Tensorflow-neuroverkot Fyysisten robottien apuna

Kolikkojärjestelijärobotti kuvatunnistuksella

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, tietojenkäsittely

kevät, 2019

Valtteri Valonen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tradenomi, Tietojenkäsittely
Visamäki, Hämeenlinna

Tekijä Valtteri Valonen **Vuosi** 2019

Työn nimi Tensorflow-neuroverkot Fyysisten robottien apuna

Työn ohjaaja/t Tommi Lahti

TIIVISTELMÄ

Tämän opinnäytetyön tavoite oli selvittää neuroverkkojen soveltuvuutta kevyissä päätelaitteissa ja roboteissa, kuten Raspberry Pi:ssa. Soveltuvuutta mitataan toimivuudella, helppoudella ja käytettävyydellä.

Tavoitteen selvittämiseksi rakennettiin, ohjelmoitiin ja koulutettiin kuvapohjainen kolikontunnistusrobotti. Työn kehitystyötä ja lopputulosta analysoitiin soveltuvuuden arvoimiseksi. Teoriapohjana käytettiin useita Tensorflow-koneoppimiskirjaston opetuslähteitä ja ohjemateriaaleja. Robottia varten tuotettiin kaksi erillistä ohjelmaa: Robotin ohjain ja koulutusympäristö. Robotin ohjain huolehtii koulutetun neuroverkon suorituksesta sekä uuden koulutusdatan keruusta. Koulutusympäristössä voidaan koulutusdatan ja verkon rakenteen pohjalta luoda ohjaimelle uusi koulutettu neuroverkko. Molemmat ohjelmat ovat kirjoitettu node.js pohjalle Javascriptillä.

Lopputuloksena neuroverkot toimivat hyvin kevyissä laitteissa koulutuksen jälkeen, ja auttavat kehitysprosessissa valtavasti. Koulutusprosessin jälkeen laskentatehon tarve on minimaalinen ja kevytkin laite kuten Raspberry Pi riittää.

Avainsanat Tensorflow, neuroverkot, IOT, Raspberry Pi, javascript

Sivut 23 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

Author Valtteri Valonen **Year** 2019

Subject Tensorflow neural networks aiding physical robots

Supervisors Tommi Lahti

ABSTRACT

The purpose of this thesis was to learn and gauge the viability of neural networks in physical robots, like sorting machines. This includes factors such as performance, practicality and usefulness. To achieve the goal, the author built, developed, and trained an image-based coin sorting robot that is controlled by a self-trained neural network.

Theoretical knowledge was mostly gained by utilizing a popular machine learning framework called Tensorflow through various online sources. Theory was applied to shape and develop the image recognition network for the coin sorting robot.

During the development process the author made two pieces of software, one to control the robot and gather sample data, and another one to train a network on that sample data to be used for coin recognition and sorting.

As a result of this thesis, the network was trained and worked well on a lightweight computing platform (in this robot's case the Raspberry Pi), enabling the usage of trained networks inside the robot controller. This means there's no need for networking or expensive computers to run the robot after training. The neural network helped to solve an image classification challenge very easily from the developer's point of view.

Keywords Neural networks, Tensorflow, Raspberry Pi, Javascript

Pages 23 pages

SISÄLLYS

Sanasto

Noodi	Matemaattinen objekti, jolla on yksi tai useampi syöte ja yksi ulostulo
Raspberry Pi	Yhden piirilevyn tietokone
Vastavirta-algoritmi	Neuroverkkojen matemaattinen tapa päivittää tarkkuuttaan, liian monimutkainen tämän opinnäytetyön tavoitteisiin
Neuroverkko	Yhdistelmä noodeja ja noodien välisien yhteyksien painoja,koulutettuna toimii funktion imitoijana ja ongelmanratkaisijana
node.js	serveripuolen javascript-tulkki
Tensorflow	Googlen julkaisema neuroverkkojen apukirjasto

1 JOHDANTO

Tämä opinnäytetyö käsittelee Tensorflow-kirjaston ja sen kautta neuroverkkojen hyödyntämistä kevyemmissä laskentaympäristöissä, kuten Raspberry Pi:lla ohjatuissa roboteissa. Opinnäytetyön toiminnallisen toteutuksen aiheena on kolikkojenjärjestelyrobotti, joka toimii kolikkoja kuvaamalla ja niitä tekoälyn kautta tunnistamalla. Työhön kuuluu myös neuroverkon mallinnusohjelma (tai toiselta nimeltään koulutusohjelma), jolla ennalta kerätty esimerkkidata käsitellään ja neuroverkon parametrit saadaan oikein. Koulutuksen jälkeen nämä tiedot siirretään robottiin, joka niitä hyödyntäen järjestelee kolikoita.

Opinnäytetyö pyrkii täten fyysisen demonstraatiolaiteen rakentamisen ja ohjelmoinnin pohjalta vastaamaan tutkimuskysymyksiin ja selvittämään kuinka neuroverkkoja voi hyödyntää oikean elämän käyttötarkoituksiin. Toissijainen tarkoitus on myös toimia esimerkkinä tekoälyn käytöstä ja yhdistämisestä fyysiseen laitteeseen erillisen ohjelmistodemon sijasta.

Opinnäytetyö pyrkii vastaamaan käytännön osuudellaan seuraaviin tutkimiskysymyksiin:

- Miten Tensorflow kirjastoa voidaan käyttää kevyessä päätelaitteessa?
- Mitä hyötyä neuroverkoista on robotin kehitykseen?

2 ROBOTTI JA LAITTEISTO

Opinnäytetyössä käytettävä robotti on puusta ja metallista rakennettu kolikoiden järjestelyrobotti (kuva 1), joka ottaa säiliöstä kolikkoja yksi kerrallaan, kuvaa ne ja kuvan perusteella järjestelee omiin lokerikkoihinsa. Robotin laitteistona toimii Raspberry Pi 3 ja Raspberry Pi camera V2.

