arvestama selle mõju tehisnärvivõrgu treenimisele. Valides Ploki suuruseks väikese väärtuse, muutub tehisnärvivõrgu treenimine kiiremaks, kuid arvutatud kaalude tulemused võivad muutuda ebastabiilseks. [18]

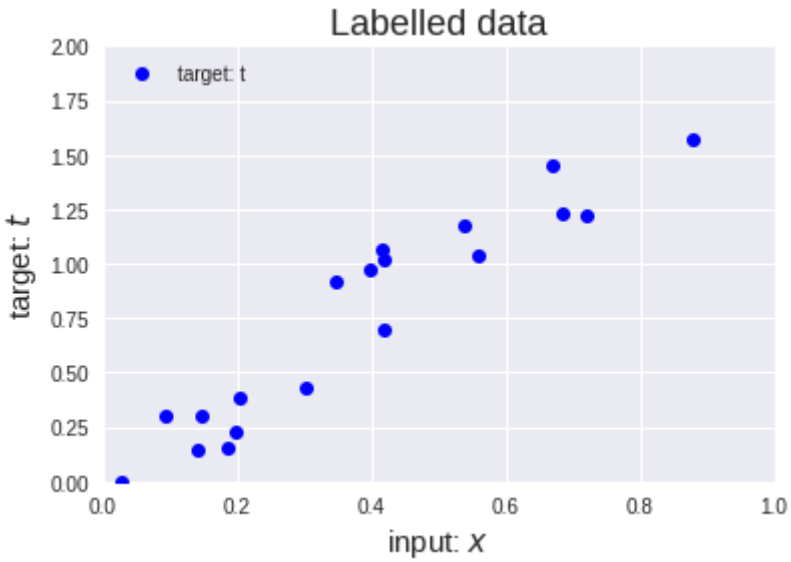
Tehisnärvivõrgu treenimisel tuleb ära määrata ka epohhi hulk. Ühe epohhi jooksul läbib terve treeningandmestik ühe täieliku treeningtsükli. Mida suurem on epohhide arv, seda rohkem muudetakse tehisnärvivõrgus kaalude väärtuseid. [19]

[1] https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification

[2] https://github.com/amir-abdi/keras_to_tensorflow

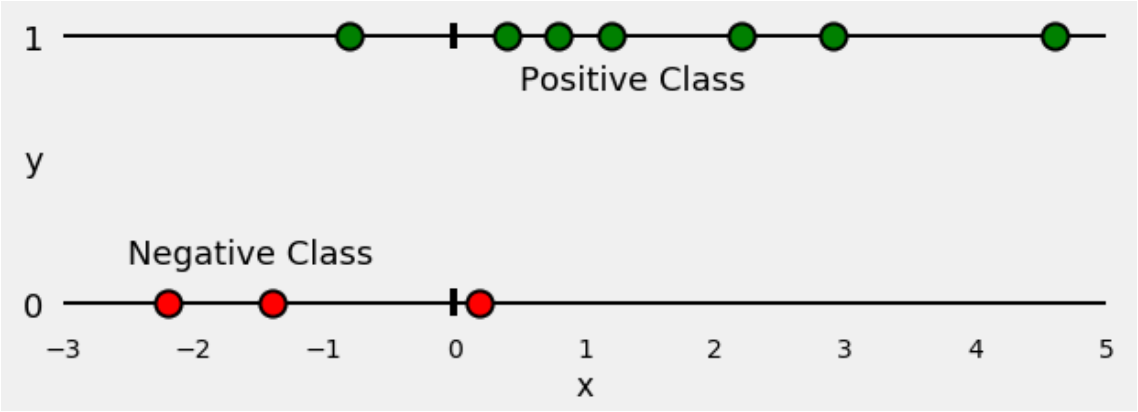
3.2.2 Gradientlaskumine ja kaotuse funktsioon

Gradientlaskumine (*Gradient Descent*) on iteratiivne optimeerimisalgoritm, mida kastutatakse masinõppes parimate neuronite kaalude leidmiseks [19]. Optimaalse tulemuse saamiseks proovitakse leida kaalude kogum, mis sobib kõige paremini sisendite ja tulemuste suhete jaoks [20].



