jamk.fi

Kuva-aineiston generointi neuroverkkoa varten

Joona Tenhunen

Opinnäytetyö
Toukokuu 2021
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Tenhunen, Joona	Opinnäytetyö, AMK	Toukokuu 2021
	Sivumäärä 68	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Kuva-aineiston generointi neuroverkkoa varten

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Mika Rantonen, Antti Häkkinen

Toimeksiantaja(t)
JAMK IT-instituutti, Eppu Heilimo

Tiivistelmä

Syväoppimisen avoimen lähdekoodin kirjastot ja tietokoneiden laskentatehon kasvu mahdollistavat monenlaiset kuvantunnistusprojektit, myös kotikäyttäjille. Mutta esteeksi nousee usein laadukkaan koulutusaineiston saatavuus ja hankinta. Tavoitteena oli kehittää menetelmä tuottaa kuva-aineistoa liikennemerkeistä 3D-mallintamisen avulla, ja käyttää aineistoa konvoluutioneuroverkon koulutuksessa.

Kirjoitettiin Blender-mallinnusohjelmaan lisäosa, jonka avulla generoitiin useasta liikennemerkistä koostuva datasetti, jonka avulla koulutettiin konvoluutioneuroverkkoa. Vertailuarvona käytettiin ennustetarkkuutta, kun saman rakenteista konvoluutioneuroverkkoa koulutettiin oikeilla kuvilla. Tarkkuutta testattiin erillisellä kuvajoukolla, jota ei käytetty mallien koulutuksissa.

Generoidun kuvajoukon avulla koulutettu malli ei pärjännyt vertailussa oikeiden kuvien avulla koulutettuun malliin. Kuitenkin kuvien keinotekoisella muokkauksella tulokset paranivat hieman, joka osoittaa, että menetelmää kehittämällä saattaa olla mahdollista tuottaa vertailussa paremmin menestyviä kuva-aineistoja.

Testitulosten lisäksi toimeksiantajalle jäi käytettäväksi työssä käytetty Blender-lisäosa, jonka avulla voi generoida lisää erilaisia kuvia liikennemerkeistä.

Avainsanat (asiasanat)
Koneoppiminen, syväoppiminen, 3D-mallinnus

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Tenhunen, Joona	Bachelor's thesis	May 2021
		Language of publication: Finnish
	Number of pages: 68	Permission for web publication: x

Title of publication

Generating image data for artificial neural network

Degree programme

Information and Communication Technology

Supervisor(s)

Mika Rantonen, Antti Häkkinen

Assigned by

JAMK Institute of Information Technology, Eppu Heilimo

Abstract

Deep learning opensource libraries and growth of calculation power enables various image recognition projects, also for home users. Often availability and gathering of quality training data becomes an obstacle. The aim was to develop a method to generate image data of traffic signs, with help of 3D-modelling and use the generated data to train a convolutional neural network.

An add-on for modelling program Blender was written, and it was used to generate data set of multiple traffic signs. Data set was used to train a convolutional neural network. To evaluate the performance of the model, a same structured convolutional neural network was trained, using real images. Accuracy was tested with a different image set, which was not used in the trainings.

The model, which was trained with generated images, did not fair in comparison with the model, which was trained with real images. However, when images were augmented artificially results showed some improvement, which implies that improving the method it might be possible to generate image data, that would be more successful in comparison.

Besides the results of the tests, assigner received the Blender-addon, which can be used to generate more different images of traffic signs.

Keywords/tags (subjects)

Machine learning, deep learning, 3D-modelling

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

1 Johdanto

Syväoppiminen on kehittynyt viime vuosina huimasti, ja vapaanlähdekoodin teknologioiden kehittyminen on tuonut myös kuvantunnistukseen tarvittavia työkaluja kaikkien saataville. Onnistuneen koneoppimisprojektin edellytyksenä on edustava ja riittävän laadukas data. Ennusteita tekevät mallit oppivat esimerkeistä, joten raakadatan keräyksen lisäksi esimerkiksi valokuviin täytyy liittää tieto oikeasta ennusteesta koulutusta varten.

Joskus esteeksi voi kuitenkin muodostua koulutukseen tarvittavan sopivan aineiston vähyys ja laatu. Tässä työssä pyrittiin tuottamaan koulutusaineistoa minimaalisista lähtökohdista ja pyrittiin sen avulla kouluttamaan malli, joka yleistyy oikeisiin kuviin. Ajatuksena on, että kuvien keräämiseltä vältytään ja niiden luokittelu tapahtuu automaattisesti generoinnin yhteydessä.

Työn toimeksiantajana toimi Jyväskylän ammattikorkeakoulun IT-instituutti. It-instituutti kouluttaa ICT-alan AMK- ja YAMK-insinöörejä Lutakon kampuksella. Lisäksi se tarjoaa täydennyskoulutusta ja tilauksesta räätälöityjä koulutuksia yrityksille. It-instituutti on osa JAMK:n teknologiayksikköä, jossa opiskelee 2700 opiskelijaa, ja se työllistää 170 työntekijää. (Teknologiayksikkö N.d.)

2 Tutkimusasetelma

2.1 Opinnäytetyön tavoite

Opinnäytetyön tavoitteena oli kehittää menetelmä tuottaa kuva-aineistoa liikennemerkeistä konvoluutioneuroverkon kouluttamista varten, ja verrata niillä koulutetun neuroverkon suoriutumista liikennemerkkien luokittelussa, kun kohteena ovat kuvat oikeista liikennemerkeistä.

Tavoitteet voidaan esittää seuraavina kysymyksinä:

- Miten konvoluutioneuroverkko oppii?
- Millaista aineistoa sen kouluttaminen vaatii?
- Onko aineiston määrällä vaikutusta?
- Kuinka generoitu aineisto vertautuu reaalimaailman aineistoon?

2.2 Tutkimusmenetelmät

Tutkimusmenetelmät tavallisesti jaotellaan laadullisiin ja määrällisiin menetelmiin, mutta ne eivät ole toisensa poissulkevia, samassa tutkimuksessa voidaan käyttää molempia menetelmiä. Laadullinen tutkimus koostuu monenlaisista lähestymistavoista ja analysointi keinoista. (Vuori N.d Johdatus laadulliseen tutkimukseen ja verkkokäsikirjaan.)

Vaikka työssä on myös määrällisen tutkimuksen merkkejä, tutkimusote on laadullinen, kun pyritään ymmärtämään aineiston eri ominaisuuksien vaikutuksia neuroverkon koulutustuloksiin.

3 3D-mallintaminen

3.1 Tietokonegrafiikasta ja sen historiasta

Tietokonegrafiikka on tieteenhaara, joka keskittyy kuvan tai kuvien tuottamiseen tietokoneen avulla, sisältäen mallintamisen, renderöinnin ja animaation. Tietokonegrafiikan terminä keksi William Fetter vuonna 1960, työskennellessään Boeing-yhtiön palveluksessa. Hän ja hänen esimiehensä Vern Hudson käytti termiä kuvaamaan tietokoneohjattua piirturia, joka piirsi kuvasarjoja joko paperille, nauhalle tai filmille. William Fetterin työ on merkittävää 3D-grafiikan, erityisesti 3D-animaation, saralla. Hänen työnsä tuloksena tuotettu 3D-malli ihmisestä oli ensimmäinen laatuaan. (A Brief History of Computer Graphics 2004; Oppenheimer 2018; Beane 2012.)

Ensimmäinen tietokoneohjelma, joka hyödynsi graafista käyttöliittymää ja osoitinlaitetta, on Ivan Sutherlandin vuonna 1963 tekemä Sketchpad-ohjelma. Se käytti valo-

kynää tietokoneen näytöllä yksinkertaisten muotojen piirtämiseksi. Ohjelman on katsottu raivaavan tietä nykyaikaisille piirto- ja suunnitteluohjelmille, jotka hyödyntävät geometrisiä rajoitteita, esimerkiksi suoran viivan ja täydellisen ympyrän piirtämiseen. (Beane 2012.)

1970-luvulla tietokoneet pienenivät kokonsa puolesta ja saivat lisää nopeutta, ja idea virtuaalisesta 3D-pinnasta syntyi. Useat 3D-grafiikan perusperiaatteista, kuten varjostin (engl. *shader*) ja renderöinti (engl. *rendering*), syntyi kyseisenä vuosikymmenenä. (Beane 2012.)

1980-luku toi tietokoneet koteihin IBM PC ja Macintosh alustoilla, ja CAD ja kuvankäsittelyohjelmat seurasivat niitä. Myös useat animaatiotalot perustettiin 1980-luvulla ja 3D-animaatiot alkoivat näkyä elokuvissa. 1990-luvulla teknologian kehittyminen paransi 3D-grafiikoiden laatua ja 3D-animaatiot yleistyivät elokuvissa. 3D-grafiikkapiirit yleistyivät PC- ja pelikonsolimaailmassa, luoden uuden ajan 3D-pelejä. 2000-luvulla ihmiset olivat jo tottuneet näkemään laadukkaita ja uskottavia 3D-animaatioita ja visuaalisia efektejä. (Beane 2012.)

3.2 3D-mallintaminen

Jokainen tietokoneen renderöimä kuva vaatii kolme perimmäistä komponenttia: kolmiulotteisen tilan kuvauksen, yhden tai useamman valonlähteen, sekä kuvauksen kamerasta, joka katsoo tilaa. Tilan kuvaus koostuu yhdestä tai useammasta mallista tai 3D-rakenteesta. Usein ajatellaan että malli on itsenäinen osa, esimerkiksi kynä tai puu, ja tila on kokoelma näitä osia, jotka yhdessä muodostaa 3D-ympäristön. Jokainen malli sisältää kaksi kuvausta: matemaattisen kuvauksen mallin rakenteesta, sekä ohje siitä kuinka saadaan selville, miltä muoto näyttäisi valaistuna. (Badler & Glassner N.d.)

Mallin rakenteen kuvaus on periaatteessa geometrinen kuvaus. Se kertoo missä esine tai asia sijaitsee tilassa, ja missä ei. Jos kuvitellaan ilmassa leijuva tyhjä kahvimuki, ja katsotaan missä kukin tilan molekyyli on pysäytetyllä hetkellä, kukin yleisesti voidaan sanoa että molekyyli joko on osa kahvimukia tai ei ole. Ja jos

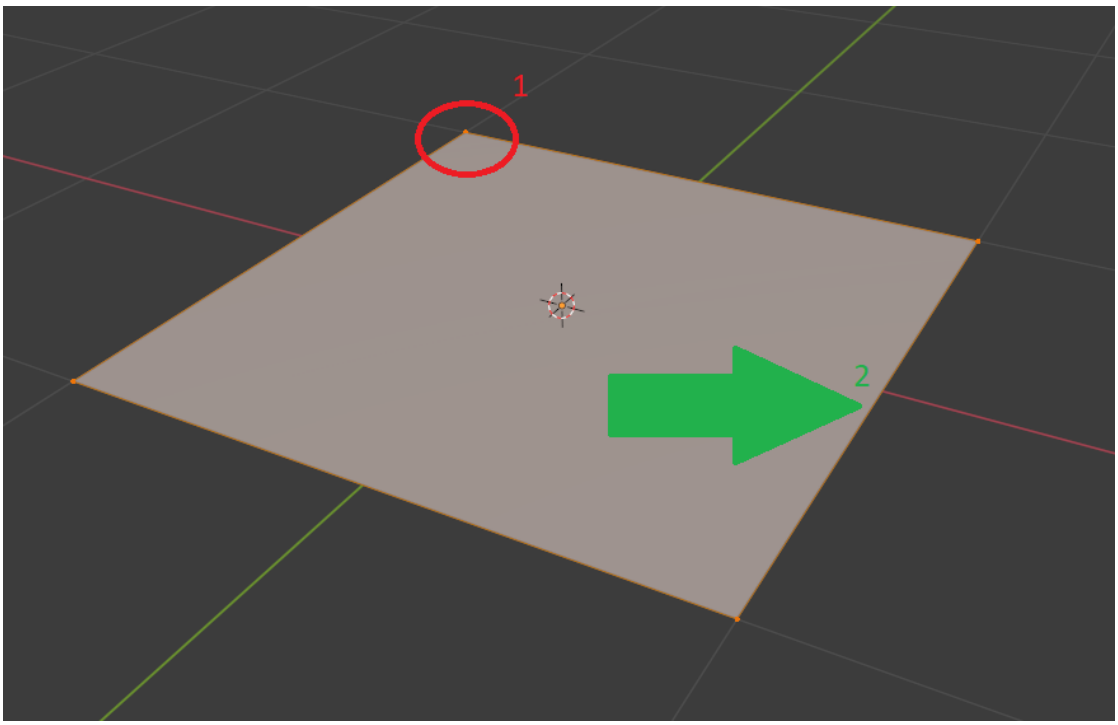
jokainen ilmamolekyyli maalataan valkeaksi ja kahvimukin molekyyli maalataan mustaksi, saadaan molekyylitason kuvaus kahvimukista. (Badler & Glassner N.d.)

Tällä ajatuskokeella on monia yhtymäkohtia tietokonegrafiikoissa käytettävien 3D-mallien kanssa. Se on käsitteellisesti selkeä, ja sillä on rajattu tarkkuus. Jotkin mallinnusmenetelmät ovat hyvin lähellä tätä lähestymistapaa: ne käyttävät pisteitä tilassa, tai pienii tilan hyvin pieniin osiin ja merkkaa osat joko tyhjiksi tai täysiksi. Toiset menetelmät yrittävät antaa mahdollisuuden abstraktimpaan kuvaukseen. (Badler & Glassner N.d.)

Erään määritelmän mukaan 3D-mallintaminen on prosessi, jossa luodaan kolmiulotteinen kuvaus mistä tahansa pinnasta tai esineestä, manipuloimalla polygoneja, reunoja ja verteksejä simuloidussa kolmiulotteisessa tilassa (Slick 2020).

Polygonit ovat geometrisiä kuvauksia, joita 3D-mallinnus ohjelmat käyttävät yleisimmin nykyään. Ne ovat selkeitä ja helposti muokattavissa monin eri työkaluin. Polygonit muodustuvat kolmesta tai useammasta kärjestä, joita sanotaan vertekseiksi (engl. yks. *vertex*, mon. *vertices*). Kahden verteksin välistä suoraa nimitetään reunaksi. Useimmat 3D-ohjelmat täydentävät reunojen muodostamat polygonin tasot tahkoiksi (engl. *face*). Kutakin komponenttia voidaan siirtää, kiertää ja uudelleenmitoittaa 3D-tilassa. Polygonin tahkot ovat nähtävissä renderöinnissä. Reunat ja verteksit ovat näkymättömiä renderöinnissä, mutta voidaan nähdä ja ne ovat muokattavissa ohjelmistossa. (Beane 2012.)

Kuviossa 1 on merkattu tason muodostavat elementit: punaisella ympyröitynä yksi vertekseistä, vihreä nuoli osoittaa yhden reunoista.