Kuva 1. Kolikonjärjestelyrobotti

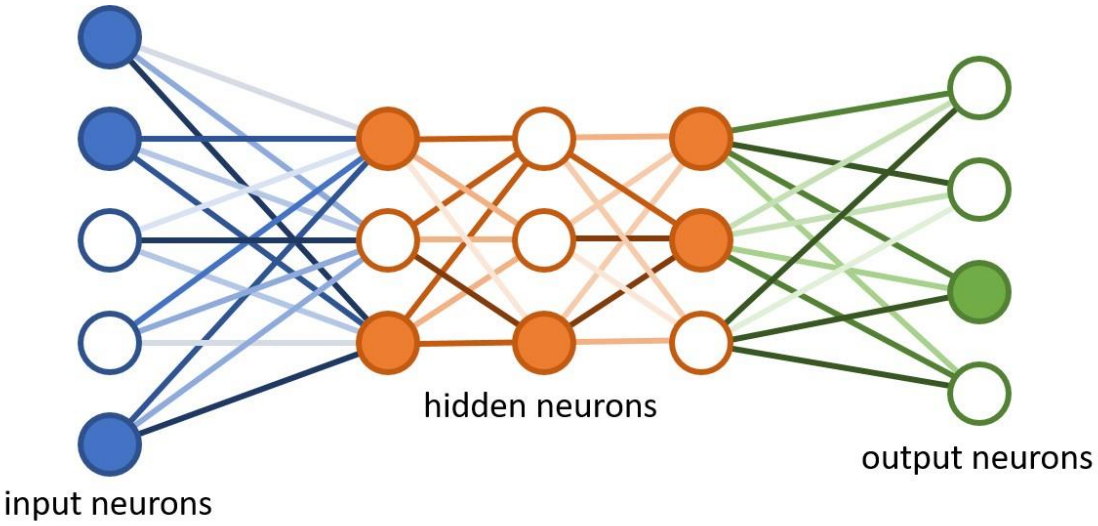
Tämän opinnäytetyön aiheena on tätä robottia ohjaavan TensorFlow neuroverkon suunnittelu, toteuttaminen, kouluttaminen ja testaus laitteessa. Neuroverkkoja suunnitellaan useampi ja niiden koulutusaikaa ja tarkkuutta verrataan keskenään. Itse laitteen rakentamista tai moottorien ohjaimia ei käsitellä.

Opinnäytetyössä on myös käytössä erillinen koulutusympäristö, jossa neuroverkon koulutus tehdään opetusdatan pohjalta. Kouluttaminen vaatii paljon laskentatehoa, joten työn hoitaa tehokas Ryzen 7 –prosessorilla ja 1080ti näytönohjaimella varustettu pöytäkone.

3 NEUROVERKOT

Neuroverkot ovat oma alakäsite laajemmasta käsitteestä koneoppiminen. Neuroverkot kuvataan usein erinäisinä kerroksina noodeja, joiden välillä on yhteyksiä seuraavaan kerrokseen, syötekerroksesta (input) aina tuloskerrokseen (output) asti. (Colah, n.d.)

Perinteisessä neuroverkossa on useampi kerros, joista tiedonsyöttökerroksessa analysoitava data syötetään eteenpäin kaikkien määritettyjen kerrosten läpi. Lopulta data päätyy tuloskerrokseen, josta verkon laskema tulos saadaan selville. Kuvassa 2 on esimerkki simppelin neuroverkon rakenteesta. (Colah, n.d.)



Kuva 2. Neuroverkkojen yleistä rakennetta kuvaava kaavio (Bromley, 2018)

3.1 Neuroverkkojen rakenne ja kerrokset

Koska neuroverkot ovat laaja käsite, tässä opinnäytetyössä keskitytään lyhyesti määrittelemään yleisimmät neuroverkkojen rakenteiden käsitteet ja niihin tarvittavat taustatiedot.

Ensimmäisenä täytyy ymmärtää neuroverkkojen kerroksien merkitys ja niiden väliset sidokset. Usein erinäisiä kerroksia yhdistetään peräkkäin saaden aikaan monimutkaisempia kokonaisuuksia, joilla on kyky ratkaista toinen toistaan hankalampia ongelmia. Kerrosten lisääminen luonnollisesti lisää suoritus- ja koulutusaikaa. Jokainen kerros sisältää yhden tai useamman noodin, eli oman entiteetin, joka saa syötteen, painot ja aktivointifunktiolla tuottaa oman aktivointitason, jota voidaan käyttää joko tuloksena tai seuraavan kerroksen syötteenä. Kerroksista kerrotaan myös lisää luvussa *3.5 Tensorflow layers API.* (Bromley, 2018)

3.1.1 Feed-forward

Feed-forward -neuroverkot ovat verkkoja, joissa tieto kulkee syötekerroksesta tuloskerrokseen, eikä verkolla ole mitään muuta tietoa, paitsi aikaisemman kerroksen arvot ja yhteyden paino. (Gupta, 2017)

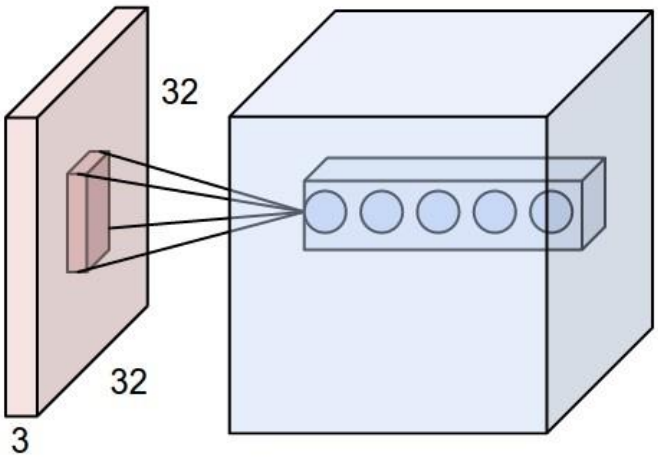
3.1.2 LSTM

LSTM-verkot (Long Short Term Memory) ovat muuten samanlaisia kuin feed-forward, mutta voivat pitää aikaisemmasta suorituskerrastaan muistissa arvoja ja käyttää näitä uudelleen aktivointitason laskennassa. Tämä on hyödyllistä esimerkiksi videoita tulkittaessa, sillä seuraava kuva videossa on usein riippuvainen myös aikaisemmista. (Colah, n.d.)

3.1.3 Konvoluutiokerros

Konvoluutiokerros (convolution) on usein kuvantunnistuksessa käytettävä kerrostyyppi. Konvoluutiokerros kuvataan usein kolmiulotteisina noodikerroksina kuvien kohdalla, sillä kuvat ovat kaksiulotteisia, ja jokaista värikanavaa varten on oma kerroksensa noodeja. Noodikerroksilla tarkoitetaan kokoelmaa neuroverkon noodeja, ja sen ulottuvuudella ja mitoilla kokoa. (C321N, n.d.)