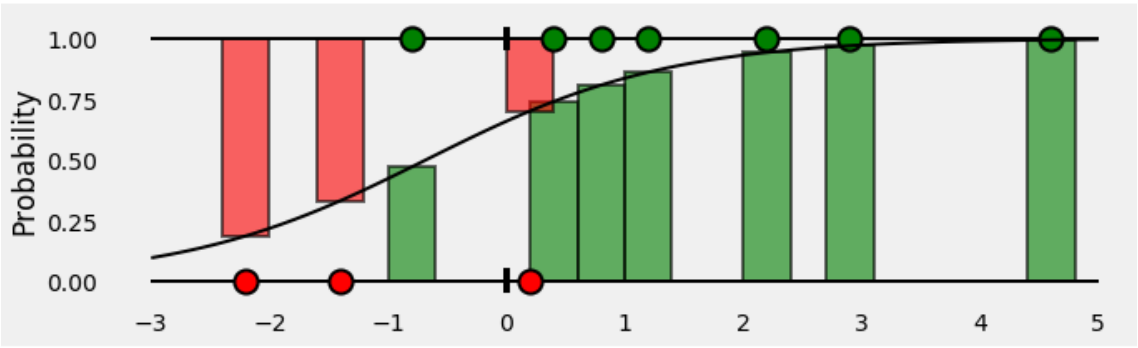
Joonis 19. Märgistatud andmete sisendite ja väljundite suhted, mille kohta tuleb leida kõige optimaalsem mudel [20].

Treeningu alguses on kaalude väärtused juhuslikud ning gradientlaskumise käigus hakatakse kaale uuendama kaotuse funktsiooni (*loss function*) minimeerimise suunas [21]. Kaotuse funktsiooni tulemus ütleb meile kui lähedal oleme mudelile, mille sisendparameetrid vastavad väljud parameetritele [20]. Projekti tehisnärvivõrgu treenimiseks kastutatakse Kerase binary_crossentropy kaotuse funktsiooni, kuna meil on ainult 2 erinevat klassifikatsiooni. binary_crossentropy kaotuse funktsioon alustab tööd andmete jagamisega, andes ühele klassile väärtuseks 1 ja teisele väärtuseks 0 [22].



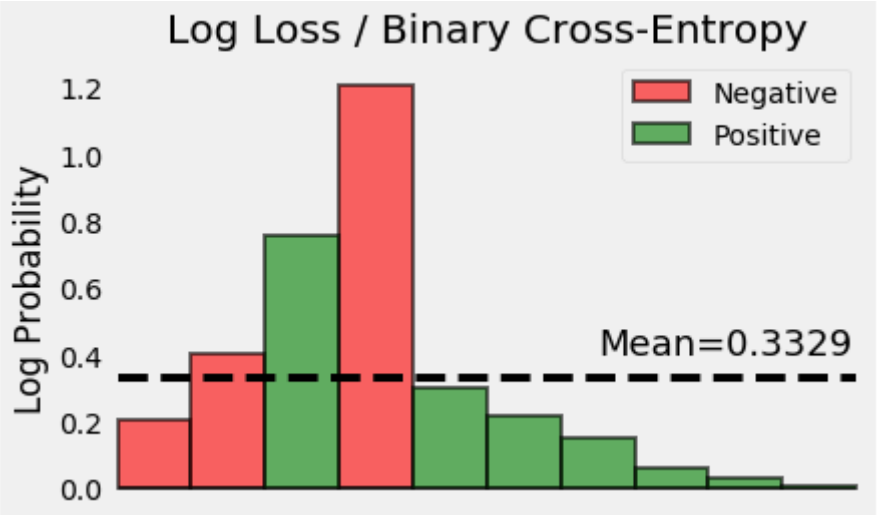
Joonis 20. Andmete jagumine kahte erinevasse klassidesse [22].

Seejärel leitakse sobiv sigmoidkõver, mis esindaks kahe klassi tõenäosusi vastavates punktides. Ühe klassi tõenäosused asuvad sigmoidkõvera all ning teise klassi tõenäosused asuvad sigmoidkõvera peal. [22]



Joonis 21. Andmete mõlema klassi tõenäosused [22].

Lõpuks on võimalik leida kaotuse funktsiooni väärtuse, arvutades negatiivse logaritmi kõikidest saadud väärtustest ning võttes nendest väärtuste keskmise. Negatiivne logaritm sobib seepärast, et mida lähemal on tõenäosuse väärtus õigele klassi algväärtusele, seda väiksema väärtuse annab meile negatiivne logaritm [22].