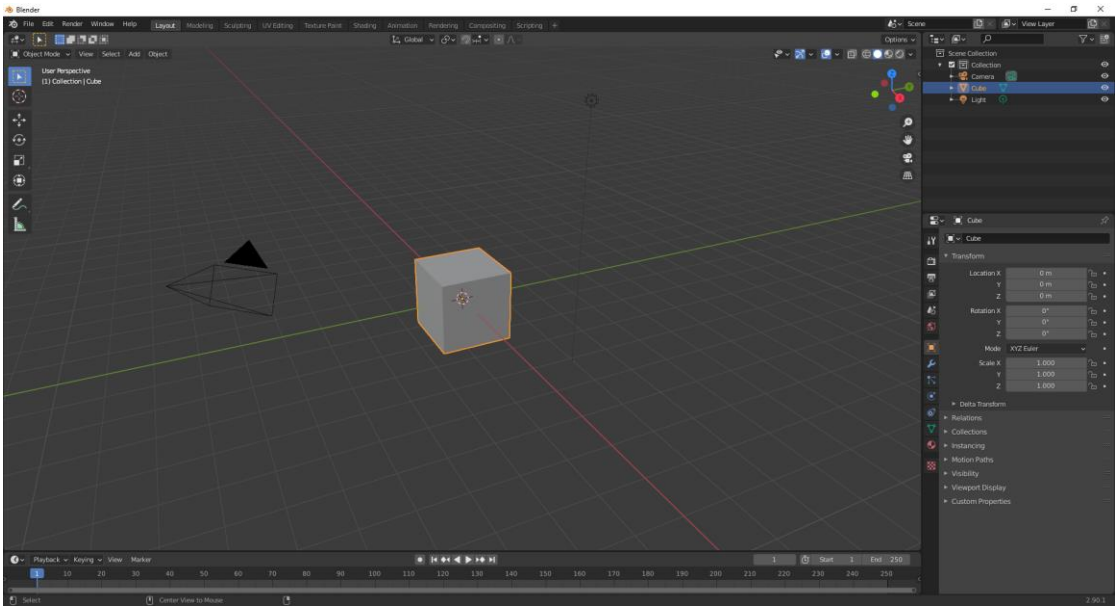
Kuvio 1. Taso Blenderissä

Yksinkertaisin polygoni on kolmesivuinen *tri*. Useiten käytetty polygoni on nelisivuinen *quad*. Jos polygoni on 5 tai useampi sivuinen kutsutaan sitä *n-goniksi.* Näitä vältetään, koska niiden muoto voi muuttua vastoin odotuksia, ja sen renderöinnissä voi tulla odottamattomia tuloksia. N-gonit voidaan muuttaa kolmi- tai nelisivuiseksi 3D-ohjelmiston työkaluilla, joten on yksinkertaisempaa olla käyttämättä n-goneja mallinnuksessa. (Beane 2012.)

3.3 Blender

Blender on ilmainen avoimen lähdekoodin 3D-ohjelmisto. Sitä voi käyttää monipuolisesti kolmiulotteisen tuotannon eri osa-alueilla esimerkiksi: mallinnus, animointi, renderöinti ja jopa videoeditointi. (About N.d.)

Kuviossa 2 on näkymä Blenderin käyttöliittymästä oletusasetuksilla.



Kuvio 2. Blenderin oletusnäkymä

Blender oli alun perin palkitun animaatioyhtiö NeoGeon sisäinen tuotantotyökalu, jonka perusti ja johti Ton Roosendaal. Kun kiinnostus Blenderiin kasvoi, Ton Roosendaal perusti uuden yhtiön, Not a Number (NaN), joka markkinoi ja möi Blenderiä. Vuonna 2002 NaN joutui sulkemaan ovensa. Tähän mennessä Blenderin ympärille oli kehittynyt vahva yhteisö, joka innokkaasti etsi tapaa estää Blenderin katoaminen. Heinäkuussa 2002 Ton Rosendaal perusti voittoa tavoittelemattoman organisaation Blender Foundation, joka yhteisön seitsemässä viikossa keräämällä n. 100 000 dollarin summalla hankki Blenderin lähdekoodin. Tämä julkaistiin GPL-lisenssillä myöhemmin samana vuonna. (Gumster 2011).

Nykyisin Blenderiä kehittää sadat myötävaikuttajat ympäri maailman. Näihin lukeutuu niin studioita kuin yksittäisiä ihmisiä, niin ammattilaisia kuin harrastelijoita. (Get involved N.d.)

Aloittelijalle löytyy lukuisia ilmaisia Blenderiin liittyviä videotutoriaaleja, tunnetuimpia lienee Andrew Pricen Blender Guru -YouTube kanava, ja erityisesti donitsin mallintamiseen keskittynyt videosarja. Kuviossa 3 donitsi, joka on mallinnettu seuraamalla kyseisiä videotutoriaaleja.

Kuvio 3. Blenderillä tuotettu kuva donitsista

4 Tekoäly, koneoppiminen ja syväoppiminen

4.1 Tekoäly

Tekoäly käsitteenä voi tarkoittaa hyvinkin eri asioita eri ihmisille. Lucci & Kopec (2013 luku 1.0.1), tarkastelee käsitettä *artificial intelligence* semantiikan kautta. Heidän mukaansa *artificial* tarkoittaa ihmisen tekemää, ja sillä on usein negatiivinen konnotaatio. Mutta keinotekoiset asiat ovat kuitenkin usein ylivertaisia luonnollisiin tai aitoihin verrattuna, he jatkavat. Esimerkkinä käytetään tekokukkaa (*artificial flower*) Tämä tekokukka ei tarvitse vettä, eikä valoa, joten se on näppärä kodin koriste-esine. Päällisin puolin se voi olla hyvinkin aidon näköinen.

Tekoälyn määrittely muuttuukin hankalammaksi tarkasteltaessa älykkyyttä. Lucci & Kopec (2013 luku 1.0.2), esittelevät älykkyyden eri ilmentymiä luonnossa, ja toteavat että älykkyys ei ole ihmisen yksinoikeus. He kannustavat pohtimaan voiko elottomat asiat kuten tietokoneet olla älykkäitä. Heidän mukaansa julistettu tavoite tekoälyn saavuttamiseksi on luoda ohjelmistoja ja/tai laitteistoja, jotka ilmentävät ajattelua,

joka on rinnastettavissa ihmisten ajatteluun, toisin sanoen esittää piirteitä, jotka yleensä yhdistetään ihmisiin.

Tekoälyntutkimus tieteenalana katsotaan alkaneen vuonna 1956, kun Dartmouth Summer Research Project -työpaja järjestettiin (Moor 2006). Alkuperäisessä ohjelmaesityksessä esitettiin kahden kuukauden, 10 hengen tekoälytutkimusta, joka pohjautuu konjektuuriin, että jokainen oppimisen näkökulma tai mikä tahansa muu älykkyyteen liittyvä ominaisuus voidaan periaatteessa kuvailla niin tarkasti, että voidaan tehdä kone sitä simuloimaan (McGarthy, Minsky, Rochester & Shannon 1956).

4.2 Koneoppiminen

4.2.1 Mitä on koneoppiminen?

Koneoppimisalgoritmi on hakuprosessi, joka on suunniteltu valitsemaan paras funktio joukosta mahdollisia funktioita, selittämään tietoaineiston ominaisuuksien suhteita. Koneoppimisprosessi sisältää kaksi askelta, kouluttaminen ja päättely. Kouluttamisvaiheessa algoritmi käsittelee tietoaineistoa ja valitsee funktion, joka parhaiten sopii aineiston yhteyksiin. Määritetty funktio muutetaan tietokoneohjelmaksi, ja tämä ohjelmaa kutsutaan malliksi. (Kelleher 2019.)

Koulutuksen päätyttyä mallia ei enää muokata. Seuraava vaihe koneoppimisessa on päättely. Tällöin mallia käytetään uusiin näytteisiin, joiden ulostuloarvoa ei tunneta, ja mallia käytetään arvon estimointiin. Koneoppiminen tarvitsee siis kolme elementtiä toimiakseen: joukon menneistä näytteistä, joukon funktioista, joista valita paras yhteensopivuus aineistoon sekä jonkinlaisen mittarin, jolla arvioida kunkin funktion suoriutumista (Kelleher 2019).

4.2.2 Ohjattu oppiminen

Ohjattu oppiminen on koneoppimisen yleisin tyyppi. Siinä jokainen tietoaineiston osa on luokiteltu odotettuun ulostuloarvoon. Algoritmi vertaa funktion tuottamia arvoja odotettuihin arvoihin, ja erotuksen tai virheen perusteella arvioi funktion suoriutumista etsiäkseen parhaiten suoriutuvan funktion. (Kelleher 2019.)

Tyypillinen ohjatun oppimisen tehtävä on luokittelu. Roskaposti-suodatin on tästä hyvä esimerkki: koulutus tapahtuu siten, että sähköpostien lisäksi on tieto siitä, ovatko sähköpostit roskapostia vai ei, ja mallin pitää oppia luokittelemaan uusia sähköposteja. Toinen tyypillinen tehtävä on numeerisen arvon ennustaminen, esimerkiksi auton hinnan ennustaminen, annettujen ominaisuuksien (ajokilometrit, ikä merkki jne.) perusteella. Malli koulutetaan aineistolla, joka sisältää sekä ominaisuuksien lisäksi hinnat. (Géron 2019, 8.)

4.2.3 Ohjaamaton oppiminen

Ohjaamatonta oppimista käytetään usein datan ryvästämiseen. Esimerkiksi tämän tyyppinen data-analyysi on hyödyllistä asiakassegmentoinnissa, jolloin yritys toivoo pystyvänsä segmentoimaan asiakaskuntansa ryhmiin, jotta voi kohdentaa kampanjoita ja/tai tuotekehitystä eri ryhmiin. Ohjaamattomassa oppimisessa tietoaineistossa ei ole kohdearvoja, vaan algoritmi yrittää yksilöidä funktiot, jotka määrittävät samankaltaiset näytteet ryppäisiin. (Kelleher 2019.)

Visualisointialgoritmit ovat hyviä esimerkkejä ohjaamattoman oppimisen algoritmeista: niille syötetään suuria määriä monimuotoista ja luokittelematonta dataa, ja ne tuottavat kaksi- tai kolmiulotteisen representaation datasta, joka on helposti piirrettävissä. (Géron 2019, 11.)

4.2.4 Vahvistusoppiminen

Vahvistusoppimista käytetään esimerkiksi robotin ohjaamisessa ja pelin pelaamisessa. Siinä toimijan, agentin, tulee oppia kuinka toimia ympäristössä tullakseen palkituksi. Agentin tavoite on siis oppia määrittämään toimiaan suhteessa sen hetkiseen havaintoon ympäristöstä ja omaan tilaansa. Esimerkiksi tulisiko robotin liikkua eteen vai taakse, tai tulisiko tietokoneohjelman liikuttaa sotilasta vai syödä kuningatar. (Kelleher 2019.)

Funktion ulostulo on agentin seuraava toiminta, ottaen huomioon nykyhetken olosuhteet. Koska on vaikea luoda tämänkaltaisia historiatietoja, agentti usein pääste-

tään ympäristöön, jossa se kokeilee eri toimintatapoja, ja päivittää toimintaansa saatujen palkintojen perusteella. Jos palkinto on positiivinen, vahvistetaan toimintamallia, ja jos palkinto on negatiivinen, toimintamallia heikennetään. (Kelleher 2019.)

4.3 Syväoppiminen

Syväoppimisen juuret ovat vahvasti tekoälyn ja tietokonenäön tutkimuksessa. Vuoden 1943 Warren McCulloch ja Walter Pitts, julkaisussaan esittelivät yksinkertaistetun mallin aivojen neuronista, joka nimettiin McCulloch-Pitts neuroniksi (*MCP-neuron*). Malli kuvasi hermosolun yksinkertaisena loogisena porttina: signaaleja saapuu tuojahaarakkeisiin, niitä yhdistellään soomassa, ja jos tietty kynnys ylittyy, lähtösignaali tuotetaan viejähaarakkeessa. (Raschka & Mirjalili 2019, s. 20.)

Tähän malliin perustuen Frank Rosenblatt kehitti *Perceptronin*, joka voidaan nähdä neuroverkkona, joka koostuu yhdestä neuronista. Ensimmäinen implementaatio oli ohjelmisto IBM 704 -järjestelmässä, mutta Rosenblattin aikomuksena oli, että perseptroni olisi fyysinen laite. Myöhemmin laite ”Mark 1 perceptron” tehtiin. Siihen oli kytketty kamera, joka tuotti 400-pikselin kuvia, jotka syötettiin 400 valokennopariin, jotka kytkettiin neuroneihin. Painokertoimina toimi potentiometrit, joita säädettiin sähkömoottoreilla. (Kelleher 2019.)

Rosenblatt määritti säännön perseptronin kouluttamiseen, painokertoimien päivittämiseksi jokaisen prosessoidun näytteen jälkeen:

1. Jos mallin ulostulo vastaa näytteen määritettyä ulostuloa, älä tee mitään painokertoimille
2. Jos mallin ulostulo on liian matala verrattuna näytteen ulostuloarvoon verrattuna, nosta painokertoimia, niille syötteille, joilla on positiivinen arvo, ja laske painokerrointa, jos syöte on negatiiviinen
3. Jos mallin ulostulo on liian korkea verrattuna näytteen ulostuloarvoon, laske painokertoimia, niille syötteille, joilla on positiivinen arvo, ja nosta painokertoimia, jos niiden syötteiden arvo on negatiivinen. (Kelleher 2019.)

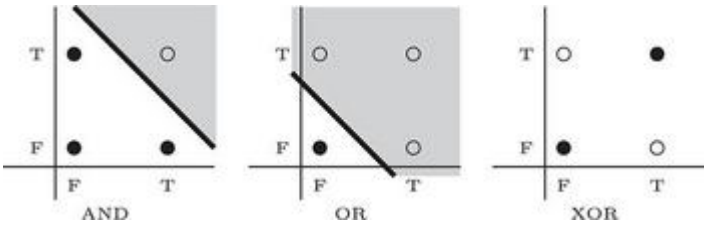
Saman voi esittää kaavamuodossa (ks. kaava 1).

$$w_i^{t+1} = w_i^t + (\eta \times (y^t - \hat{y}^t) \times x_i^t) \qquad (1)$$

Missä w_i^{t+1} on *i* painokertoimen arvo, kun verkon painokertoimet on päivitetty, w_i^t on *i* painokertoimen arvo näytteelle *t.* η on oppimistahti (engl. *learning rate*), y^t on odotettu ulostulo näytteelle *t*, $\hat{y}^t$ on perseptronin ulostulo näytteellä *t*, ja x_i^t on syötteen osa, jonka painokerroin w_i^t on. Jos oppimistahti on liian pieni, oppimisprosessilta kestää kauan, ennen kuin se suppenee, ja jos se on liian suuri verkon painokertoimet saattavat heitellä liikaa, eikä koulutus suppene lainkaan. (Kelleher 2019.)

Rosenblatt osoitti, että jos määrätyt painokertoimet ovat olemassa, joilla perseptroni kykenee luokittelemaan kaikki koulutusnäytteet oikein, koulutusalgoritmi lopulta suppenee noihin arvoihin (*perceptron convergence theorem*). Samoihin aikoihin, kun Rosenblatt kehitti perseptronia, Bernard Widrow ja Marcia Hoff kehittivät samankaltaista mallia nimeltä ADALINE (*adaptive linear neuron*), ja sen myötä oppimissääntöä LMS (*least mean square*). ADALINE oli samankaltainen kuin perseptroni, mutta se ei hyödyntänyt kynnysfunktiota. Sen sijaan ADALINE-verkon ulostulo on vain painotettu syötteiden summa. (Kelleher 2019.)