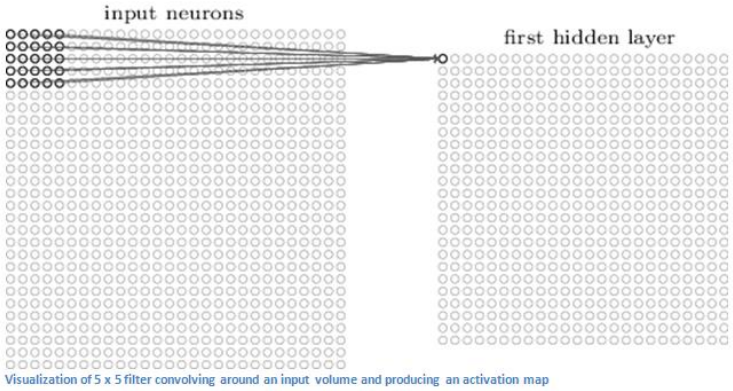
Kuva 3 on esimerkki 32x32x3 syötteestä, jonka tarkkailevassa konvoluutiokerroksessa on useampi suodatin jokaista ikkunaa kohden. Konvoluutiokerroksessa käytettävistä termeistä tämän aliotsikon viimeisessä kappaleessa.



Kuva 3. Konvoluutiokerroksen osa ja sen yhteydet RGB-kuvaan (C321N, n.d.)

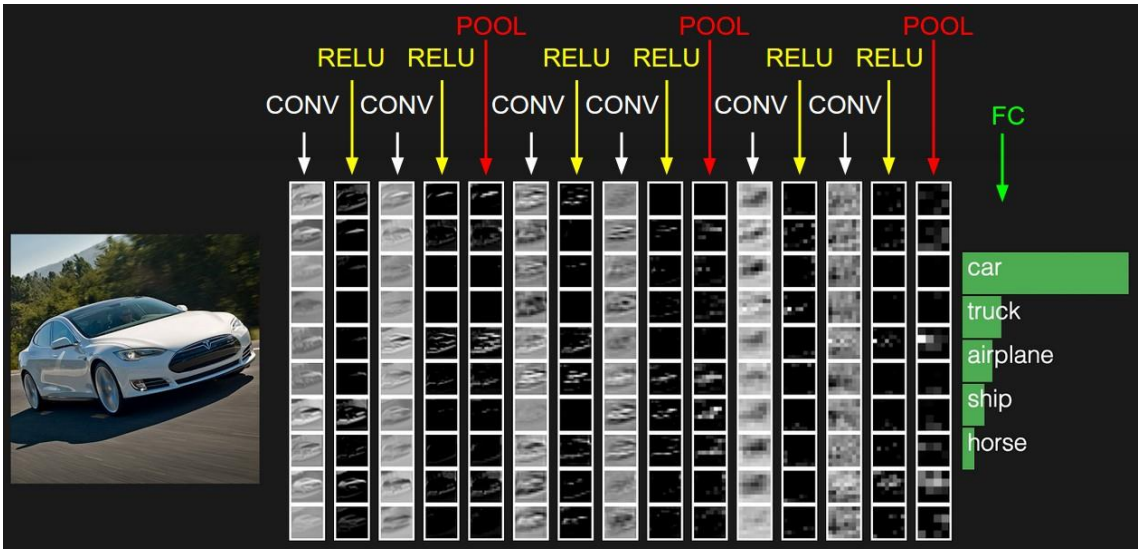
Yksinkertainen esimerkki konvoluutiokerroksesta on mustavalkokuva. Koska mustavalkokuvissa on vain yksi värikanava, kuvalla on vain kaksi ulottuvuutta: x ja y.

Kuva 4 esimerkissä näkyy hyvin, että jokaisella konvoluutiokerroksen noodilla on oma ”katseluikkunansa” syötekerrokseen. Esimerkissä jokainen noodi näkee ja reagoi sitä lähimpiin 5x5 syötteisiin, moniulotteisissa, kuten RGB kuvissa, jokaista väriä varten sama kaava vain toistetaan. (C321N, n.d.)



Kuva 4. Kaksiulotteisen syötteen (esim. mustavalkokuva) jälkeinen 2d konvoluutiokerros (Adeshpande3, n.d.)

Konvoluutiokerrokset ovat erittäin hyviä havaitsemaan esimerkiksi muotoja ja kuvioita kuvista, joita esimerkiksi kuva- ja esinetunnistuksessa käytetään. Toistamalla tätä kuviota aktivointifunktioiden ja useamman konvoluutiokerroksen lävitse voidaan saada hyvinkin tarkkoja kuvientunnistimia. Esimerkki konvoluutiokerroksia käyttävästä neuroverkkorakenteesta on kuvassa 5. (Adeshpande3, n.d.)



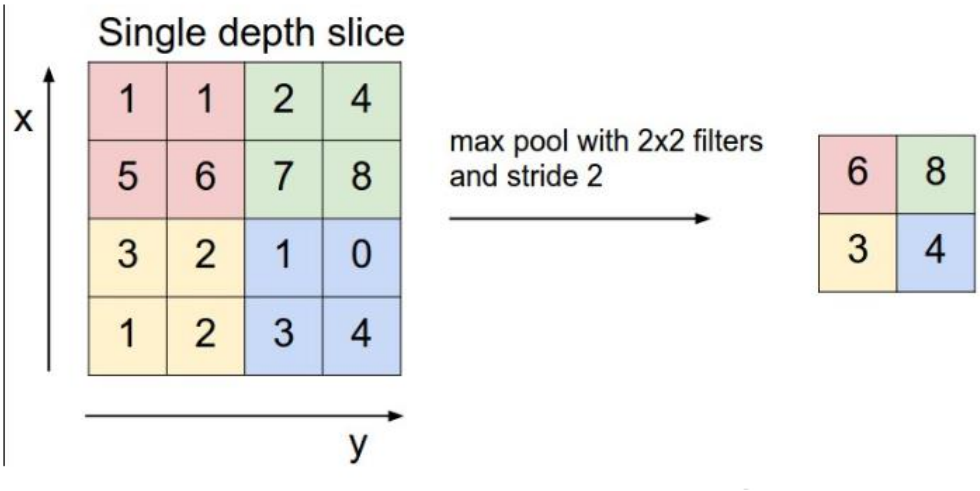
Kuva 5. Kuvatunnistin useammalla konvoluutiokerroksella käyttäen aktivointifunktiona ReLua (CS321N, n.d.)

Konvoluutiokerroksissa on useita termejä, joista tärkeimmät ovat ikkunat, suodattimet ja askeleet. Ikkuna kuvaa aikaisemmasta kerroksesta kunkin filtterin näkemää noodiyhdistelmää, tämä mitta on usein ilmoitettu

neliskulmaisen muodon reunoina (Kuva 4 esimerkissä tämä olisi 5 x 5). Suodatin kuvaa yksittäistä ikkunaa tarkastelevaa noodia. Askeleet kuvastaa kuinka monella noodilla suodattimet ovat etäällä toisistaan pysty- ja vaakasuunnassa. Esimerkiksi jos kaksi 5x5 suodatinta olisivat vierekkäin, niin että niiden vasemmat ylänurkat olisivat koordinaateissa 1x1 ja 2x1, olisi niiden askelluku 1.