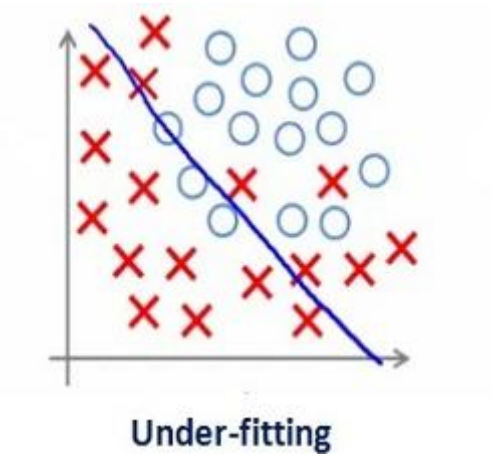


Joonis 22. Kaotuse funktsiooni keskmine väärtus pärast negatiivse logaritmi leidmist tõenäosustest [22].

3.2.3 Ala- ja ülesobitamine

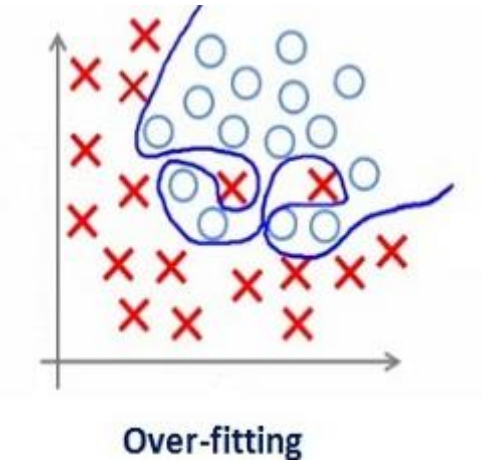
Treenimise jooksul on eesmärk leida selline mudel, mis suudaks uusi andmeid edukalt üldistada. Üldistamine viitab sellele, kui hästi on mudel suuteline omandatud mõisteid rakendama näidetele, mida treeningandmetes ei olnud.

Mudeli alasobitamine (*underfitting*) võib tekkida liiga väikese epohhi arvu või kehva treeningandmete tõttu. Mudel jääb liiga lihtsustatuks, ning seetõttu ei leita sobivat andmete suundumust ja mudeli täpsus langeb [23].



Joonis 23. Mudel, mis on liiga lihtne, et selgitada erinevusi [23].

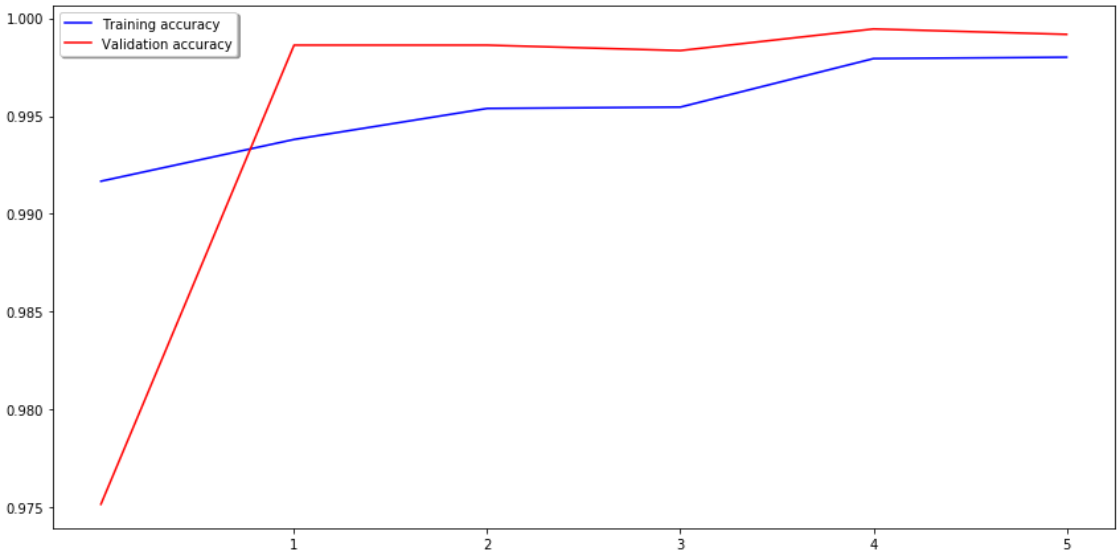
Ülesobitamine (*overfitting*) tekib treeningandmete liiga suurest kogusest ja mudeli mahtuvusest. Tulemuseks on mudel, mis õpib treeningandmetelt liiga hästi ning suudab hästi toimida ainult treeningandmetel [24]. Uute andmete esitamisel on mudeli täpsus madal.