Rosenblattin, Widrown ja Hoffin, ja muiden, onnistumiset neuroverkkomallien automaattisessa oppimisessa, eri kuvioiden oppimisessa, synnytti innostusta tekoälyn ja neuroverkkojen tutkimuksessa. Marvin Minsky ja Seymour Papert kuitenkin onnistuivat latistamaan odotuksia, julkaisemalla 1969 tutkimuksen, jossa he osoittivat, että yhden kerroksen perseptroni ei kykene oppimaan epälineaarisia funktioita, kuten XOR-funktio. Kuviossa 4 esitetään kuinka AND ja OR funktiot ovat lineaarisesti erotettavia, kun taas XOR ei ole. (Kelleher 2019.)



Kuvio 4. Loogisia funktioita (Kelleher 2019).

Tuon julkaisun aikoihin tiedettiin, että oli mahdollista tehdä neuroverkkoja, jotka kykenivät oppimaan myös epälineaarisesti erotettavia funktioita. Tämä onnistui laajentamalla verkon kerrosten määrää. Oli kuitenkin epäselvää, kuinka kouluttaa näitä neuroverkkoja, jotka koostuivat useammasta kerroksesta. (Kelleher 2019.)

Vasta 1980-luvulla alettiin tarkastella neuroverkkojen potentiaali uudestaan. Kaksi edistysaskelta uudelleen sytytti innostuksen alaan: Hopfieldin verkot ja vastavirta-algoritmi. Hopfieldin verkko on John Hopfieldin 1982 julkaisun tulos, jossa kuvaillaan verkko, joka kykeni toimimaan assosiatiivisena muistina. Koulutuksen aikana assosiatiivinen muisti opettelee syötteiden yhteydet. Koulutettu malli kykenee palauttamaan oikean arvon, kun sille syötetään korruptoitunut syöte. (Kelleher 2019.)

Syväoppimisen kannalta todennäköisesti tärkein algoritmi on vastavirta-algoritmi (engl. *back-propagation*). Sen avulla kyetään kouluttamaan neuroverkkoa, jossa on piilotettuja kerroksia. Algoritmi alkaa määrittämällä satunnaiset painoarvot jokaiselle verkon kytkökselle. Sen jälkeen algoritmi iteratiivisesti päivittää painoarvoja esittämällä verkolle koulutusmateriaalia ja päivittää painoarvoja, kunnes verkko toimii toivotusti. Algoritmin ydin toimii kahdessa vaiheessa: ensimmäisessä vaiheessa, engl. *forward pass*, annetaan syötteen virrata verkon läpi, kunnes ulostulo generoidaan.

Toisessa vaiheessa, englanniksi *backward pass*, aloitetaan ulostulokerrokselta ja työskennellään verkon vastasuuntaan, kunnes syötekerros saavutetaan. Tässä vaiheessa ensin lasketaan virhe jokaiselle neuronille ulostulokerroksella. Tämän virheen avulla päivitetään painoarvot, jotka vaikuttavat ulostulokerroksen neuroneihin. Sen jälkeen virhe jaetaan taaksepäin piilotetuille neuroneille, suhteessa painoarvoihin, jotka liittyvät piilotetun ja ulostuloneuronin ´väliseen kytkökseen. Kun jakaminen on tapahtunut piilotetun neuronin kaikkien eteenpäin menevien kytköksien virhe, summataan ja tällä arvolla päivitetään neuronin painoarvoa verkossa. Tämä virheen jakaminen ja arvojen päivitys jatkuu verkkoa pitkin, kunnes syötekerros saavutetaan ja kaikkia painoarvot on päivitetty. (Kelleher 2019.)

Vastavirta-algoritmin mahdollistava innovaatio oli aktivaatiofunktion muuttaminen neuroneissa. Algoritmi tarvitsee differentioituvan funktion toimiakseen, kuten logistinen funktio tai hyperbolinen funktio. Vastavirta-algoritmin käyttämisessä neuroverkon kouluttamisessa oli kuitenkin eräs ongelma. Se toimi hyvin, kun piilotettuja kerroksia oli vähän, mutta kun verkkoa syvennettiin koulutus joko vei kauan aikaa tai ei löytänyt sopivia painoarvoja lainkaan. 1991 Sepp Hochreiter tunnisti ongelman syyksi tavan, jolla algoritmi jakoi virhettä taaksepäin verkossa. Pohjimmiltaan kyseessä on differentiaalilaskennan ketjusääntö, joka sisältää termien kertomista. Vastavirta-algoritmin virhe neuronien välillä saattaa olla alle 1, ja kun algoritmi jatkaa verkkoa eteenpäin näiden arvojen tulo jatkaa verkkoa eteenpäin ja virheen arvo lähestyy nollaa. Tämä tekee sen, että verkon alkupään painoarvoja muutetaan hyvin vähän tai ei lainkaan. Tästä huolimatta vastavirta-algoritmi avasi mahdollisuuden tehdä uusia syvempiä neuroverkkoja. (Kelleher 2019.)

5 Tietokonenäkö

5.1 Tietokonenäkö

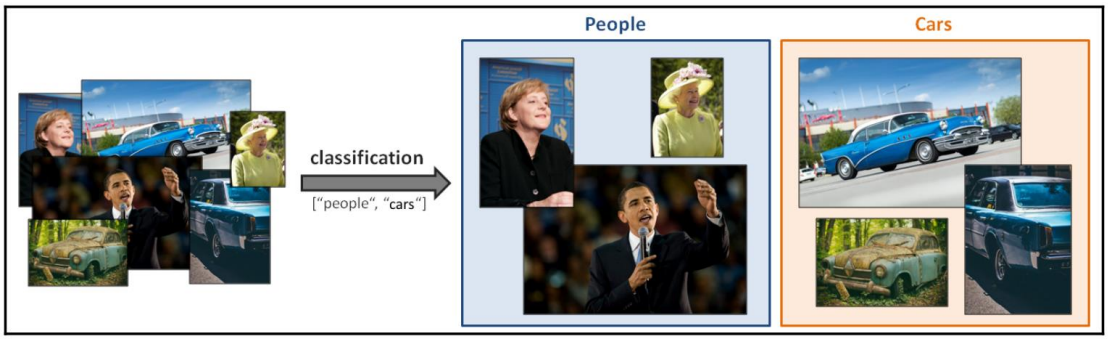
Tietokonenäön määrittely voi olla hankalaa, koska se sijoittuu usean tieteenalan leikkauspisteeseen, mutta kiteytettynä sen voidaan sanoa olevan automatisoitua tiedon johtamista digitaalisista kuvista (Planche, Andres 2019, 10).

SAS-yhtiön verkkosivuilla sanotaan, että tietokonenäkö on tekoälyn aihealue, joka kouluttaa tietokoneita tulkitsemaan ja ymmärtämään visuaalista maailmaa (Computer Vision -What it is and why it matters N.d).

5.2 Tietokonenäön tehtäviä

5.2.1 Esineen luokittelu (Object classification)

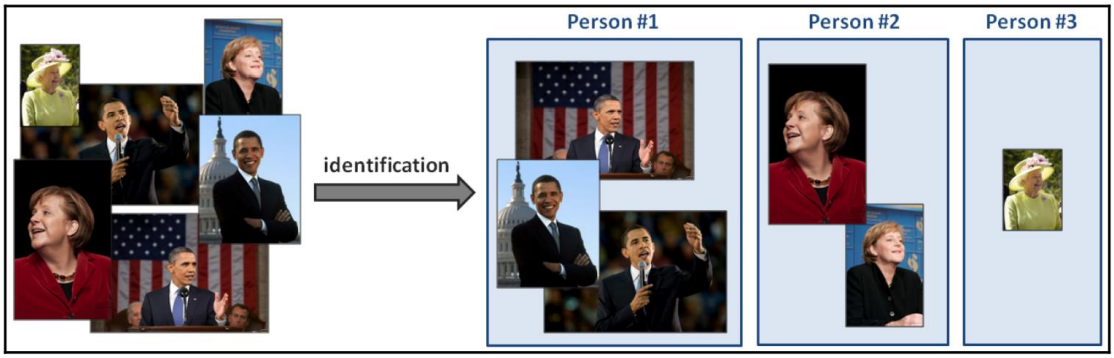
Esineen tai kuvan luokittelu on tehtävä, jossa kuvaan liitetään nimike ennalta määrätystä joukosta nimikkeitä. Esineen luokittelu on ohjatun oppimisen ongelma: määritetään joukko kohdeluokkia ja koulutetaan malli tunnistamaan niitä käyttäen nimikoituja kuvia. Kuviossa 5 luokitellaan kuvat joko ihmisiin tai autoihin. (Planche, Andres 2019, 11; ML Practicum: Image Classification 2020.)



Kuvio 5. Luokittelua (Planche, Andres 2019, 11.)

5.2.2 Esineen tunnistaminen (Object identification)

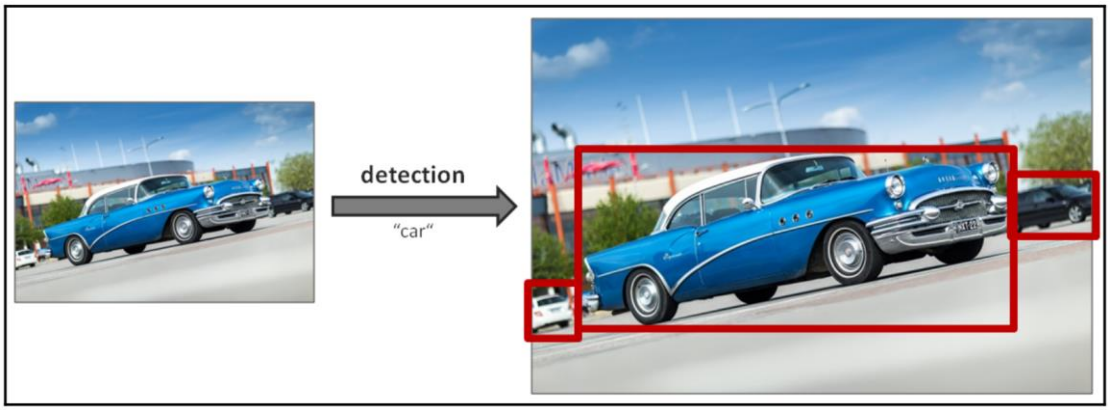
Kun esineen luokittelun metodit keskittyvät määräämään nimikkeitä ennalta määrätystä joukosta, esineen tunnistamisen metodit oppivat tunnistamaan tiettyjä luokan esiintymiä. Esimerkiksi niitä voidaan käyttää tunnistamaan henkilöitä kuvista (ks. kuvio 6). (Planche, Andres 2019, 12.)



Kuvio 6. Henkilöiden tunnistamista (Planche, Andres 2019, 12.)

5.2.3 Esineen havaitseminen ja lokalisaatio (Object detection and localization)

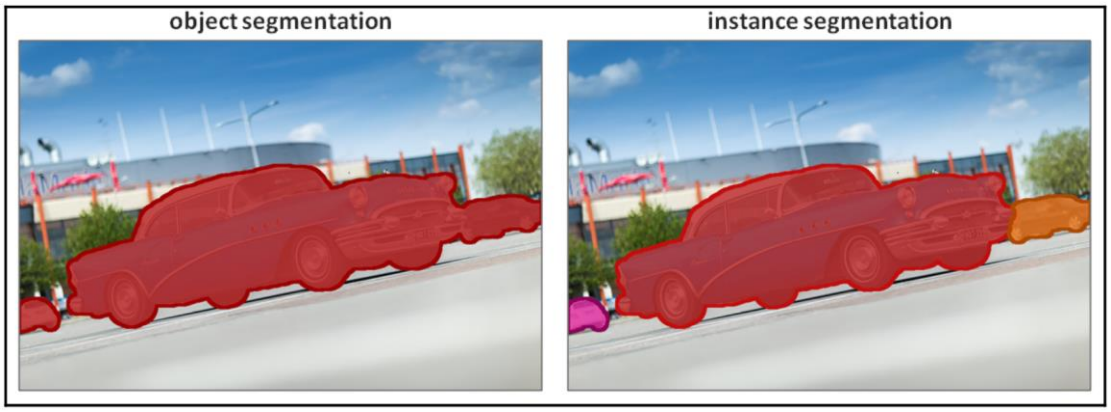
Eräs tietokonenäön tehtävä on havaita tiettyjä osia kuvissa. Sitä voidaan käyttää esimerkiksi syöpäsolujen havaitsemiseen lääketieteessä. Havaitseminen on usein alustava askel ennen jatkolaskentaa, sillä saadaan pienempiä osioita kuvasta erikseen analysoitavaksi, esimerkiksi kasvojen erottaminen kuvasta tunnistamista varten. Kuviossa 7 syötteenä on kuva ja ulostulona kuvasta havaituille autoille on piirretty rajasulaatikot. (Planche, Andres 2019, 13.)



Kuvio 7. Esineen havaitsemista (Planche, Andres 2019, 13.)

5.2.4 Esineen ja ilmentymän segmentointi (Object and instance segmentation)

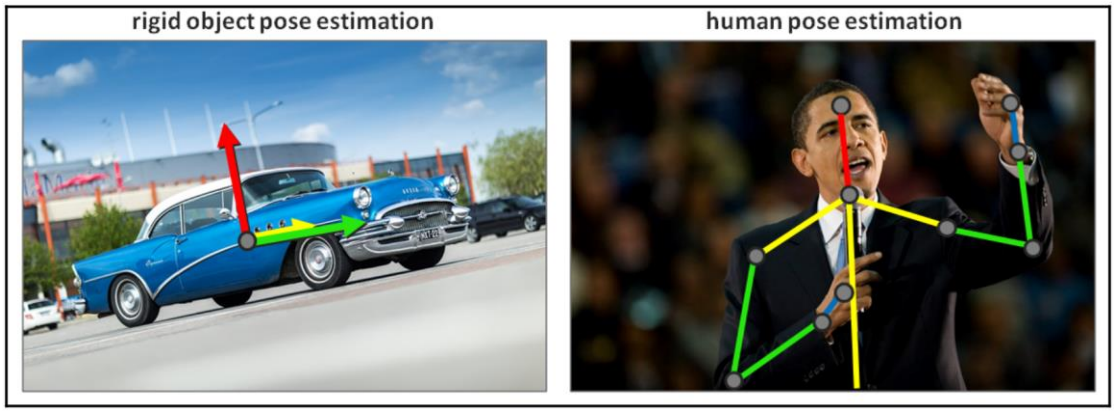
Segmentointi voidaan katsoa olevan edistyneempi havaitsemisen tyyppi. Yksinkertaisten rajauslaatikoiden sijaan segmentointi tuottaa maskin, joka nimiöi kaikki tietyn luokan esineet tai tietyn luokan ilmentymät kuvasta. (Planche, Andres 2019, s. 14.)



Kuvio 8. Esineen ja ilmentymän segmentointi (Planche, Andres 2019, s. 14.)