3.1.4 Pooling-kerros (kasaus, yhdistys)

Usein konvoluutiota käyttävissä kuvatunnistusverkoissa on myös pooling- tai kasauskerroksia, jotka pakkaavat tulevan datan, jotta laskentatehon tarve tulevissa kerroksissa vähenee. Usein pakkausalgoritmi ottaa syötteestä 2x2 alueen ja palauttaa sen maksimiarvon, täten pienentäen seuraavan konvoluutiokerroksen kokoa. Tätä kutsutaan max pooliksi, mutta muitakin on, esimerkiksi average pool. Max pool on todettu käytännössä toimivimmaksi ratkaisuksi. Esimerkki Max Pooling-operaatiosta löytyy Kuvasta 6. (CS321N, n.d)



Kuva 6. Esimerkki Max pool operaatiosta (CS321N, n.d.)

3.2 **Neuroverkkojen toimintaperiaate ja aktivointifunktiot**

Neuroverkot toimivat reagoimalla aikaisemman kerroksen aktivointitasoihin, joita määrää aktivointifunktio ja yhteyden vahvuus noodien välillä (Tunnettu myös nimellä paino). Kaava 1 kertoo miten jokainen noodi laskee oman aktivointitasonsa. Lopullinen aktivointitaso ottaa syötteeksi jokaisen sille tulevan noodin aktivointitasot kerrottuna yhteyden painolla, laskee ne yhteen ja suorittaa saadulle tulokselle aktivointifunktion.

Aktivointitaso = aktivointifunktio((jokaista yhteyttä kohden)Toisen noodin aktivointitaso * yhteyden paino)

Kaava 1. Aktivointifunktion kaava

Seuraavan kerroksen noodilla on usein oma yhteytensä aikaisemman kerroksen noodeihin, ja jokaisella näillä yhteyksillä on oma painonsa. Jokainen noodi syöttökerroksen jälkeen laskee aikaisemman kerroksen aktivointitasojen ja painojen perusteella oman aktivointitason käyttäen aktivointifunktiota. (Gupta, 2017)

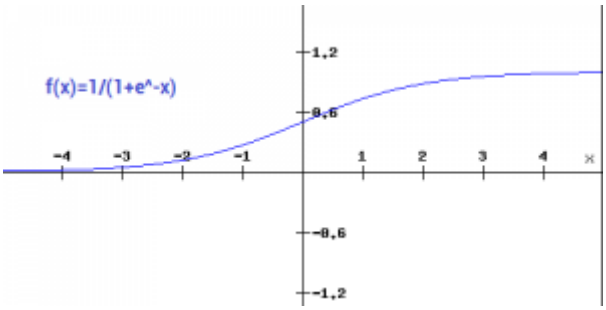
Neuroverkon kerrosten välisiä yhteyksiä on myös erilaisia, niin kutsuttu tiivis yhteys on kaikista simppelein, ja tarkoittaa että nykyisen kerroksen jokaisella noodilla on oma yhteytensä aikaisemman kerroksen jokaiseen noodiin. (Gupta, 2017)

Aktivointifunktiot neuroverkoissa ovat usein epälineaarisia, täten parantaen verkon kykyä oppia monimutkaisempia ongelmia. Ilman epälineaarisuutta verkko olisi rajattu vain lineaaristen ongelmien ratkaisemiseen. (Gupta, 2017)

3.2.1 Logistinen funktio (Sigmoid)

Logistinen funktio on funktio, jolla voidaan kuvata S-muotoinen käyrä. Käyrän minimin ja maksimin voi erikseen määrittää, mutta koneoppimiskäytössä se on usein 0 ja 1 välillä, Tämän kaltainen epälineaarinen arvo on hyödyllinen esimerkiksi esineiden tai asioiden tunnistuskäytössä, sillä todennäköisyys esineelle on usein nollan ja yhden välillä. Kuvassa 7 on funktion kuvaaja. (Gupta, 2017)

Logistisen funktion hyviä puolia on myös sen gradientti, se on sulava ja riippuvainen syötteen arvosta, joten Vastavirta-algoritmilla painojen muutto onnistuu helpommin. (Gupta, 2017)



Kuva 7. Logistinen aktivointifunktio (Gupta, 2017)

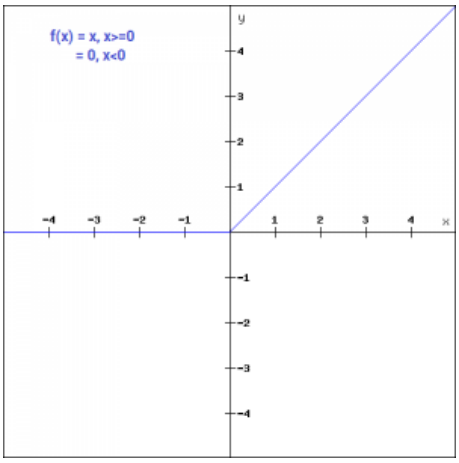
3.2.2 ReLu ja vuotava ReLu

ReLu-termi tulee sanoista Rectified Linear Units. ReLu ja sen eri versiot ovat yleisimmin käytettyjä aktivointifunktioita ja niitä kuvastaa kaava:

$$F(x) = \max(0,x)$$

Kaava 2. ReLu-aktivointifunktion kaava

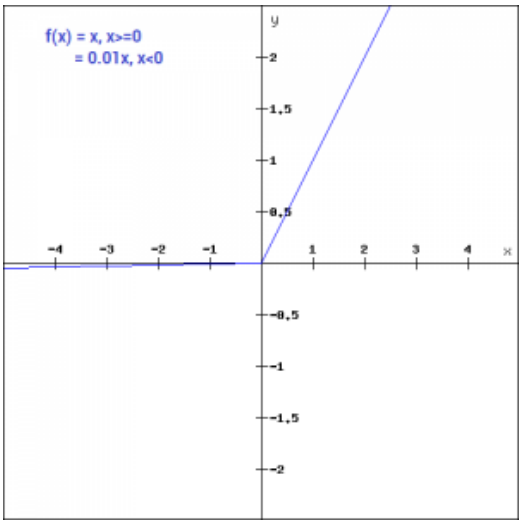
Eli jos aktivointitaso on negatiivinen, niin noodi ei aktivoidu ja muuten se aktivoituu saadun arvon mukaan. Tämä epälineaarisuus on hyödyksi Vastavirta-algoritmilla painojen päivityksessä. Kuvassa 8 on funktion kuvaaja. (Gupta, 2017)