Joonis 24. Mudel, mis on liiga detailne ning kus üldistamine on madal [23].

Ülesobituse tekkimist saab diagnoosida, jälgides mudeli täpsust nii treeningandmetel kui ka eraldiseisvatel valideerimisandmetel treeningprotsessi käigus [24].

Mudeli treenimise käigus jälgiti nii treeningandmete kui ka valideerimisandmete täpsust pärast igat epohhi lõppu, vt. lisa 3.



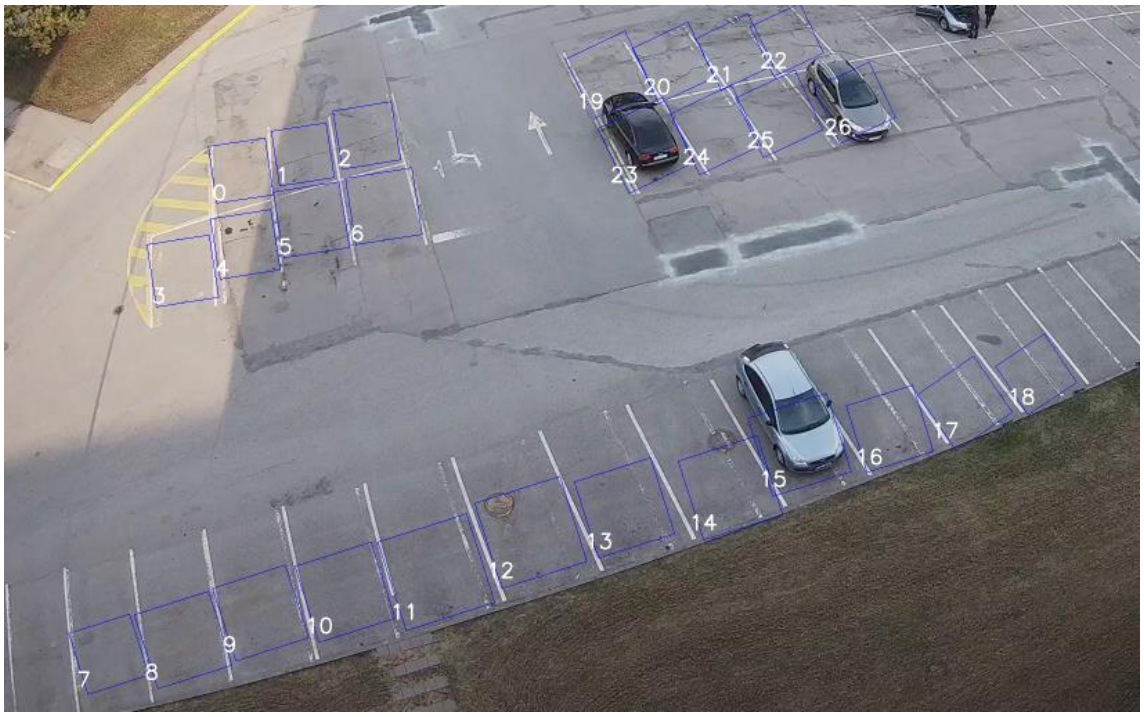
Joonis 25. Treeningandmete ja valideerimisandmete täpsus pärast igat epohhi.

4 Tulemused

Selles peatükis vaadeldakse nii olemasoleva mAlexNet-on-PKLot2Days mudeli kui ka isetreenitud mudeli täpsust, tuues välja protsessi käigus leitud nõrkused ja võimalikud lahendused.

4.1 mAlexNet-on-PKLot2Days mudeli täpsus

mAlexNet-on-PKLot2Days mudelit katsetati 27 erineva parkimiskoha peal, mis võeti Tallinna Tehnikaülikooli väliparkla kaadritest. Parkimiskohti jälgiti kahe tööpäeva jooksul ning parkimiskohtadest tehti pilt iga viie minuti järel. Kõik pildid sorteeriti vastavalt mudeli väljundile vaba või hõivatud siltidega kaustadesse.