5.2.5 Asennon estimointi (Pose estimation)

Asennon estimointi voi tarkoittaa eri asioita eri kohteille. Kiinteille esineille se yleensä tarkoittaa esineen asentoa ja orientaatiota suhteessa kameraan. Tämä on erityisen hyödyllistä roboteille, jotta ne voivat vuorovaikuttaa ympäristön kanssa. Ei kiinteille esineille asennon estimointi tarkoittaa niiden aliosien sijaintia suhteessa toisiinsa. Esimerkiksi tyypillinen ihmiseen kohdennettu sovellus tunnistaa ihmisen asennon kuvasta (istuu, seisoo, juoksee jne.), kuviossa 9 esimerkit niin kiinteästä esineestä kuin ihmisen asennosta. (Planche, Andres 2019, 15.)



Kuvio 9. Asennon estimointi (Planche, Andres 2019, 15.)

6 Konvoluutioneuroverkko

David H. Hubelin ja Torsten Wieselin tutkimus näköaivokuoren toiminnasta kissoja tutkimalla vuonna 1959 toi ymmärrystä siihen kuinka aivojemme näköjärjestelmän tiedonkäsittely tapahtuu. Vuonna 1989 LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Hubbard, W., Jackel, L. D., ja Henderson, D. esittelivät uuden neuroverkkoarkkitehtuurin, käsin kirjoitettujen numeroiden luokittelemiseksi kuvista julkaisussa ”*Handwritten Digit Recognition with a Back-Propagation Network*”. Tuo neuroverkkoarkkitehtuuri pohjautui näköaivokuoren toimintatapaan ja oppi tunnistamaan käsin kirjoitettuja numeroita suoraan kuvista hyödyntämällä koulutuksessa vastavirta-algoritmia. (LeCun, Boser, Denker, Howard, Hubbard, Jackel, & Henderson 1989; Raschka, Mirjalili 2019, 518.)

Onnistunut piirteiden (engl. *feature*) valinta on avain minkä tahansa koneoppimisalgoritmin suoriutumiseen, ja perinteiset koneoppimismallit pohjautuvat siihen, että syötepiirteet tulevat esimerkiksi asiantuntijalta. Osa neuroverkkotyypeistä, kuten konvoluutioneuroverkko kykenee oppimaan automaattisesti noita piirteitä, ilman esikäsittelyä. Tämän takia konvoluutioneuroverkkoja käytetään usein piirteiden erottamiseen datasta. Alkupään kerrokset erottavat piirteet ja myöhemmät kerrokset käyttävät noita piirteitä esimerkiksi luokitteluun. (Raschka, Mirjalili 2019, 518–519.)

Useampi kerroksiset neuroverkot, erityisesti syvät konvoluutioneuroverkot, rakentavat niin sanotun piirrehierarkian yhdistelemällä matalantason piirteitä kerroksittain korkeamman tason piirteiden muodostamiseksi. Esimerkiksi kuvissa matalantason piirteet ovat reunoja ja möykkyjä, jotka alkupään tasot erottavat, ja joita yhdistämällä saadaan korkeamman tason piirteitä esimerkiksi esineiden yleismuotoja. (Raschka, Mirjalili 2019, 519.)

Konvoluutioneuroverkoilla pyrittiin ratkaisemaan kaksi perustavaa ongelmaa, jotka esiintyvät, kun käytössä oli perinteinen täysin kytketty neuroverkko (engl. *fully connected neural network*): parametrien räjähdysmäinen kasvu ja tilallisen päättelyn puute. Digitaaliset kuvat koostuvat suurista joukoista arvoja, jotka tyypillisesti ilmoitetaan muodossa *H*W*D*, jotka tarkoittavat seuraavia: *H* on kuvan pikselin määrä korkeus suunnassa, *W* pikselien määrä leveys suunnassa, sekä *D* kuvan kanavien määrä (1 mustavalkokuvissa ja 3 tyypillisessä RGB-koodatussa värikuvassa).

Tämä tarkoittaa, että jo yksinkertainen yksikanavainen 28*28*1, kuva sisältää 784 arvoa, ja jos neuroverkon kerros koostuu esimerkiksi 64 neuronista, muodostuu 50 176 parametriä, jotka optimoida. Parametrien lukumäärä kasvaa entisestään, jos kuvat ovat suurempia värikuvia, tai neuroverkko muodostuu useammasta kerroksesta. Tämän lisäksi, kun neuroverkko on täysin kytketty, neuroni saa arvoja edellisen kerroksen kaikilta neuroneilta, ja se ei voi sisältää tilaan liittyvää tietoa, vaan esimerkiksi kuvien pikselien sijaintitieto toisiinsa nähden sekoittuu kerrosten välillä. (Planche, Andres 2019, 76).

Konvoluutioneuroverkot tarjoavat ratkaisun näihin ongelmiin. Konvoluutioneuroverkot kykenevät käsittelemään moniulotteista dataa. Tämän lisäksi kukin neuroneista ei olekaan kytketty kaikkiin edellisen kerroksen neuroneihin, vaan ainoastaan osaan vierekkäisistä osioista. Tämä ei ainoastaan hillitse parametrien määrää vaan myös säilyttää kuvan piirteiden sijaintitiedon. (Planche, Andres 2019, 77–78).

Konvoluutioneuroverkot saavat nimensä konvoluutiokerroksista, jotka ovat kyseisen neuroverkkoarkkitehtuurin perusta. Konvoluutiokerroksen, joka ottaa kuvan syötteenä, kukin neuroni kytketään vain osaan kuvan pikseleistä, ja vastaavasti seuraavan

tason konvoluutiokerroksen neuronit kytketään vain pieneen nelikulmioon ensimmäisen kerroksen neuroneista. Tämä arkkitehtuuri mahdollistaa neuroverkon keskittyä pieniin matalantason piirteisiin ensimmäisellä piilokerroksella, joista koostetaan korkeamman tason piirteitä seuraavalla tasolla ja niin edelleen. (Géron 2019, 448.)

7 TensorFow ja Keras

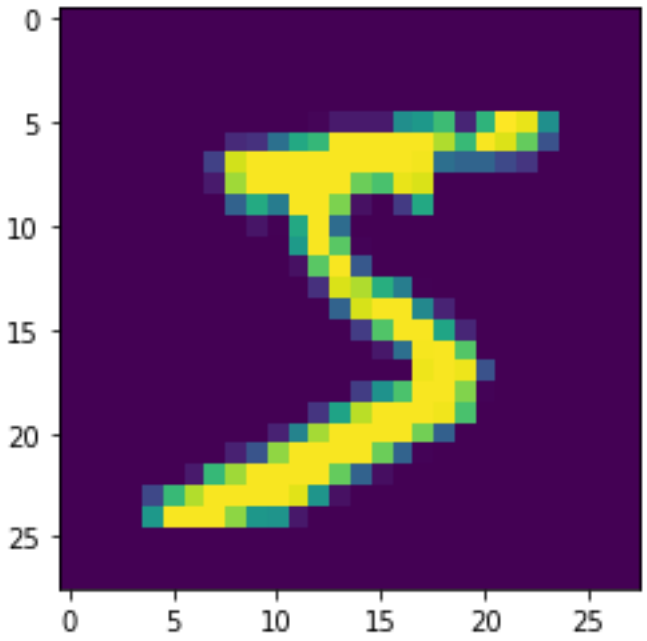
TensorFlow on avoimen lähdekoodin koneoppimisalusta. Se muodostuu kattavasta ja joustavasta ekosysteemistä työkaluja, kirjastoja ja yhteisön tuottamia aineistoja. Alun perin se kehitettiin Googlella, jotta tutkijat ja kehittäjät voisivat suorittaa koneoppimistutkimusta. TensorFlow:n lähdekoodi tehtiin julkiseksi marraskuussa 2015 (Tensorflow 2020; Planche, Andres 2019, 50).

Keras on syväoppimisrajapinta, joka toimii Tensorflown päällä. Se on kirjoitettu Python-ohjelmointikielellä, ja se on suunniteltu erityisesti huomioiden nopeat kokeillut. (About 2020.)

8 Konvoluutioneuroverkko esimerkki

8.1 Datasetti

Tarkastellaan konvoluutioneuroverkon toimintaa esimerkin avulla, esimerkki noudattelee Keraksen sivuilta löytyvää esimerkkiä. Tavoitteena on luoda malli, joka tunnistaa käsin kirjoitettuja numeroita. Aineistona käytetään MNIST-datasettiä, joka koostuu 60.000 koulutuskuvasta ja 10.000 testikuvasta käsin kirjoitettuja numeroita nollasta yhdeksään. Kuvien resoluutio on 28*28 pikseliä, ja ne ovat harmaasävykuvia. Kuviossa 10 esimerkki kuvasta, kun värikarttana on Viridis. Datasetin voi näppärästi ladata Keraksen kautta, ja se on jaettu valmiiksi koulutus- ja testikuviin.



Kuvio 10. Esimerkki MNIST-sarjan kuvasta

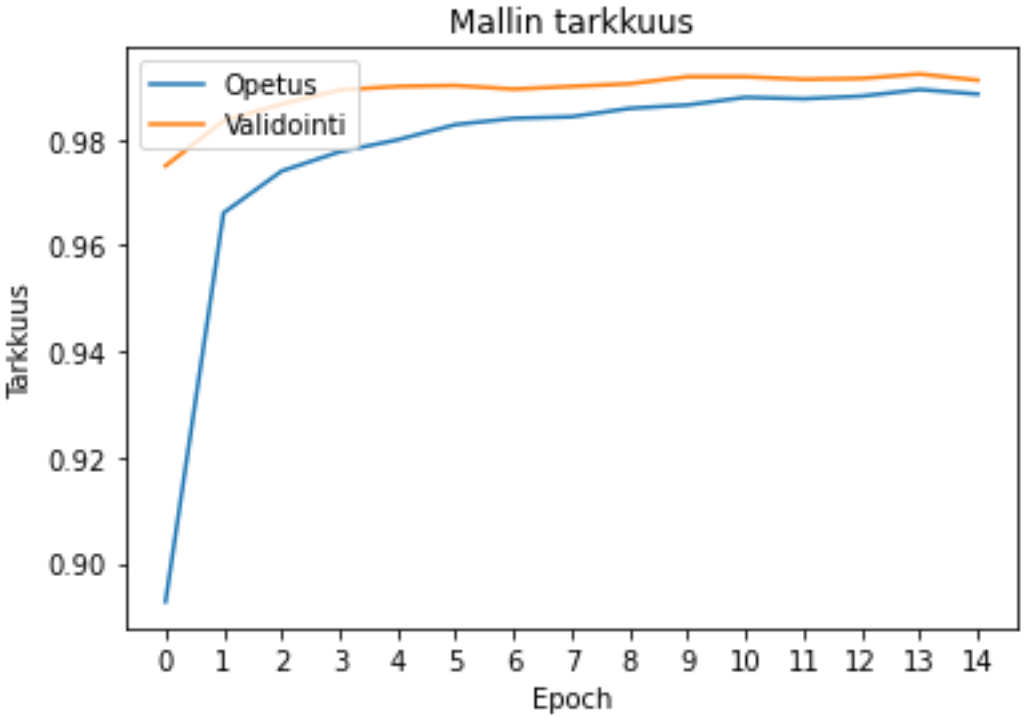
8.2 Neuroverkko ja sen koulutus

Mallia varten luotiin yksinkertainen konvoluutioneuroverkko. Neuroverkon rakenne on kuviossa 10. Neuroverkkoa koulutettiin 15 epochia, ja kuvia syötettiin neuroverkolle kerrallaan 128 (engl. *batch size*). Optimointifunktiona käytettiin Adamia, ja kuvista kymmenys erotettiin validointia varten.

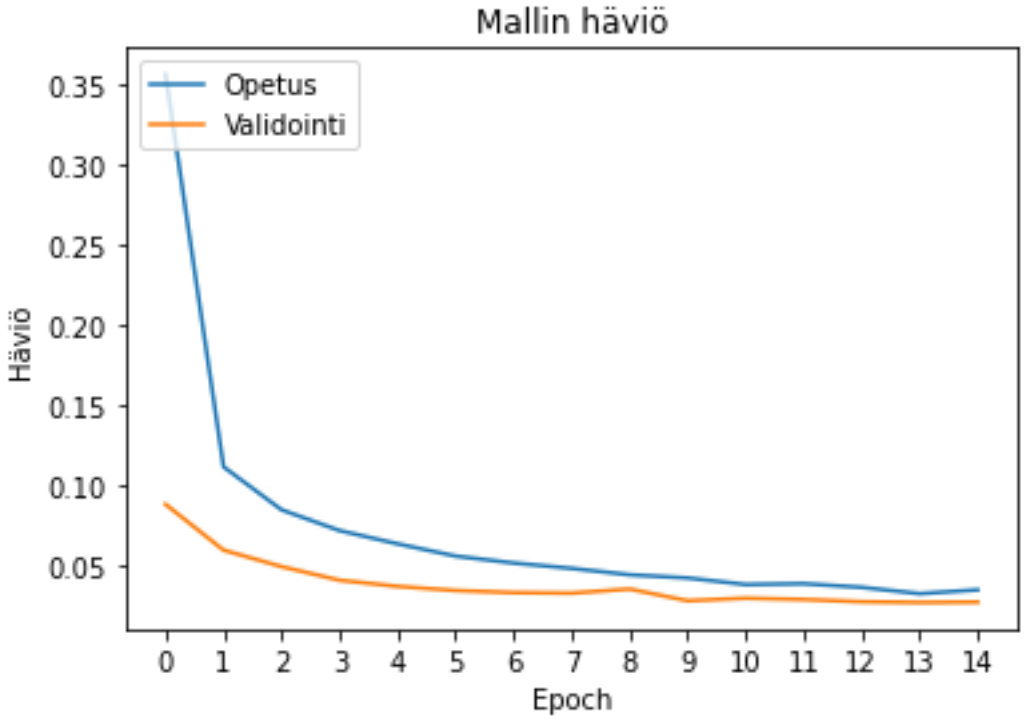
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
activation (Conv2D)          (None, 26, 26, 32)        320
_________________________________________________________________
maxpool (MaxPooling2D)       (None, 13, 13, 32)        0
_________________________________________________________________
activation2 (Conv2D)         (None, 11, 11, 64)        18496
_________________________________________________________________
maxpool2 (MaxPooling2D)      (None, 5, 5, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Kuvio 11. Konvoluutioneuroverkko

Kuviossa 11 on mallin tarkkuus koulutuksen aikana ja kuviossa 12 mallin häviö. Malli oppii siis tunnistamaan jo ensimmäisen epochin aikana tunnistamaan testikuvia yli 90 % tarkkuudella.