Kuva 8. Normaali ReLu-funktio (Gupta, 2017)

Negatiivisella syötteellä aktivoimattomuudessa on omat hyödyt ja haittansa, esimerkiksi kaikki verkon noodit eivät välttämättä aktivoidu ollenkaan joistain syötteistä, joten niiden painoja ei vastavirrassa muuteta, eikä niitä tarvitse ottaa laskennassa huomioon. (Gupta, 2017)

Tämä tosin saattaa aiheuttaa tilanteen, jossa osa verkon noodeista ei hyödytä olemuksellaan mitään, tätä varten on kehitetty Vuotava ReLu. Tässä versiossa negatiiviset aktivointitasot ovat kerrottu esimerkiksi 0.01:llä, eivätkä vain nollattu, jolloin aktivointi negatiiviseen suuntaan tapahtuu, mutta paljon vähemmän kuin positiiviseen. Tämä riittää vähentämään riskiä kuolleista noodeista. Kuvassa 9 on funktion kuvaaja. (Gupta, 2017)



Kuva 9. Vuotava ReLu (Gupta, 2017)

3.3 **Neuroverkon koulutus**

Neuroverkot itsessään eivät tee paljoa mitään, vaan ne pitää kouluttaa tai mallintaa. Neuroverkot ovat funktioarvioinnin työkaluja ja voivat täten imitoida mitä tahansa tehtävää, johon ne koulutetaan. Monimutkaisemmat tehtävät vaativat täten monimutkaisemmat verkot. (ml4a, n.d.)

Neuroverkoissa on myös yleinen käsite painoista, eli noodien yhteydestä toiseen, mitä vahvempi yhteys kahden noodin välillä on, sitä herkemmin seuraavan kerroksen noodi aktivoituu syötteestä, ja tämä jatkuu aina tuloskerrokseen asti. Kouluttamisella pyritään näitä painoarvoja muuttamalla saada kokonaisuus imitoimaan mahdollisimman tarkasti haluttua lopputulosta. (ml4a, n.d.)

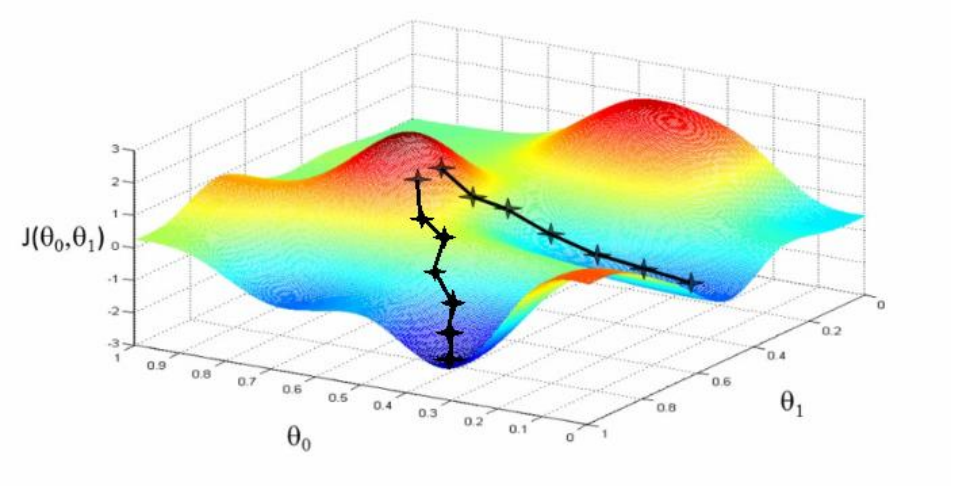
Kouluttaminen tapahtuu riippuen valitusta tehtävästä eri tavoin, mutta jokainen tarvitsee tarkkuuden, tai virheen laskemisfunktion (Loss function). Neuroverkon sen hetkistä sopivuutta haluttuun tehtävään arvioidaan tämän funktion avulla, joka pisteyttää kunkin verkon sen tekemän työn tarkkuuden perusteella. Kuten funktion englanninkielinen nimi kertoo, se pyrkii arvioimaan virhepisteitä. Mitä pienempi virhepisteytys, sitä paremmin verkko suoriutuu. Toisaalta on poikkeuksiakin kuten tehtävät, joissa parasta onnistumista ei tiedetä, jolloin pyritään maksimoimaan onnistumispisteytys . Tällaiset tehtävät ovat kuitenkin harvinaisia ja lähinnä onnistumispisteytyksen maksimointia on progressiivisissa peleissä ja ympäristöön reagoinnissa. (ml4a, n.d.)

Neuroverkon koulutukseen erittäin tärkeä konsepti on myös virhefunktion paikallinen ja globaali minimi. Globaali minimi olisi yhdistelmä painoarvoja, joilla verkko olisi tehtävässään paras mahdollinen, mutta tähän on erittäin vaikea päästä varsinkaan isoissa verkoissa. Paikallinen minimi taas on kohta, johon koulutuksessa usein päädytään. Siinä pienillä painojen muutoksilla havaitaan vain huonompia tuloksia. Usein koulutusmenetelmät pyrkivät pääsemään näistä pois ja etsimään parhaan mahdollisen virheminimin joko lisäämällä satunnaisuutta tai vaihtelemalla painoja enemmän etsien laajemmalta alueelta. (ml4a, n.d.)

3.3.1 Gradienttilaskeuma

Gradienttilaskeuma on usein ensimmäinen koulutustyyli, jolla neuroverkkoja ohjeistetaan kouluttamaan ja se on kieltämättä myös usein käytetty. Gradienttilaskeumassa aloitetaan verkon painot satunnaisesti, ja testataan mitä yksittäistä painoa siirtämällä päästään koulutusdataa käyttäen parhaiten kohti haluttua tulosta (Kuva 10). Tämä askel toistetaan

riittävän monta kertaa, kunnes päädytään funktiogradienttia laskemalla kohti paikallista virheminimiä. (ml4a, n.d.)



Kuva 10. Gradienttilaskeuman koulutuspolku (ml4a, n.d.)

Gradienttilaskeumassa lasketaan siis tarkkuusfunktion gradientti ja pyritään löytämään mahdollisimman matala virhepisteytys.

Perinteisen gradienttilaskeuman yleisimmät ongelmat on listattu Taulukossa 1.

Taulukko 1. Gradienttilaskeuman yleisimpiä ongelmia (ml4a, n.d.)