Joonis 26. Konfigureeritud parkimiskohad kaaderpildis. Iga parkimiskoht on nummerdatud ning sinised kastid määravad ära kaadril väljalõigatavad piirkonnad, mida hiljem kastutatakse tehisnärvivõrgu sisenditena.

Kokku tehti kahe tööpäeva jooksul 18928 pilti, millest õigesti oli ennustatud 17991 ning mudeli täpsuseks jäi 95%, vt. lisa 1. Mudeli täpsuse määramisel peab arvesse võtma ka

kas valed ennustused oli vale-positiivset (*false positive)* või vale-negatiivset (*false negative)* tüüpi.

Vale-positiivsed on juhtumid, kus mudel on ennustanud parkimiskoha seisundiks „vaba“, kuid tegelikkuses on parkimiskoht hõivatud. Vale-negatiivsed on juhtumid, kus mudel on ennustanud parkimiskoha seisundiks „hõivatud“, kuid tegelikkuses on parkimiskoht vaba.

Valedest ennustusest vale-positiivseid juhtumeid oli 157 ning vale-negatiivseid juhtumeid 780. Kasutajakogemuse poolelt tekitavad vale-positiivsed juhtumid rohkem kahju, kui vale-negatiivsed juhtumid, kuna kasutaja loodab näha antud informatsiooni põhjal vaba parkimiskohta, mida ta saaks kasutada.

Kogutud katseandmete põhjal on võimalik välja selgitada ka valede ennustuste põhjuseid. Vale-positiivsed juhtumid tekkisid kaamera kehva nurga, parkimiskoha kauguse, heleda valguse ning auto osalise väljalõike kombinatsioonist. Vale-negatiivsed juhtumid tekkisid aga keskpäeval tekkinud varjudest või öösel vähese valguse tõttu.

Joonis 27. Näiteid vale-positiivsetest (üleval) ja vale-negatiivsetest (all) juhtumitest.

4.2 Isetreenitud tehisnärvivõrgu mudeli täpsus

Isetreenitud tehisnärvivõrgu jaoks kasutati kahe tööpäeva jooksul Tallinna tehnikaülikooli peaparklast kogutud parkimiskohtade andmestikku, mis oli eelnevalt kategoriseeritud mAlexNet-on-PKLot2Days mudeli poolt ning kus valed väärtused asendati õigetega.

Isetreenitud tehisnärvivõrgu mudeli ja mAlexNet-on-PKLot2Days mudeli võrdlemiseks koguti uus treeningandmetest erinev parkimiskohtade andmestik, mis koosnes 11338 vabast ja 3998 hõivatud parkimiskohast.

Tabel 1. mAlexNet-on-PKLot2Days mudeli ja isetreenitud tehisnärvivõrgu mudeli tulemused.

	Vaba	Hõivatud	Vale-positiivne	Vale-negatiivne	Täpsus
mAlexNet-on-PKLot2Days mudel	10212	3000	998	1126	86.15%
Isetreenitud tehisnärvivõrgu mudel	11302	3989	19	26	99.7%

5 Kokkuvõte

Käesoleva uurimistöö eesmärgiks oli koostöös Tallinna Tehnikaülikooliga luua süsteem, mis suudaks reaalajas videokaadrist tuvastada parkimiskohtade hõivatud või vaba seisundit.

Analüüsi käigus leiti, et probleemi lahendamiseks sobib kõige paremini konvolutsiooniline tehisnärvivõrk, mis on mõeldud just pilttuvastusprobleemide lahendamiseks.

Olemasolevate tehisnärvivõrkudega lahendused andsid piisavalt täpseid tulemusi, et luua mitmekesine treeningandmestik parkimiskohtade seisunditest kahe tööpäeva jooksul, kasutades Tallinna Tehnikaülikooli poolt installeeritud kaamerat. Treeningandmestiku loomisel parandati valesti tuvastatud parkimiskohtade seisundid õigete väärtustega.

Saadud andmestikku kasutati uue tehisnärvivõrgu loomisel, mis koosnes kolmest konvolutsioonilisest kihist, et tuvastada treenimise käigus vajalikud sisendite omaduste filtrid ning täielikult ühendatud kihist, mis andis tulemustele vastavad klassifikatsioonid.

Tulemuseks paranes parkimiskohtade seisundite tuvastuse täpsus 86-lt protsendilt 99 protsendini, kuna uus tehisnärvivõrk oli treenitud kasutatavate parkimiskohtade andmete peal, võttes arvesse kaamera nurka, kaugust ning eriolukordi, näiteks puude varjud ja vähene valgus.