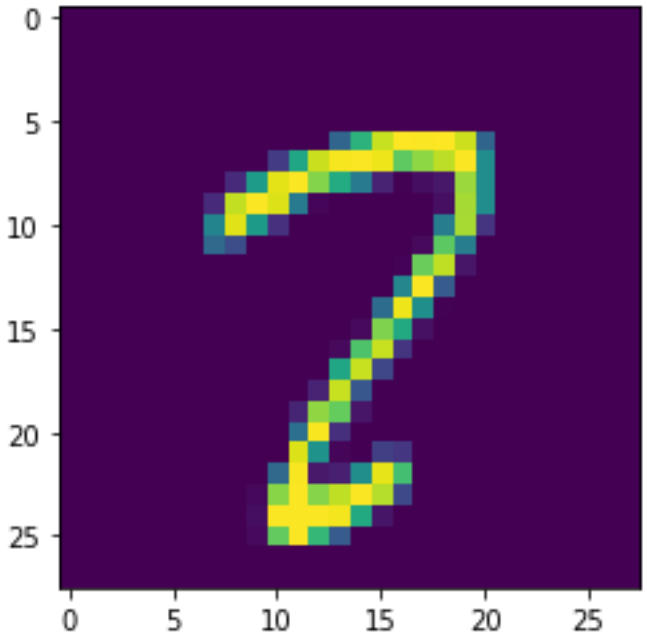


Kuvio 12. Mallin tarkkuus



Kuvio 13. Mallin häviö

Seuraavaksi mallin selviytymistä koeteltiin 10.000 testikuvalla, joita ei käytetty koulutuksessa. Mallin tarkkuus oli tällöin 99,24 % eli 10.000 kuvasta väärin oli 76. Kuviossa 14 näkyvän numeron malli tunnisti virheellisesti numeroksi 7, n. 53,1 % varmuudella.

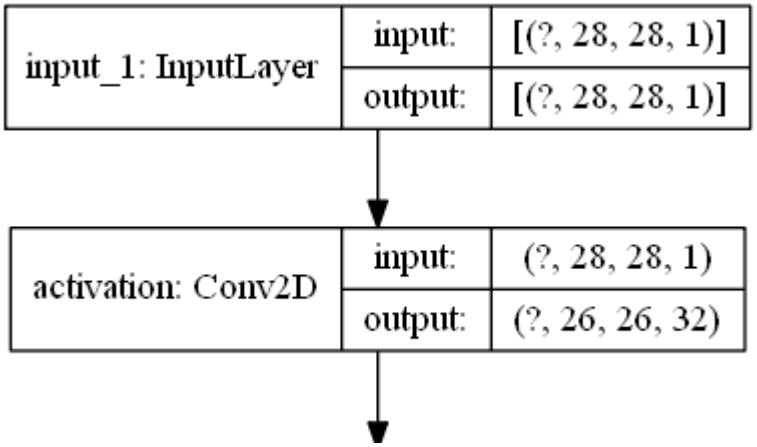


Kuvio 14. Esimerkki väärin tunnistetusta numerosta

8.3 Neuroverkon toiminta

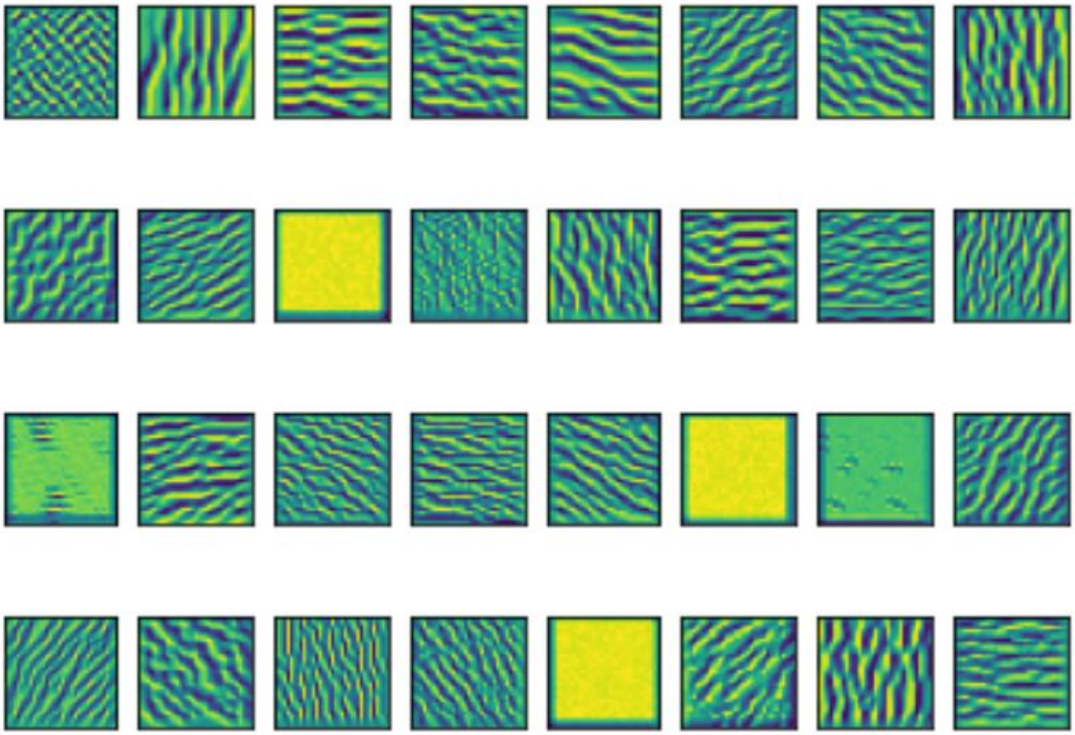
Konvoluutioneuroverkon keskeinen kerros on siis konvoluutiokerros, joka suorittaa syötteille konvoluutiota. Konvoluutiossa syötteeseen kohdennetaan suodatinta (engl. *filter*), joka tuottaa aktivaation. Kun samaa suodatinta kohdennetaan toistuvasti, tuloksena on piirrekartta (engl. *feature map*), joka osoittaa havaittujen piirteiden sijainteja ja voimakkuuksia syötteissä, esimerkiksi kuvissa. Konvoluutioneuroverkkojen innovaatio on kyky oppia automaattisesti suuria määriä suodattimia rinnakkain, tiettyyn datasettiin kohdennettuna ja rajattuna tiettyyn ennustavaan mallinnusongelmaan, esimerkiksi kuvan luokitteluun. Tuloksena on hyvin erikoistuneita piirteitä, joita voidaan havaita missä tahansa syötekuvassa. (Bronwlee 2019.)

Kuviossa 15 on käytetyn konvoluutioneuroverkon ensimmäiset kerrokset. Neuroverkko siis hyväksyy syötteenä 1-sävy kuvia, jotka ovat 28*28 pikseliä. Ensimmäisen konvoluutiokerroksen suodattimen eli kernelin koko on 3*3, ja koska ei käytetä minkäänlaista täytetoimintoa, se tuottaa 26*26 pikselin kokoisen piirrekartan, joka muodostuu 32 kerroksesta.



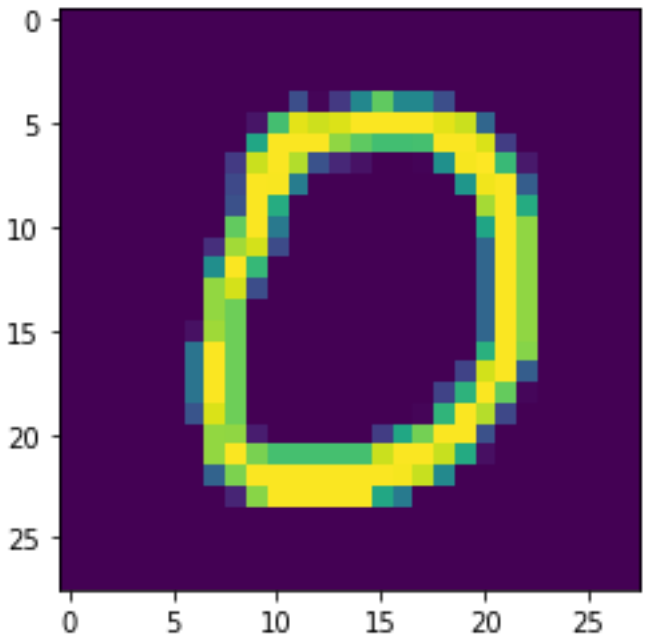
Kuvio 15. Ensimmäiset kerrokset

Kerneliä siis periaatteessa liu'utetaan alkuperäisen syötteen päällä, siten että tarkastelun alla on pieni osa kerrallaan. Otetaan esimerkiksi 9*9 pikselin suuruinen kuva, jonka ylitse liu'utetaan 3*3 suuruista kerneliä askelluksella yksi pikseli, saadaan lopputuloksena 3*3 pikselin piirrekartta. Kullekin kernelin askellukselle lasketaan ristikorrelaatio. Kuviossa 16 on ensimmäisen kerroksen konvoluution tuottamat piirrekartat.

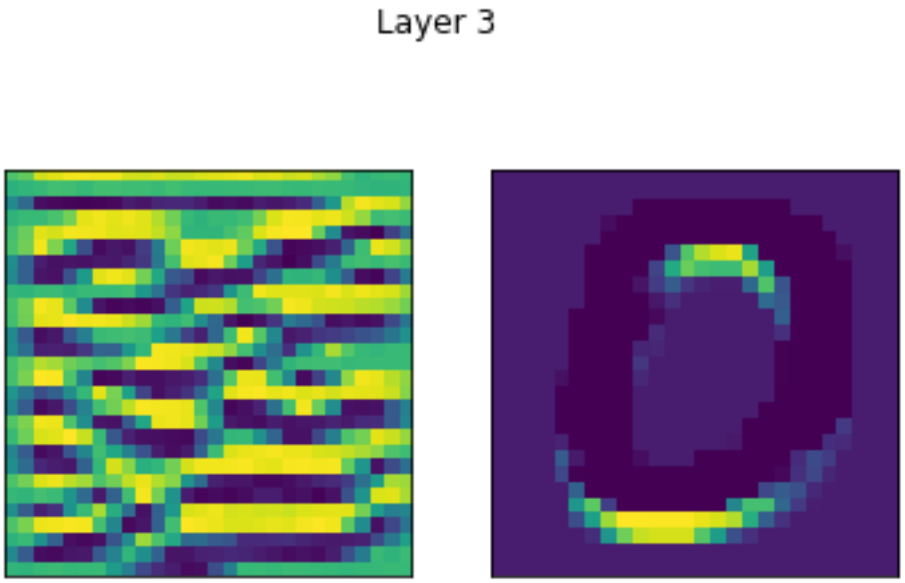


Kuvio 16. Suodattimet

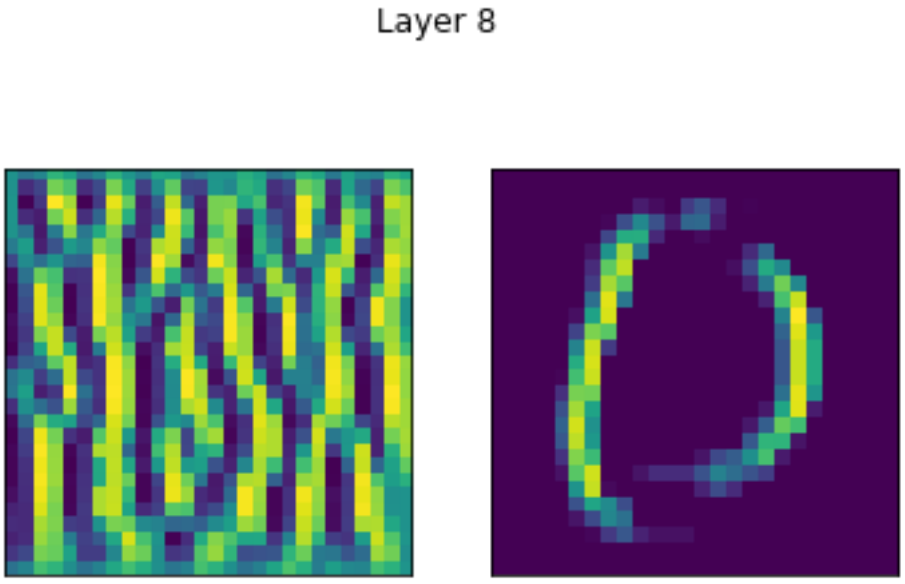
Käytetään syötteenä numeroa 0 esittävää kuvaa (ks. kuvio 17) ja tarkastellaan joidenkin kiinnostavimpien kerroksien tuottamaa aktivaatiota. Kuviossa 18 aktivoituu viivan alareunat ja kuviossa 19 aktivoituu pystyviivat.



Kuvio 17. Numero 0

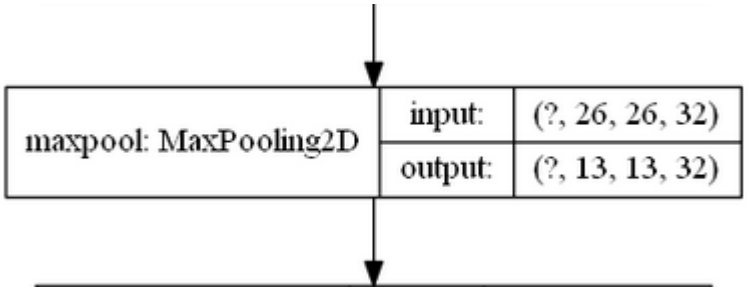


Kuvio 18. Kerros 3:n suodatin ja aktivaatio



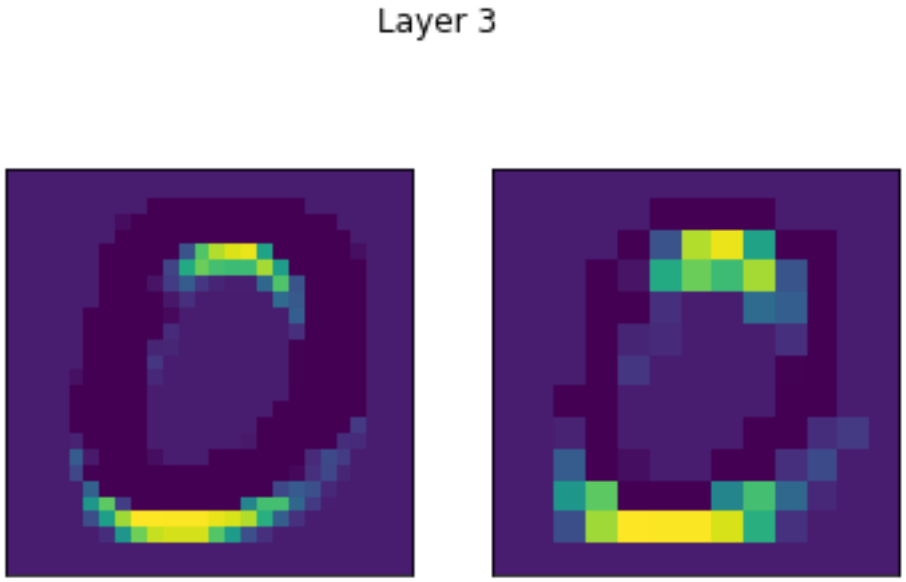
Kuvio 19. Kerros 8 ja aktivaatio

Konvoluutiokerroksen tuottamat piirrekartat viedään MaxPooling2d-kerrokselle (ks. kuvio 20), joka pyrkii korostamaan piirteitä, ja pienentää resoluutiota ja näin ollen myös laskettavien yhteyksien lukumäärää. Tässä neuroverkossa käytettiin 2*2 matriisia. Piirrekartan tasojen yli siis liu'utetaan 2*2 matriisia, joka poimii neljästä kentässä olevasta luvusta suurimman aktivaation. Piirrekartan resoluutio puolittuu.