Termi	Selite
Satuloituminen	Virheen hidas laskeutuminen
Paikallinen minimi	Liian pieni paikallinen minimi, koulutus pysähtyy huonoon lopputulokseen
Nopeus	Koko koulutusdatan läpikäyminen jokaista kolutusoperaatiota varten on erittäin hidasta, sillä dataa voi olla satoja gigatavuja

Perinteisen Gradienttilaskeuman ongelmien lievitykseksi on kehitelty vaihtoehtoisia metodeja kuten stokastinen gradienttilaskeuma ja batch eli osaerä gradienttilaskeuma. Näissä versioissa käydään koko koulutusdatan sijasta läpi vain osia siitä, joko yksittäisiä (stokastinen) tai tasaisen kokoisiin pienempiin määriin jaettuja (osaerä) osia koulutusdatasta. Jokaisella testauskerralla satunnaisesti koulutusdatan vaihtaminen toiseen voi aluksi kuulostaa hullulta, mutta se auttaa pääsemään pois paikallisista minimeistä muuttamalla virhegradienttia. Tarpeeksi monen testauskerran ja koulutusmateriaalin jälkeen opetusdatan yhtäläisyydet silti ohjaavat tarkkuusfunktion gradientin kohti virheminimiä, ja täten kohti optimaalisia painovalintoja neuroverkolle. (ml4a, n.d.)

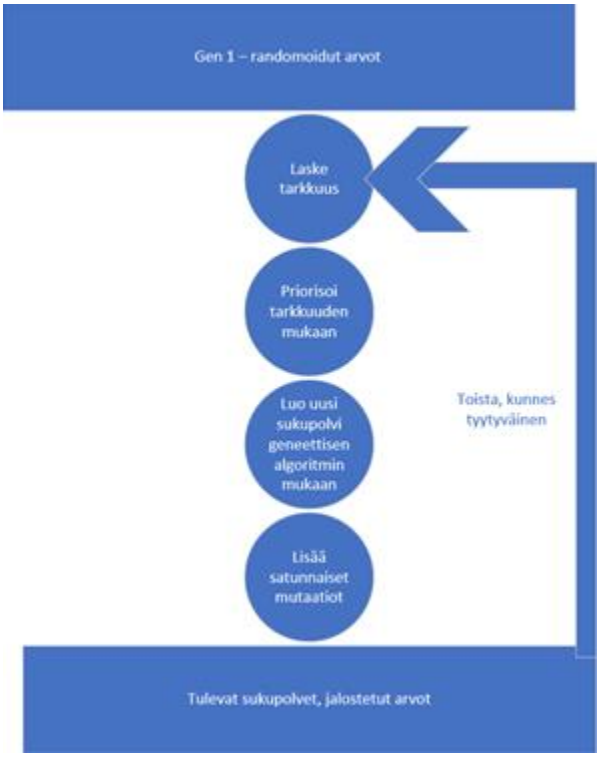
Viimeisenä vaiheena gradienttipohjaisissa koulutusmenetelmissä on derivaatan laskeminen, eli miten ja kuinka paljon jokainen neuroverkon painoarvo vaikuttaa lopputulokseen, usein tähän käytetään Vastavirta(backpropagation)-algoritmia. Tämä algoritmin selittäminen menee jo tämän opinnäytetyön tavoitteen yli, mutta lyhyesti selitettynä se sallii jokaisen painon vaikutuksen arvioinnin pienemmällä määrällä laskentatehoa, kuin erikseen jokaisen painon vaikutuksen laskemista koko neuroverkon lävitse. (ml4a, n.d.)

3.3.2 Geneettinen koulutusalgoritmi

Geneettinen koulutusalgoritmi on perinteisistä koulutusalgoritmeistä poikkeava. Sillä ne keskittyvät enemmän koulutuksen optimointiin ja aikaan. Geneettinen koulutusmenetelmä taas kouluttaa verkon biologisen evoluution tyyliin.
Ensin alustetaan sarja satunnaisia arvoja painoille ja näistä luodaan niin sanottu populaatio. Populaation eri jäseniä verrataan keskenään tarkkuusfunktion avulla ja tämän mukaan päätetään todennäköisyys olla osana seuraavaa kierrosta. (Harvey, 2017)

Seuraavalla kierroksella otetaan aikaisemman kierroksen painot ja onnistumiskyvyn kanssa painotetusta listasta valitaan satunnaisesti kaksi verkkoa, joiden painot yhdistetään biologisen yhtymisen tyyliin, molemmista edeltävistä verkoista otetaan puolet ja näistä yhdistetään seuraava jäsen. Populaation jäsenten määrä on usein loppuun asti sama, joten jokaista vanhaa jäsentä varten luodaan uusi. Tämän jälkeen jälkeläisiin voidaan vielä satunnaisgeneraattorin avustuksella luoda satunnaisia mutaatioita minimoimaan riski kehityksen hidastumiselle. Tämän jälkeen suoritetaan taas populaation tarkistus tarkkuusfunktiolla ja uudelle kierrokselle. (Harvey, 2017)

Vaikka tässä kappaleessa on puhuttu vain painojen muuttamisesta kouluttamisessa, niin koulutuksessa voidaan muuttaa myös itse verkon kokoa, kerrosten määrää ja noodien määrää kussakin kerroksessa, tähän tehtävään geneettinen algoritmi sopii erittäin hyvin, sillä muilla metodeilla oikean verkon koon etsiminen on huomattavasti työläämpää. Kuva 11 havainnollistaa geneettisen koulutusalgoritmin prosessia. (Harvey, 2017)



Kuva 11. Geneettisen algoritmin koulutusprosessi

Tarpeeksi monen sukupolven jälkeen verkko alkaa suoriutua tarkkuusfunktiosta paremmin ja paremmin, sillä parhaiten aikaisemmassa sukupolvessa suoriutuneet todennäköisemmin jakavat onnistuneet arvot eteenpäin. (Harvey, 2017)

Geneettinen koulutusmenetelmä ei täten vaadi vastavirta-algoritmia painojen hienosäätöön, sillä evoluutioprosessi hoitaa tämän itsestään tarpeeksi monen iteraation ja mutaatio kautta. (Harvey, 2017)

3.4 **TensorFlow**

Tensorflow on Googlen vuonna 2015 julkaisema sisäinen koneoppimisen ja nopean matriisimatematiikan laskentakirjasto. Tensorflow on erittäin suosittu tieteellisessä laskennassa ja neuroverkoissa sen tarjoamien työkalujen ja tehokkuuden vuoksi. (Tensorflow, n.d.)

Tensorflow toimii useilla eri alustoilla ja laskentaprosessoreilla, perusprosessoreilla, näytönohjaimilla ja jopa erityisillä neuroverkkojen laskentaan tarkoitetuilla laitteilla kuten ”Intel compute stick”. Koulutetut mallit voivat olla käytössä kevyissä päätelaitteissa kuten Raspberry Pi:ssa ja puhelimissa. (Tensorflow, n.d.)