Edasiarenduse järgmiseks etapiks on luua süsteemile rakendusliides, mis annaks päringute põhjal konfigureeritud parkimiskohtade seisunditest reaalajas informatsiooni. Samuti saab jätkata parkimiskohtadest andmestiku kogumisega ja tehisnärvivõrgu uuendamisega, lisades erinevatel aastaaegadel ja ilmastikuoludes kogutud andmeid.

Kasutatud kirjandus

[1] Smart Parking, „Vehicle detection sensors,“ [Võrgumaterjal]. Available: https://www.smartparking.com/smartpark-system/smart-sensors. [Kasutatud 14 mai 2019].

[2] PNI, „PlacePod Vehicle Detection Sensor Installation Guide,“ 14 september 2018. [Võrgumaterjal]. Available: https://www.pnicorp.com/download/pni-placepod-vehicle-detection-sensor-installation-guide/. [Kasutatud 14 mai 2019].

[3] Wise Moving, „Smartdevices – Overhead indicator sensors,“ [Võrgumaterjal]. Available: https://www.wisemoving.co/overhead-indicator-sensors. [Kasutatud 14 mai 2019].

[4] V. Shchutskaya, „What hides behind the effectiveness of neural networks and deep learning,“ 30 jaanuar 2017. [Võrgumaterjal]. Available: https://indatalabs.com/blog/effectiveness-neural-networks?cli_action=1557921299.549. [Kasutatud 15 mai 2019].

[5] D. Fumo, „A Gentle Introduction To Neural Networks Series—Part 1,“ 4 august 2017. [Võrgumaterjal]. Available: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc. [Kasutatud 5 mai 2019].

[6] Ujjwalkarn, „A Quick Introduction to Neural Networks,“ 9 august 2016. [Võrgumaterjal]. Available: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/. [Kasutatud 5 mai 2019].

[7] V. Bushaev, „How do we ‘train’ neural networks ?,“ 27 november 2017. [Võrgumaterjal]. Available: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73. [Kasutatud 5 mai 2019].

[8] R. Vasudev, „How to Initialize weights in a neural net so it performs well?—Super fast explanation for Xavier’s Random Weight Initialization,“ 28 mai 2018. [Võrgumaterjal]. Available: https://hackernoon.com/how-to-initialize-weights-in-a-neural-net-so-it-performs-well-3e9302d4490f. [Kasutatud 24 mai 2019].

[9] G. Amato, F. Carrara, F. Falchi, C. Gennaro ja C. Vairo, „Deep learning for decentralized parking lot occupancy detection,“ 15 april 2017. [Võrgumaterjal]. Available: https://www.sciencedirect.com/science/article/pii/S095741741630598X. [Kasutatud 16 mai 2019].

[10] G. Amato, F. Carrara, F. Falchi, C. Gennaro ja C. Vairo, „Car parking occupancy detection using smart camera networks and Deep Learning,“ 18 august 2016. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/abstract/document/7543901. [Kasutatud 15 mai 2019].

[11] A. Deshpande, „A Beginner's Guide To Understanding Convolutional Neural Networks,“ 20 juuli 2017. [Võrgumaterjal]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Kasutatud 5 mai 2019].

[12] Ujjwalkarn, „An Intuitive Explanation of Convolutional Neural Networks,“ 11

august 2016. [Võrgumaterjal]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Kasutatud 5 mai 2019].

[13] Super Data Science, „Convolutional Neural Networks (CNN): Step 1(b) - ReLU Layer,“ 17 august 2018. [Võrgumaterjal]. Available: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1b-relu-layer. [Kasutatud 7 mai 2019].

[14] P. Joshi, „What Is Local Response Normalization In Convolutional Neural Networks,“ 5 aprill 2016. [Võrgumaterjal]. Available: https://prateekvjoshi.com/2016/04/05/what-is-local-response-normalization-in-convolutional-neural-networks/. [Kasutatud 16 mai 2019].

[15] „Convolutional Neural Networks (CNNs / ConvNets),“ [Võrgumaterjal]. Available: http://cs231n.github.io/convolutional-networks/. [Kasutatud 7 mai 2019].

[16] narasimman, „Max Pooling,“ 29 juuni 2017. [Võrgumaterjal]. Available: https://machinelearningonline.blog/2017/06/29/max-pooling/. [Kasutatud 7 mai 2019].