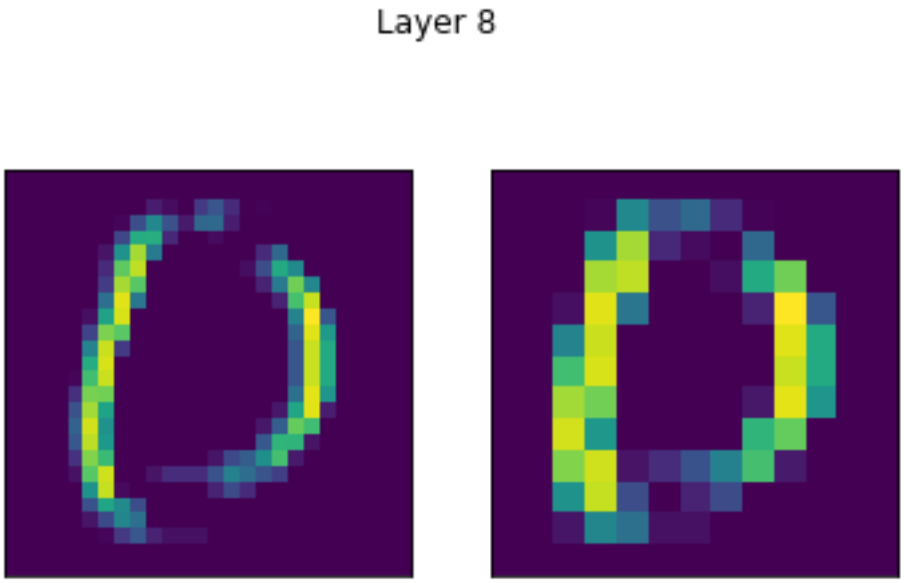


Kuvio 20. MaxPooling kerros

Kuvioissa 21 ja 22 havainnollistetaan MaxPooling2d toimintoa. Vasemmalla on syöte ja oikealla on yhdisteen ulostulo.

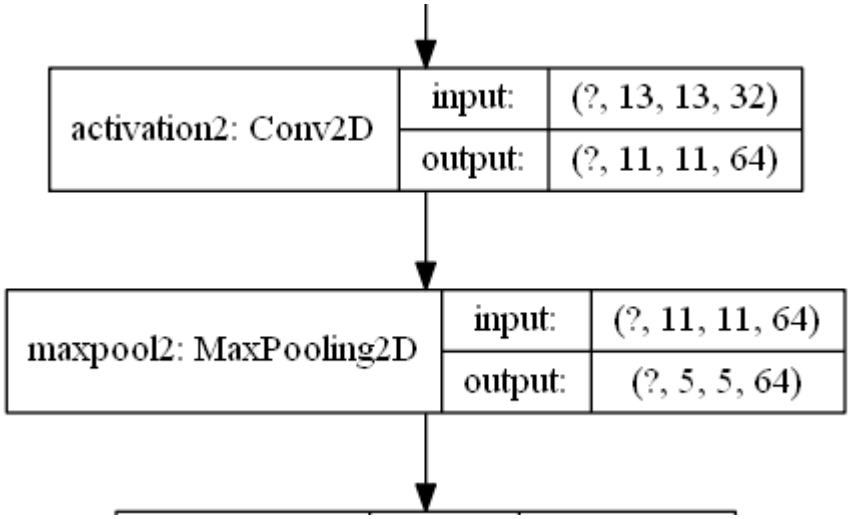


Kuvio 21. MaxPooling2d:n vaikutus

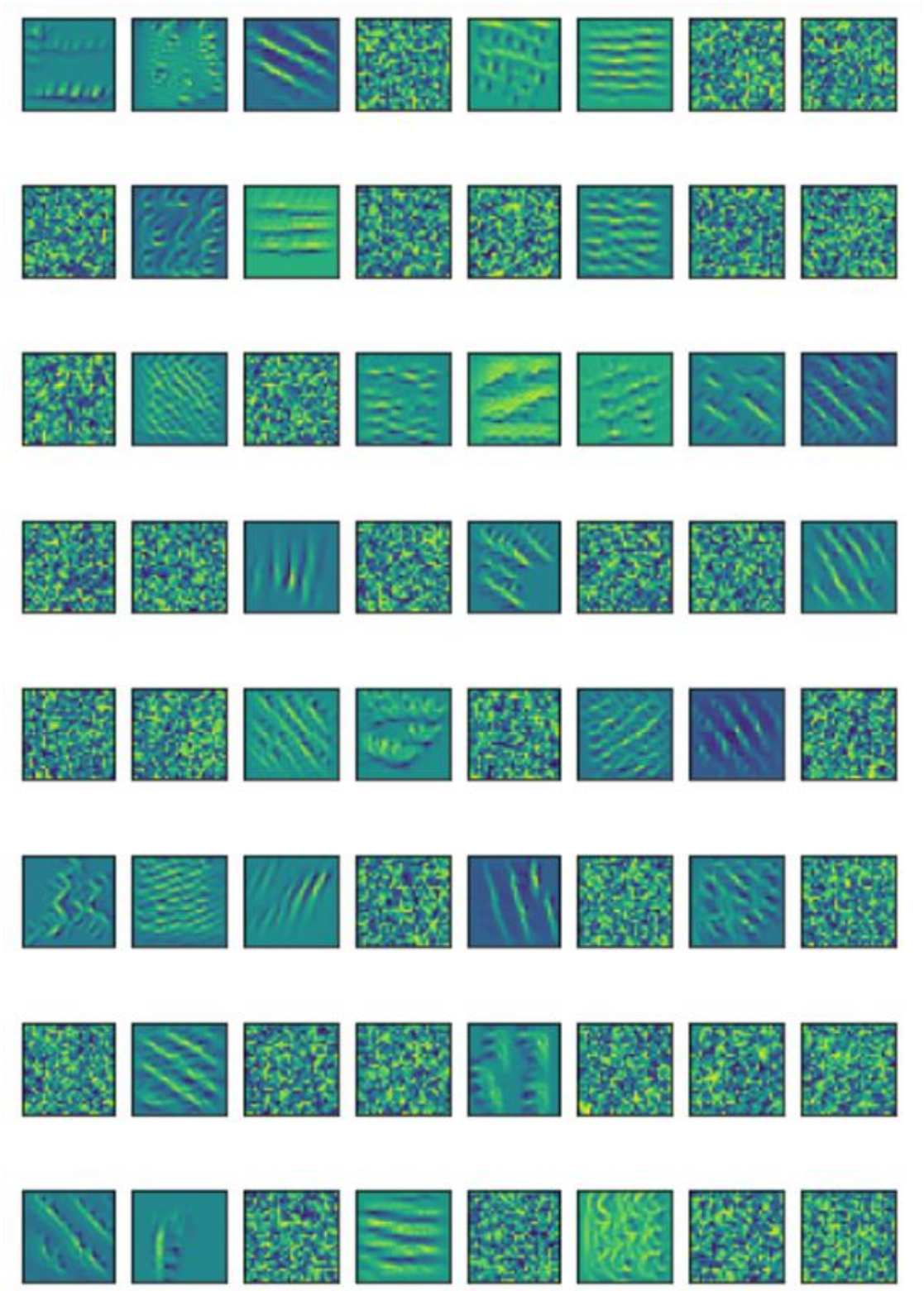


Kuvio 22. MaxPooling2d:n vaikutus

Neuroverkon seuraava kerros on toinen konvoluutiokerros, jolla kernelin koko on taas 3*3 ja tällä kertaa piirrekartan kerroksien määrä tuplataan 64:ään (ks. kuvio 23). Kuviossa 24 piirrekartat, useat ovat silmämääräisesti katsottuna kohinaa, mutta joillain tasoilla erottu selkeitä piirteitä.



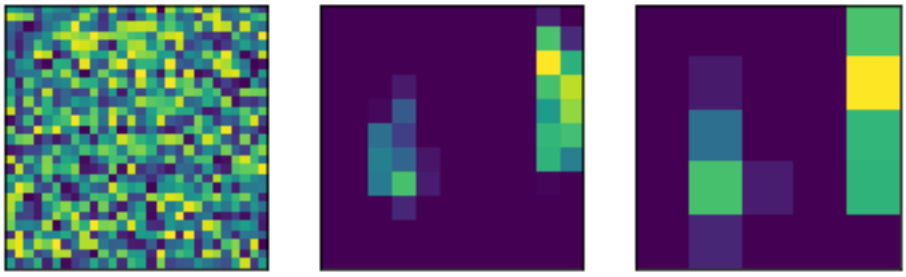
Kuvio 23. Konvoluutiokerros ja MaxPooling2D



Kuvio 24. Toisen konvoluutiokerroksen piirrekartat

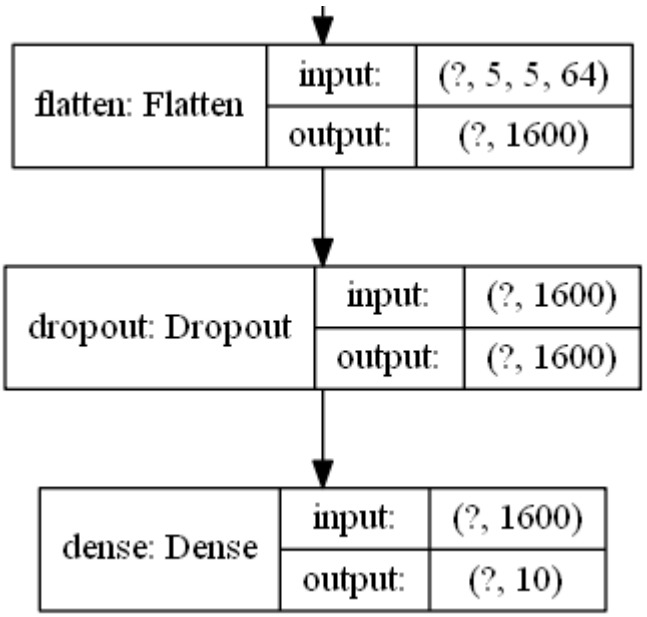
Ilmeisesti yhteyksien monimutkaisuudesta johtuen, toisen konvoluutiokerroksen aktivaatiosta ei ole enää numerosta paljoakaan jäljellä ihmissilmälle tulkittavaksi (ks. kuviot 25 ja 26). Kuvioissa ensin piirrekartta, toiseksi konvoluutiokerroksen aktivaatio ja kolmanneksi MaxPooling2D-kerroksen ulostulo.

Kuvio 25. Aktivaatiot



Kuvio 26. Aktivaatiot

Toisen yhdistekerroksen jälkeen piirrekarttojen ulottuvuudet ovat 5*5*64, ja ne kytketään Flatten-kerrokseen, joka hävittää ulottuvuudet ja suoristaa neuronit yksiulotteiseksi. Välissä on dropout-kerros, jolla osa neuronien välisistä kytköksistä jätetään pois, jolla voidaan ehkäistä ylioppimista. Lopuksi viimeisenä kerroksena on täysin kytketty kerros, jossa on ulostulo neuroni jokaiselle luokalle, eli tässä tapauksessa 10 (ks. kuvio 27).



Kuvio 27. Viimeiset kerrokset neuroverkossa

9 Saksan liikennemerkkien kuvat

Koska haluttiin testata generoitujen kuvien avulla koulutetun neuroverkon suoriutumista reaalimaailman kuviin käytettynä, tarvittiin riittävästi testikuvia, joilla testata luokittelun tarkkuutta. Suomalaisista liikennemerkeistä ei löytynyt kattavaa datasettiä, mutta saksalaisista liikennemerkeistä löytyi datasetti, jota on käytetty erilaisissa koneoppimisprojekteissa. Sekä Saksa, että Suomi noudattavat Wienin liikennemerkkisopimusta, jolloin merkit ovat yhtenevät suurelta osin, suurimpana erona on Suomessa käytetty keltainen väri valkoisen sijaan monissa merkeissä.

Koska Suomen liikennemerkit poikkesivat väritykseltään datasetin kuvista, päätettiin muokata käsin Suomen liikennemerkkejä esittäviä SVG-kuvia vastaamaan Saksan liikennemerkkejä (ks. kuvio 28). Liitteessä 1 on esimerkkikuvat kaikista luokista.

Kuvio 28. Valokuva liikennemerkistä, Suomen versio ja muokattu versio

Saksan liikennemerkkien data setti koostuu 43 luokasta ja yli 50 000 kuvasta. Kuvien koot vaihtelevat 15*15 pikselistä aina 250*250 pikseliin saakka, ja ne ovat PPM-formaatissa. (Dataset 2010.)

Nopea tarkastelu osoitti, että kaikille datasetin luokille ei löytynyt suomalaista vastinetta, joten ne rajattiin pois tätä työtä tehdessä. Jäljelle jäi koulutuskuvia 37409 kuvaa 38 eri liikennemerkki luokasta. Testaukseen tarkoitettuja kuvia oli 12030 kuvaa myös 38 eri luokkaa rajauksen jälkeen.