Tensorflown ehdottomasti suurin hyöty on grafiikkakorttien laskentatehon hyödyntäminen neuroverkkojen mallinnusprosessissa. Näytönohjaimet voivat suorittaa matriisilaskuja tuhansia kertoja perusprosessoria nopeammin ja täten leikata koulutusajan murto-osaan. (Tensorflow, n.d.)

Tensorflown perusperiaate on käsitys noodeista, eli tensoreista, joista data *virtaa* kohti lopputulosta, tästä kirjaston nimikin tulee, tieto virtaa noodeista seuraavaan. Tensorflow sisältää myös valmiiksi useita koneoppimisessa käytettyjä oppimismetodeja ja aktivointifunktioita. Stokastinen- ja osaerä gradienttilaskeuma ovat suoraan tuettuja koulutuksen optimointimenetelmiä. (Tensorflow, n.d.)

3.5 **TensorFlow'n Layers-rajapinta**

Layers-rajapinta (Toiselta nimeltään Layers API) on TensorFlow'n tarjoama työkalu neuroverkkojen rakentamiseen. Layers-rajapinnan avulla voidaan luoda erimuotoisia neuroverkkoja määrittelemällä kerrokset, niiden toiminnallisuudet ja rakenteet. Sen avulla voidaan myös kouluttaa ja käyttää kyseisiä verkkoja.

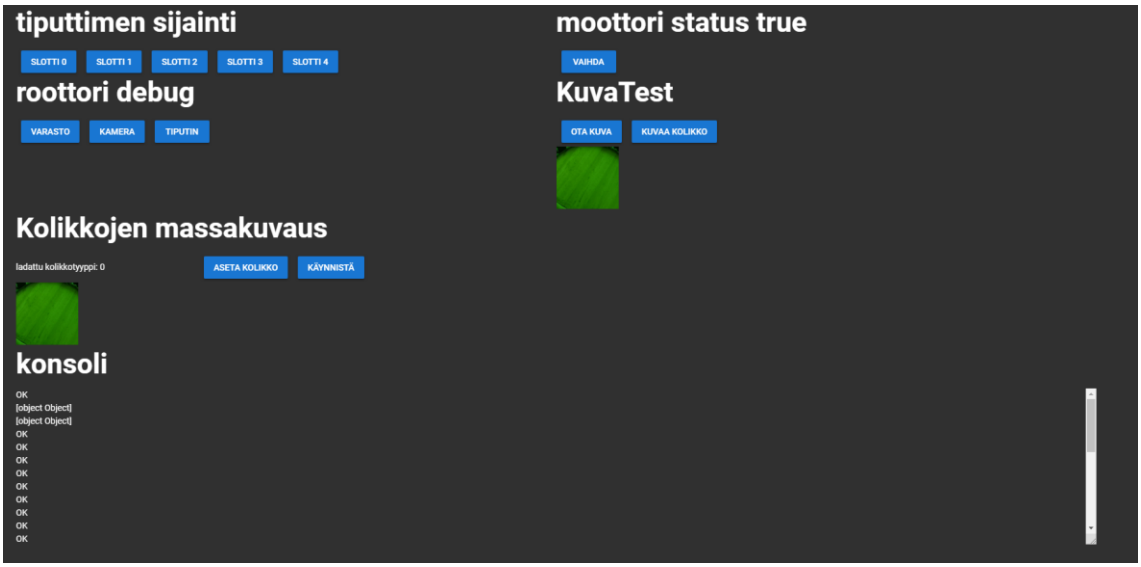
Layers API tekee hyvin helposti kaikki tässä opinnäytetyössä aiemmin käsitellyt neuroverkon ominaisuudet ja vaatimukset, kuten luvussa 5 tulee ilmi. (Tensorflow, n.d.a)

4 ROBOTILLE TEHDYT OHJELMAT

Opinnäytetyön robotissa on kaksi erillistä ohjelmaa, robotin ohjain ja koulutusympäristö. Robotissa käytettävästä neuroverkosta ja rakentamisesta on kerrottu enemmän luvussa 5.1 Mallien rakennus.

4.1 Robotin ohjain

Robotin ohjain on Raspberry Pi 3 - alustalla suoritettava node.js - sovellus, joka ohjaa robotin servoja, kameraa sekä tarjoaa kuvassa 12 nähtävän käyttöliittymän koulutusdatan keruulle ja toiminnnan seuraamiselle.



Kuva 12. Robotin käyttöliittymä

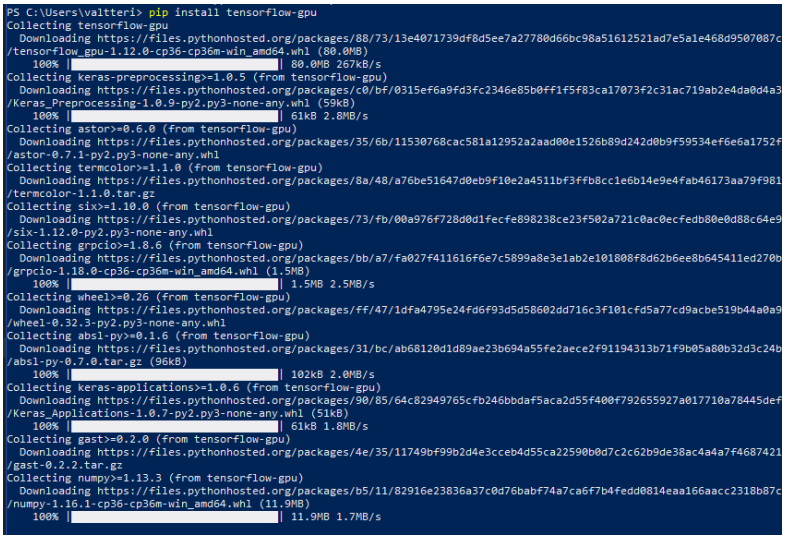
Robotin manuaalisilla ohjauksilla voi joko sammuttaa moottorit, keskeytttää suorituksen, testata kameraa tai liikuttaa moottoreita manuaalisesti, konsoli myös tulostaa kaikki potenttiaaliset virheviestit luettavaksi. Robotti ottaa kolikoista 100x100 resoluutiolla värikuvia ja käyttää näitä opetusmateriaalina sekä kolikon tunnistuksen pohjana.