[17] J. Brownlee, „How to use Different Batch Sizes when Training and Predicting with LSTMs,“ 15 mai 2017. [Võrgumaterjal]. Available: https://machinelearningmastery.com/use-different-batch-sizes-training-predicting-python-keras/. [Kasutatud 11 mai 2019].

[18] „What is batch size in neural network?,“ 22 mai 2015. [Võrgumaterjal]. Available: https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network. [Kasutatud 11 mai 2019].

[19] S. Sharma, „Epoch vs Batch Size vs Iterations,“ 23 september 2017. [Võrgumaterjal]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Kasutatud 16 mai 2019].

[20] P. Roelants, „Machine Learning 101,“ 16 mai 2017. [Võrgumaterjal]. Available: https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a. [Kasutatud 16 mai 2019].

[21] A. S. Walia, „Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent,“ 10 juuni 2017. [Võrgumaterjal]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f. [Kasutatud 16 mai 2019].

[22] D. Godoy, „Understanding binary cross-entropy / log loss: a visual explanation,“ 21 november 2018. [Võrgumaterjal]. Available: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a. [Kasutatud 16 mai 2019].

[23] GeeksforGeeks, „Underfitting and Overfitting in Machine Learning,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning. [Kasutatud 18 mai 2019].

[24] J. Brownlee, „How to Avoid Overfitting in Deep Learning Neural Networks,“ 17 detsember 2018. [Võrgumaterjal]. Available: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error. [Kasutatud 18 mai 2019].

Lisa 1 – mAlexNet-on-PKLot2Days mudeli klassifitseerimiskatsete tulemused

Parkimiskoha nr	Vaba	Hõivatud	Vale-positiivne	Vale-negatiivne
0	419	233	1	46
1	432	227	5	35
2	428	212	18	41
3	413	241	1	44
4	369	270	5	56
5	441	226	0	32
6	407	266	3	23
7	346	282	3	68
8	387	245	0	67
9	311	288	40	60
10	292	328	15	64
11	378	245	6	70
12	398	287	4	10
13	433	263	0	3
14	386	303	6	4
15	435	261	0	3

Parkimiskoha nr	Vaba	Hõivatud	Vale-positiivne	Vale-negatiivne
16	417	272	9	1
17	514	166	0	19
18	627	63	0	9
19	470	202	0	27
20	416	262	1	20
21	433	232	7	27
22	493	233	1	26
23	449	233	2	15
24	452	225	14	8
25	443	238	16	2
26	424	275	0	0

	Vaba	Hõivatud	Vale-positiivne	Vale-negatiivne
Kokku	11413	6578	157	780

Lisa 2 – Isetreenitud tehisnärvivõrgu struktuur

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 254, 254, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 254, 254, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 127, 127, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 125, 125, 64)      18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 125, 125, 64)      256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 62, 62, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 60, 60, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 60, 60, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 30, 30, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 115200)            0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               58982912
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 59,079,617
Trainable params: 59,078,145
Non-trainable params: 1,472
_________________________________________________________________
```

Lisa 3 – Isetreenitud tehisnärvivõrgu treenimistulemused

```
Epoch 1/6
1453/1453 [==============================] - 6281s 4s/step - loss: 0.0297 -
acc: 0.9917 - val_loss: 0.1106 - val_acc: 0.9752
Epoch 2/6
1453/1453 [==============================] - 6255s 4s/step - loss: 0.0214 -
acc: 0.9938 - val_loss: 0.0053 - val_acc: 0.9986
Epoch 3/6
1453/1453 [==============================] - 6267s 4s/step - loss: 0.0175 -
acc: 0.9954 - val_loss: 0.0067 - val_acc: 0.9986
Epoch 4/6
1453/1453 [==============================] - 6280s 4s/step - loss: 0.0173 -
acc: 0.9955 - val_loss: 0.0045 - val_acc: 0.9983

Epoch 00004: ReduceLROnPlateau reducing learning rate to
0.0005000000237487257.
Epoch 5/6
1453/1453 [==============================] - 6282s 4s/step - loss: 0.0076 -
acc: 0.9979 - val_loss: 0.0012 - val_acc: 0.9994
Epoch 6/6
1453/1453 [==============================] - 6656s 5s/step - loss: 0.0084 -
acc: 0.9980 - val_loss: 0.0028 - val_acc: 0.9992
```