10 Liikennemerkkien kuvien generointi

Uudistetun tieliikennelain mukaiset liikennemerkkien kuvat ovat saatavilla Väyläviraston internetsivuilla monessa eri formaatissa, myös SVG:nä, jota tässä työssä käytettiin. Kuvat jaetaan zip-paketteina, jotka ladattiin ja purettiin samaan kansioon. Osa kuvista sisälsi ääkkösiä, joten kuvat päätettiin nimetä uudelleen ja erottaa SVG-tiedostot erilleen. Tätä varten kirjoitettiin Python-kielellä skripti (ks. kuvio 29). Tällä skriptillä saatiin kuvat pää- ja alaversioista yhteen kansioon, ja nimet eivät enää sisältäneet hankalia merkkejä, ja tiedostonimien pituus oli sopivampi Blenderiä varten.

```
import glob
import shutil
```

```
paaversiot = glob.glob('..\\liikennemerkit\\*\\1_*\\SVG\\*.svg')
alaversiot = glob.glob('..\\liikennemerkit\\*\\2_*\\SVG\\*.svg')
tallennuspolku = '..\\merkit\\'
```

```
for polku in paaversiot:
    shutil.copy(polku, tallennuspolku + polku.split('SVG\\')[-1].split('_')[0] + '.svg')
```

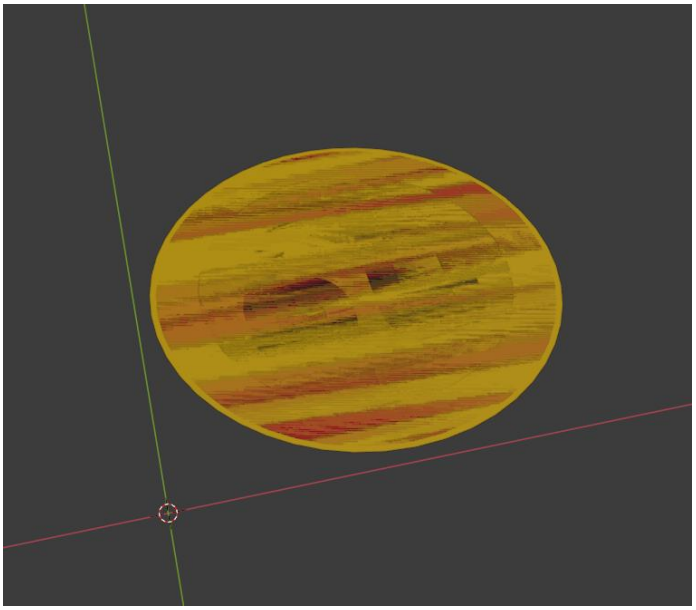
```
for polku in alaversiot:
    tmp = polku.split('SVG\\')[-1].split('_')[0] + '_' + polku.split('SVG\\')[-1].split('_')[1]
    shutil.copy(polku, tallennuspolku + tmp + '.svg')
```

Kuvio 29. Python-skripti

10.1 SVG:n tuonti Blenderiin

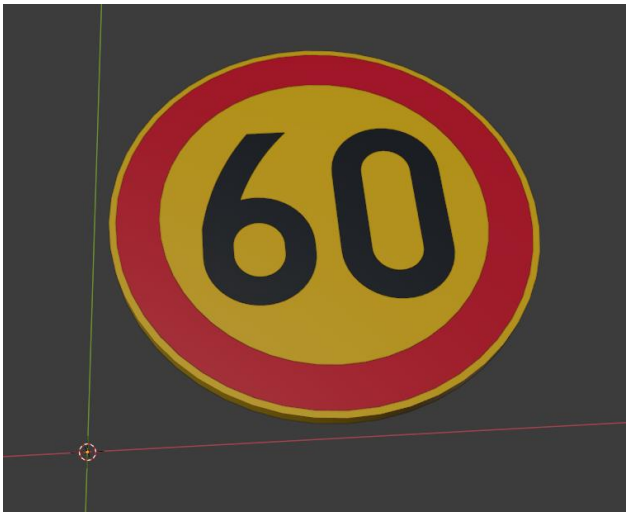
SVG-muotoiset kuvat on helppo tuoda Blenderiin, kun siihen tarkoitettu Blenderin lisäosa otetaan käyttöön (ks. kuvio 30). Lisäosa tekee vektorigrafiikasta Bezier-käyriä, jotka ovat aluksi samassa tasossa, joka aiheuttaa ongelmia renderöinnissä (ks. kuvio 31).

Kuvio 30. SVG-lisäosa



Kuvio 31. Blenderiin tuotu liikennemerkki

Ongelmallinen renderöinti ratkaistiin lisäämällä Bezier-käyrille myös paksuutta, ensimmäiselle käyrälle 2 mm ja kasvattamalla pursotuksen vahvuutta 0,01 mm jokaiselle käyrälle. Lopputulos on siisti mallinnos liikennemerkistä (ks. kuvio 32).



Kuvio 32. Pursotettu liikennemerkki

10.2 Lisäosa

Blenderiin on sisällytetty Python-tulkki, joka käynnistetään Blenderin kanssa ja pysyy aktiivisena. Tämä tulkki ajaa skriptejä käyttöliittymän piirtämiseen, ja sitä käytetään myös joihinkin Blenderin sisäisiin työkaluihin. Kyseessä on tyypillinen Python ympäristö, normaalit Python-skriptit toimivat myös Blenderissä. Blender tarjoaa moduulin Python tulkille, jonka avulla saadaan pääsy Blenderin dataan, luokkiin ja funktioihin. (Python API overview, 2020.)

Blenderiin voi itse kirjoittaa lisäosia (engl. *Add-On*), joiden avulla voi laajentaa Blenderin toiminnallisuutta. Kuviossa 33 yksinkertaisin lisäosa, joka tulostaa viestin konsoliin otettaessa käyttöön, ja poistettaessa käytöstä.

```
bl_info = {
    "name": "My Test Add-on",
    "blender": (2, 80, 0),
    "category": "Object",
}
def register():
    print("Hello World")
def unregister():
    print("Goodbye World")
```

Kuvio 33. Yksinkertaisin lisäosa

Blenderiin kirjoitettiin lisäosa, jotta koulutuskuvien renderöinti eri asetuksilla olisi helpohkoa ja nopeaa. Lisäosaan tehtiin seuraavat ominaisuudet:

- Alkualustus, jolla tuhotaan kaikki objektit, ja luodaan kamera, valo ja taso, joka toimii taustana.
- SVG-tiedoston/tiedostojen valinta tiedostoselaimella, kuvien perusteella mallinnetaan liikennemerkin 3D-malli
- Luodaan liikennemerkin ympärille kehiä, joita käytetään kameran liikeratoina
- Animoidaan kameran liike liikennemerkin ympärillä, ja valitaan kuvaruutujen määrä per kehä
- Taustalle animoidaan kohinaa
- Valon voimakkuutta animoidaan halutulla välillä joka kuvaruudulle satunnainen määrä
- Renderöidään kuvasarjat

11 Konvoluutioneuroverkon rakenne ja koulutus

Konvoluutioneuroverkon pohjaksi valittiin VGG16, joka ladattiin Keraksen kautta ilman viimeisiä kerroksia. Mukana ladattiin Imagenetin kuvilla koulutetut painoarvot, ja malliin lisättiin tulkintaan tarvittavat kerrokset (ks. kuvio 34).

```
base_model = VGG16(weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(CLASSES, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)


for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=[CategoricalAccuracy()])
```

Kuvio 34. Mallin koodia

Ensin mallin runkona toimivat kerrokset jäädytettiin ja koulutettiin kolme epochia ainoastaan lisättyjä kerroksia (ks. kuvio 35). Koulutuksessa käytettiin Saksan liikennemerkkien datasetin koulutuskuvia eli 37409 kuvaa, kuuluen 38 eri luokkaan. Kuvat muokattiin resoluutioon 50*50 pikseliä, käyttäen Keraksen kuvadatageneraattoria.

```
Epoch 1/3
585/585 [==============================] - 28s 47ms/step - loss: 1.4092 - categorical_accuracy: 0.5543
Epoch 2/3
585/585 [==============================] - 16s 28ms/step - loss: 0.8685 - categorical_accuracy: 0.7144
Epoch 3/3
585/585 [==============================] - 16s 27ms/step - loss: 0.6622 - categorical_accuracy: 0.7808
```

Kuvio 35. Mallin karkeakoulutus

Tämän jälkeen mallin jäädytettyjä kerroksia vähennettiin, ja koulutettiin lisää. Tällä kertaa koulutettiin pienemmällä opetuskertoimella ja eri optimointifunktiolla (ks. kuviot 38).

```
for layer in model.layers[:11]:
    layer.trainable = False
for layer in model.layers[11:]:
    layer.trainable = True
```

Kuvio 36. Mallin kerrosten jäädytykset

```
from tensorflow.keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy',
              metrics=[CategoricalAccuracy()])

history12 = model.fit(train_generator, epochs=3, validation_data=validation_generator)
```

Kuvio 37. Mallin hienokoulutuksen parametrejä

```
Epoch 1/3
585/585 [==============================] - 34s 58ms/step - loss: 0.3340 - categorical_accuracy: 0.8902
Epoch 2/3
585/585 [==============================] - 34s 59ms/step - loss: 0.1404 - categorical_accuracy: 0.9552
Epoch 3/3
585/585 [==============================] - 35s 59ms/step - loss: 0.0743 - categorical_accuracy: 0.9775
```

Kuvio 38. Hienokoulutus

Mallin suoriutumista kuvien luokittelussa arvioitiin käyttämällä Keraksen *evaluate*-funktiota, testikuvia oli 12030. Tarkkuudeksi tuli n. 78 % (ks. kuvio 39). Tätä arvoa käytettiin vertailuarvona, kun generoidulla datalla koulutettua mallia arvioidaan.

```
1 model.evaluate(test_generator)
188/188 [==============================] - 8s 43ms/step - loss: 0.9809 - categorical_accuracy: 0.7839
```

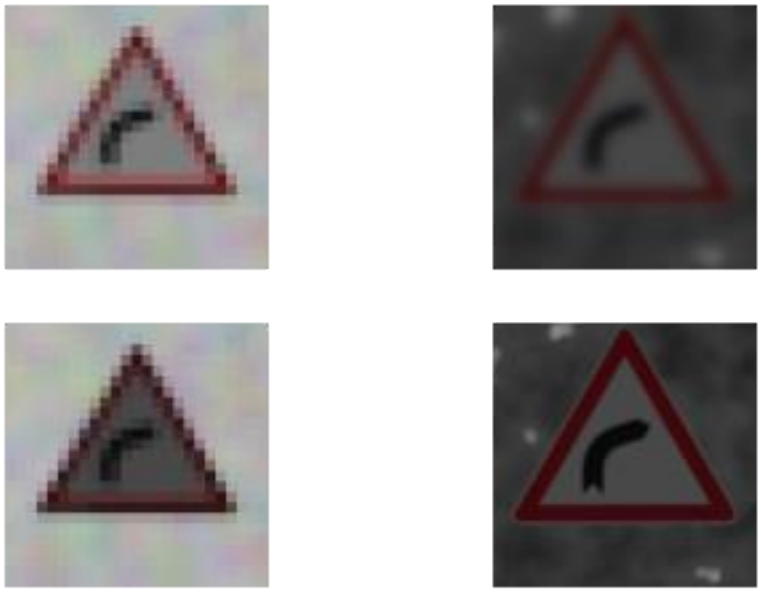
Kuvio 39. Mallin suoriutumisen arviointi

12 Generoitu data

12.1 Generoidut kuvat yleisesti

Opetuskuvia generoitiin vaihdellen välillä kuvan taustaa, kameran kohdistusta, resoluutiota, ja valaistuksen määrää. Lisäksi renderöitiin yksi kuvasarja, johon lisättiin Gaussilaista sumennusta, sekä yksi kuvasarja, jota pikselöitiin tarkoituksella. Kuvia renderöitiin aina 60 kuvaa per liikennemerkki, kun kamera liikkui sivuttaissuunnassa

kehällä liikennemerkkiin nähden. Kuvassa 40 muutama esimerkki generoiduista kuvista. Kuvasarjoja generoitiin kaikkiaan 12. Esimerkit kaikista sarjoista liitteessä 2.



Kuvio 40. Esimerkkejä generoiduista kuvista

12.2 Kuvien ominaisuuksia kuvasarjoittain

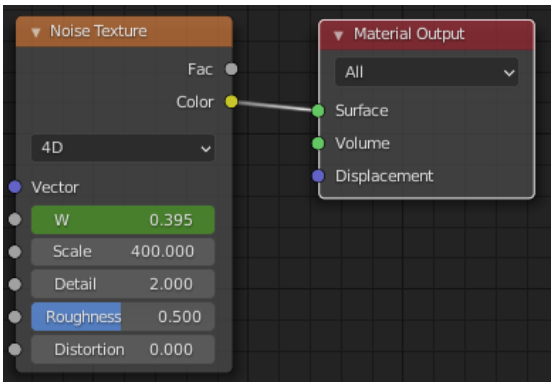
12.2.1 Musta tausta

Ensimmäiseen koulutuksessa käytettyyn kuvasarjaan luotiin musta tausta ja sen kuvakoko oli 50*50 pikseliä. Esimerkki kuviossa 41.

Kuvio 41. Esimerkki mustasta taustasta

12.2.2 Värikohina

Toiseen kuvasarjaan tehtiin värikohinaa Blenderin tekstuuri-nodeilla (ks. kuviot 42 ja 43), ja kohinaa animoitiin joka kuvalle erilaiseksi. Esimerkki kuviossa 44. Kuvan koko oli 25*25 pikseliä.



Kuvio 42. Kohina nodet

Kuvio 43. Taustalla oleva värikohina

Kuvio 44. Esimerkki värikohinasta

12.2.3 Gaussialainen sumennus

Kolmanteen kuvasarjaan tehtiin kohina-tekstuurin avulla tausta ja käytettiin Blenderin videoeditointiominaisuuden avulla luotiin Gaussialaista sumennusta (ks. kuvio 45). Kuvakoko oli 100*100 pikseliä.

Kuvio 45. Gaussialainen sumennus

12.2.4 Matala valoisuus

Neljäntenä käytettiin jälleen värikohina taustaa ja valoisuus pudotetiin minimiin ja pikselikoko 25*25 pikseliin. Esimerkki kuviossa 46.

Kuvio 46. Matala valontaso

12.2.5 Matalavaloisuus, tumma tausta

Käytettiin myös toista matalaa valoisuutta, ja taustana tummempi tausta (ks. kuvio 47).

Kuvio 47. Matala valoisuus, tumma tausta

12.2.6 Matalavaloisuus, suurempi koko

Kuviossa 48 muutoin sama kuin edellinen, mutta pikselien määrä kasvatettiin 50*50 pikseliin.

Kuvio 48. Tumma tausta, suurempi koko

12.2.7 Matalavaloisuus, suurin koko

Kuviossa 49 esimerkki kuvasta, jossa pikselien määrää kasvatettiin edelleen kokoon 100*100 pikseliä.

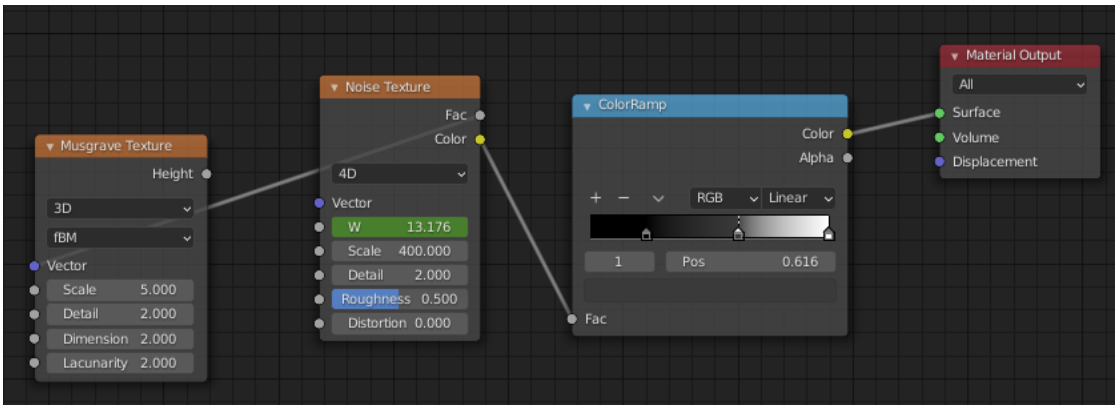
Kuvio 49. 100*100 pikseliä

12.2.8 Musgrave-tausta

50*50 pikselin Musgrave-taustalla kuviossa 50.

Kuvio 50. Musgrave tausta

Kuviossa 51 on taustan luovat tekstuurinodet.



Kuvio 51. Musgrave-taustan nodet

12.2.9 Musgrave pienempi koko

Yhdeksäs koulutuksessa käytetty kuvasarja oli samankaltainen kuin edellinen, mutta pikselimäärä pudotettiin 25*25 pikseliin. Esimerkki kuviossa 52.

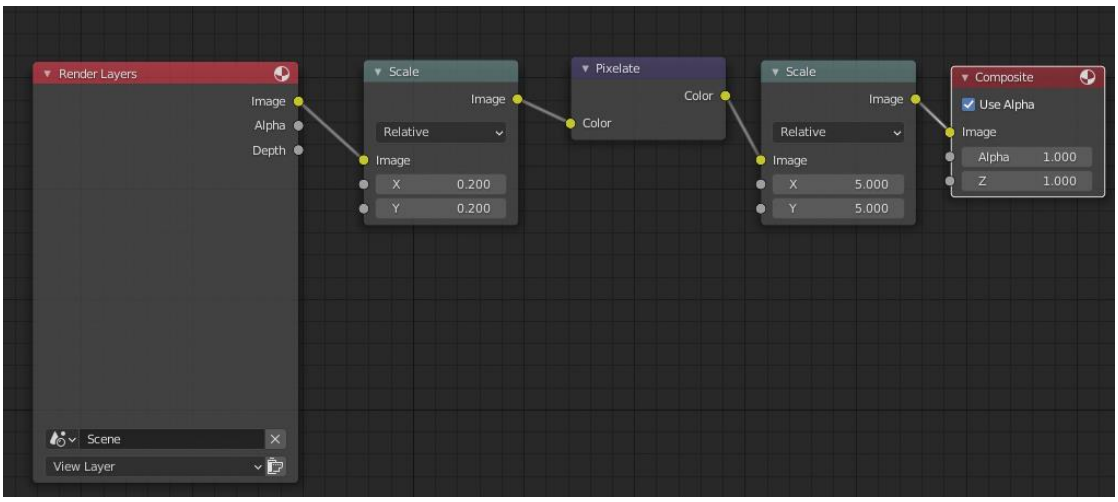
Kuvio 52. Musgrave tausta

12.2.10 Pikselöinti

Koska osa testikuvista oli myös pikselöitynyt pahasti, testattiin myös tarkoituksella pikselöityä kuvaa (ks. kuvio 53).

Kuvio 53. Pikselöity kuva

Pikselöintiin käytettiin Blenderin Compositing-tilan nodeja (ks. kuvio 54).



Kuvio 54. Kuvan pikselöinti

12.2.11 Vaalean tausta raidoilla

Tehtiin myös kuvasarja, jossa oli väri raitoija, mutta koska pikselimäärä oli pieni (25*25 pikseliä) ei raidat juuri erotu (ks. kuvio 55).

Kuvio 55. Raidat

12.2.12 Valkoinen tausta

Viimeinen käytetty kuvasarja oli valkea tausta (ks. kuvio 56). Pikselimäärä 50*50.

Kuvio 56. Valkea tausta

13 Mallin koulutus

Mallia päätettiin kouluttaa kuvasarjoja lisäämällä yksi kerrallaan, jotta myös datamäärän lisäämisen vaikutuksia voitiin arvioida edes jossain määrin. Mallin rakenne ja koulutus parametrit olivat samat kuin oikeilla kuvilla koulutettaessa. Taulukossa 1 on mallin koulutuksen tuloksia. Johtuen koulutuksen sattumanvaraisuudesta, tulokset eivät ole täysin vertailtavissa keskenään, mutta niistä voidaan kuitenkin tulkita, että malli oppii hyvin koulutusaineiston jo pienellä koulutuskuvien määrällä ja yleistyy oikeisiin kuviin kohtaisella tarkkuudella, mutta tarkkuus ei parane merkittävästä kuvia lisättäessä.

Taulukko 1. Mallin koulutusta

	Kuvien lkm.	Koulutus häviö	Koulutus tarkkuus	Testi häviö	Testi tarkkuus
1.	2280	0,1701	0,9851	4,4759	0,1720
2.	4560	0,1009	0,9807	6,8691	0,1981
3.	6840	0,0465	0,9931	8,5514	0,2098
4.	9120	0,0291	0,9964	9.2447	0,2244
5.	11400	0,0262	0,9968	6,8417	0,3182
6.	13680	0,0152	0,9985	7.1846	0,3207
7.	15960	0,0274	0,9944	6,3391	0,3524
8.	18240	0,0337	0,9923	6,5391	0,3754
9.	20520	0,0269	0,9937	6,4746	0,3744
10.	**22800**	**0,0206**	**0,9948**	**6,4204**	**0,3794**
11.	25080	0,0192	0,9953	6,8984	0,3914
12.	27360	0,0215	0,9947	7,7030	0,3684

Tulosten perusteella tehtiin oletus, että generoitujen kuvasarjojen samankaltaisuus keskenään aiheuttaa, sen että kuvien lisääminen ei paranna mallin suoriutumista. Lisäksi kuvat eivät todennäköisesti muistuttaneet riittävästi oikeita kuvia, esimerkiksi generoitujen kuvien liikennemerkit ovat samassa kulmassa, kun oikeissa kuvissa oli havaittavissa liikennemerkkien vinoutta. Päätettiin kokeilla olisiko Keraksen kuvadatageneraattorin kuvien rikastamistoiminnoilla vaikutusta mallin suoriutumiseen. Taulukossa 2 on testauksen tulokset. Esimerkkejä kuvien rikastamistoiminnoista kuvioissa 57–59.

Taulukko 2. Mallin koulutus rikastamalla

Kuvien lkm.	Koulutus häviö	Koulutus tarkkuus	Testi häviö	Testi tarkkuus	Shear	Zoom	Kirkkaus
15960	0,0188	0,9969	6,7055	0,3502	0,5	x	x
27360	0,0208	0,9954	7,7309	0,3564	0,5	x	x
15960	0,1467	0,9517	4,2025	0,4632	x	0,5	x
27360	0,1285	0,9574	4,1810	0,4768	x	0,5	x
15960	0,0946	0,9692	4,1312	0,4730	x	x	0,1–1,5
27360	0,0762	0,9766	4,5488	0,4819	x	x	0,1–1,5
15960	0,1466	0,9537	4,2848	0,4571	0,5	0,5	x
27360	0,1390	0,9536	4,2530	0,4569	0,5	0,5	x
15960	0,2563	0,9119	2,9437	0,5110	x	0,5	0,1–1,5
27360	**0,2132**	**0,9274**	**2,9188**	**0,5637**	**x**	**0,5**	**0,1–1,5**
15960	0,0900	0,9714	4,4154	0,4500	0,5	x	0,1–1,5
27360	0,0692	0,9772	5,2184	0,4458	0,5	x	0,1–1,5
15960	0,2418	0,9194	2,9612	0,5416	0,5	0,5	0,1–1,5
27360	0,2013	0,9329	2,9950	0,5556	0,5	0,5	0,1–1,5

Kuvio 57. Shear

Kuvio 58. Zoom

Kuvio 59. Kirkkaus

Kuvien rikastaminen paransi mallin suoriutumista kohtalaisesti, joka tarkoittaa, että generoitua datasettiä voisi kehittää edelleen monimuotoisemmaksi, jotta se olisi edustavampi.

14 Pohdinta

Tavoitteena oli tuottaa kuva-aineistoa, jolla konvoluutioneuroverkkoa koulutetaan ja suoritumista mitataan reaalimaailman kuviin. Generoidulla aineistolla koulutettu malli suoriutui huomattavasti heikommin, kuin oikeilla kuvilla koulutettu malli, mutta tulokset ovat lupaavia. Aineiston monimuotoisuutta lisäämällä pääsisi parempiin tuloksiin, mutta on haasteellista tietää ennalta millaisilla muutoksilla tulokset paranevat merkittävästi. Generoidun kuvan ja tosimaailman kuvan välille jää melkein väistämättä jotain eroa, ja on mahdollista että neuroverkko tarttuu juuri tuohon eroon.

Datasetin generointi on kiinnostava tutkimuksen aihe, koska laadukkaan datasetin luominen on monivaiheinen ja usein työläs prosessi. Esimerkiksi liikennemerkkien tapauksessa ensin vaaditaan riittävästi kuvamateriaalia liikennemerkeistä. Tämän jälkeen täytyy mahdollisesti rajata merkit kuvista tai tehdä rajauslaatikot, ja sitten ne voi luokitella.

Koska Suomen liikennemerkeistä ei löytynyt aineistoa ja testikuvien hankkimiseksi opinnäytetyötä varten kokeiltiin kahta tekniikkaa: kuvata itse ja merkata kuvat käsin, sekä etsiä valmiita kuvia, joista rajattiin liikennemerkit. Molemmat tekniikat osoittautuivat aikaavieviksi, eikä niiden kattavuus olisi ollut riittävä. Tämän takia päädyttiin käyttämään Saksan liikennemerkkejä, joista löytyi kattava datasetti.

Opinnäytetyö haki lopullista muotoaan pitkään, tarkempi rajaus alussa olisi varmastikin helpottanut tekemistä, mutta oli hyvin opettavaista tutustua tarkemmin aiheisiin, jotka jäivät raportissa lähinnä alikappaleiksi (tai kokonaan ulkopuolelle), kuten esimerkiksi erilaiset kuvan segmentointitekniikat ja suosituimmat konvoluutioneuroverkkoarkkitehtuurit. Työtä tehdessä oppi paljon syväoppimisprojektin vaatimuksista ja erityisesti oppia tuli pitkäjänteisestä projektityöskentelystä.

Työssä ei paneuduttu lopulta hirveästi itse neuroverkon kehittämiseen, voi olla että toisenlaisella neuroverkon rakenteella voisi saada parempia tuloksia. Käytetty malli suoriutui reaalimaailman kuvilla koulutettunakin vain n. 78% tarkkuudella, josta on varmaankin mahdollista parantaa jo pelkillä koulutuksen hyperparametrien muokkauksella.

Generoidun datasetin avulla koulutetut mallit jättivät siis toivomisen varaa, mutta tekniikkaa voisi käyttää eräänlaisena aloitusdatasetin generoimisena. Generoidulla datasetillä voisi aluksi kouluttaa mallin, jonka avulla tunnistetaan haluttuja esineitä tai asioita kuvista. Tämän jälkeen asiantuntija hyväksyy tai korjaa luokittelut. Näitä uusia luokiteltuja kuvia voitaisiin käyttää mallin koulutuksessa uudelleen, jolloin mallin tarkkuus paranee, ainakin toivottavasti. Kun oikeita kuvia on luokiteltu riittävästi, voisi generoidusta datasetistä luopua.

Olisi kiinnostavaa lisäksi testata esimerkiksi esineen rajausta kuvista vastaavilla tekniikoilla, koulutukseen tarvittavat maskikuvat olisi helppo generoida vaikka Blenderin materiaaleja muokkaamalla ja sopivalla taustan valinnalla.

Työn tuloksena toimeksiantajalle jäi raportin lisäksi työssä käytetty scripti, jolla kuvia generoitiin ja ohjeet sen käyttöön.

Lähteet

About. N.d. Tietoja Blenderistä verkkosivu. Viitattu 17.11.2020. https://www.blender.org/about/

About Keras. N. d. Tietoja Keraksesta verkkosivu. Viitattu 28.7.2020. https://keras.io/about/

A Brief History of Computer Graphics. 2004. Verkkosivusto. Viitattu 9.11.2020. http://www.comphist.org/computing_history/new_page_6.htm

Badler, N. I., Glassner, A. S. N.d. 3D object modeling. Viitattu 12.11.2020. http://gamma.cs.unc.edu/courses/graphics-s09/LECTURES/3DModels_SurveyPaper.pdf

Beane, A. 2012. 3D animation essentials. Books24x7 versio. Indianapolis, Ind.: J. Wiley & Sons.

Brownlee, J. 2019. How Do Convolutional Layers Work in Deep Learning Neural Networks?. Verkkoartikkeli. Muokattu 17.4.2020. Viitattu 12.2.2021. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Computer Vision - What it is and why it matters. N.d. Internetartikkeli. Viitattu 8.11.2020. https://www.sas.com/en_us/insights/analytics/computer-vision.html

Dataset. 2010. Internetsivusto Saksan liikennemerkkien aineistosta. Muokattu 10.5.2019. Viitattu 25.3.2021. https://benchmark.ini.rub.de/gtsrb_dataset.html

Get Involved. N.d. Blenderin verkkosivusto. https://www.blender.org/get-involved/

Géron, A. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. 2 p. Sebastopol, CA: O'Reilly.

Gumster, J. van. 2011. Blender for dummies, 2nd edition. Books24x7 versio. J. Wiley & Sons.

Kelleher, J. D. 2019. Deep learning. Books24x7 versio. MIT press.

LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Hubbard, W., Jackel, L. D., ja Henderson, D. 1989. Handwritten Digit Recognition with a Back-Propagation Network. Viitattu 9.11.2020. http://yann.lecun.com/exdb/publis/pdf/lecun-90c.pdf

Lucci, S., Kopec, D. 2013. Artificial intelligence in the 21st century: a living introduction. Books24x7 versio. Mercury Learning 2p. Viitattu 9.11.2020

McGarthy, J., Minsky, M. L., Rochester, N., Shannon, C. E., 1956. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. Viitattu 11.11.2020. http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf

ML Practicum: Image Classification. 2020. Verkkosivu. Viitattu 17.11.2020. https://developers.google.com/machine-learning/practica/image-classification

Moor, J. 2006. The Dartmouth College Artificial Intelligence Conference: The Next Fifty Years. AI Magazine, 27(4), 87-91.

Oppenheimer R. 2018. William Fetter, E.A.T., and 1960s Computer Graphics Collaborations in Seattle. Verkkoessee. Viitattu 10.11.2020. https://www.history-link.org/File/20542

Planche, B., Andres, E. 2019 Hands-On Computer Vision with TensorFlow 2. Birmingham: Packt Publishing

Python API overview. 2020. Blenderin dokumentaatiosivusto. Viitattu 11.12.2020. Muokattu 25.1.2020. https://docs.blender.org/api/current/info_overview.html

Raschka, S., Mirjalili, V. 2019. Python Machine Learning Third edition. Birmingham: Packt Publishing

Slick, J. What is 3D modelling?. 2020. Verkkoartikkeli. Viitattu 21.7.2020. https://www.lifewire.com/what-is-3d-modeling-2164

Teknologiayksikkö. Tietoa JAMK:n teknologiayksiköstä verkkosivu. Viitattu 18.4.2021. https://www.jamk.fi/fi/Tietoa-JAMKista/Teknologiayksikko/

Tensorflow. 2020. Tensorflow repositorion README.md. Viitattu 28.7.2020 https://github.com/tensorflow/tensorflow/blob/master/README.md

Vuori, J. N.d. Laadullisen tutkimuksen verkkokäsikirja. Kappale Johdatus laadulliseen tutkimukseen ja verkkokäsikirjaan. Viitattu 31.3.2021. https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/mita-on-laadullinen-tutkimus/johdatus-laadulliseen-tutkimukseen-ja-verkkokasikirjaan/

Liitteet

Liite 1.









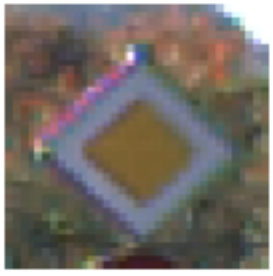
















STOP

STOP

STOP
















60

60


20

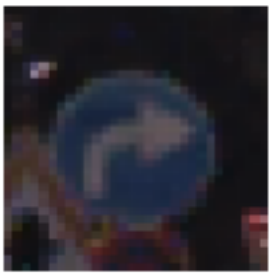
20

20



















Liite 2.