Robotin päätoiminnallisuus on koulutetun mallin hyödyntäminen kolikoiden tunnistamisessa. Robotti automaattisesti kuvaa kolikoita ja tunnistaa ne mallin pohjalta. Tunnistuksen tulos näytetään käyttöliittymässä prosenttina kustakin kategoriasta. Jos tulos on joko virheellinen tai robotin luottamus omaan vastaukseen on liian matala, niin robotti keskeyttää suorittamisensa.

4.2 Koulutusympäristö

Koulutusympäristö on tehokkaalla pöytäkoneella suoritettava node.js-ohjelma, joka kouluttaa Tensorflo-mallia kerätyn koulutusdatan perusteella robotissa käytettäväksi.

Ensimmäinen asia on asentaa Tensorflow-gpu, joka on näytönohjaimella kiihdytetty Tensorflow, joka nopeuttaa koulutusta huomattavasti. Kuvassa 13 näkyy Tensorflow-gpu kirjaston asennusprosessi Windows-käyttöjärjestelmälle.



Kuva 13. Tensorflow-gpu asennus

Tämän jälkeen tarvitaan vielä noden windows-build-tools -pakkaus, jonka saa asennettua komennolla ’npm install --global windows-build-tools’ ylläpitäjän oikeuksin.

Tämän jälkeen koulutusta varten olevaan ohjelmaan voidaan lisätä Tensorflow-node paketti, joka käyttää aiemmin asennettua Tensorflow-gpu -ohjelmaa. Tämä asennetaan komennoilla ’npm install @tensorflow/tfjs-node’ ja ’npm install @tensorflow/tfjs-node-gpu’ node-projektissa.

Koulutusohjelma jaottelee saatavilla olevan koulutusdatan kahteen kategoriaan. Yksi on koulutusta varten, ja toinen tarkkuuden testaamista koulutuksen jälkeen. Jos koulutus ja testausdata on sama, niin riski mallin ylikoulutukseen on suuri. Ylikouluttamistilanteessa neuroverkko oppii tunnistamaan koulutusmateriaalin kuvat eikä niiden geneerisiä ominaisuuksia, jolloin se on uusien kolikoiden tunnistamiseen hyödytön.

Tämän jälkeen ohjelma alkaa määritetyn mallin kouluttamisen ja lopussa antaa valmiin Tensorflow-mallitiedoston robottiin siirrettäväksi. Mallien rakenne määritetään koulutusohjelman skriptissä layers apia käyttäen, Kuvassa 14 esiteltävässä ohjelmakoodissa luodaan uusi konvoluutiokerros neuroverkon ensimmäiseksi kerrokseksi.

```
model.add(tf.layers.conv2d({
  inputShape: [100, 100, 3], //kuvat on 100x100 RGB kuvia
  kernelSize: 2, //loput ovat oletuksia ohjeista
  filters: 16,
  strides: 4,
  activation: 'relu',
  kernelInitializer: 'VarianceScaling'
}))
```

Kuva 14. ohjelmakoodi, jolla tehdään syötekerroksen lisäys neuroverkon malliin

5 ROBOTIN KOULUTUS

Tässä kappaleessa käsitellään robotin verkon rakennus-, ja koulutusvaihe. Aluksi suunnitellaan koulutettavan verkon rakenne ja valitaan käytettävä koulutusalgoritmi, seuraavaksi kerätään ja järjestellään tarvittava koulutusdata ja lopuksi koulutetaan useampi neuroverkko ja verrataan tuloksia ja verkon rakenteista syntyviä eroja.

5.1 Mallien rakennus

Ensimmäinen vaihe uuden robotin tekoälyn kehityksessä on verkon rakenteen määritys. Rakenne sisältää määritykset verkon kerroksista, käytettävistä aktivointifunktioista, sekä tuloksesta.

Robotille kehitettiin testimielessä useampi erirakenteinen verkko, mutta koulutusmetodi oli kaikissa sama stokastinen gradienttilaskeuma.

Luvun 5.3 taulukossa 2 kuvataan kaikki testatut verkkorakenteet, koulutusdatan ja testausdatan lukumäärät sekä koulutusaika ja lopputulos. Yksittäisille tietueille annettiin myös omat nimet selkeyden ja erotettavuuden vuoksi.

Koulutusaika lasketaan koulutuksen alusta 99.9% tarkkuuteen, ja koulutus/testausdata määrittää kuvien määrän jokaisesta kategoriasta koulutus- ja testaustarkoitukseen.

5.2 Koulutusdatan keruu

Robotin kouluttaminen kolikoiden tunnistamiseen alkaa koulutusdatan keruusta, eli robotti käy läpi useamman tunnetun kolikon satunnaisissa asennoissa, jonka jälkeen kuvat tallennetaan. Tähän esimerkkiin koulutusdataa kerättiin noin 100-200 kuvaa per tunnistettava kolikkotyyppi. Kolikkotyyppejä ovat 10 senttiä, 20 senttiä, 50 senttiä, euro ja kaksi euroa.

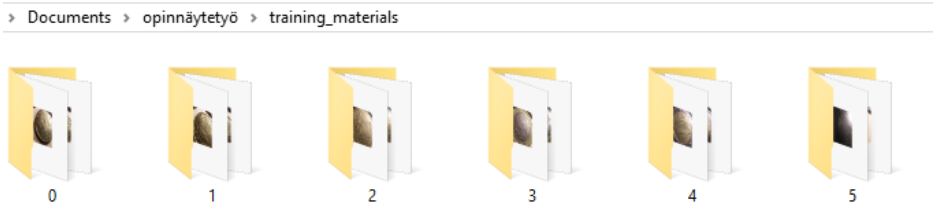
On erittäin tärkeää, että koulutusdata on tarkkaa ja sitä on laajasti. Neuroverkon kyky ratkaista ongelmia riippuu täysin sen koulutusdatan tarkkuudesta ja monipuolisuudesta. Kyseessä olevaa kolikonsuodatustehtävää varten tarkkoja tuloksia robotissa alkoi ilmaantua vasta noin 150 kuvan kohdalla jokaista järjestelyluokkaa kohden.

Kolikot kuvataan yhdeltä puolelta ja kategorisoidaan tämän perusteella. Neuroverkko siis joutuu oppimaan kolikon piirteet puolesta riippumatta, sillä robotissa ei voida varmistaa kumpi puoli on kameraa kohti.

Kolikoiden lisäksi koulutusdata sisältää myös virheellisiä kuvia, sillä robotin on hyvä tunnistaa sellaiset ja tarvittaessa sammuttaa itsensä laiterikkojen välttämiseksi. Tällaisia virhekuvia ovat esimerkiksi tyhjä kolikkopaikka ja pyörivä alusta väärässä asennossa.

Kerätyt kuvat on järjestetty omiin ryhmiinsä, jotka on esitelty kuvassa 16. Ryhmässä 0 on 10 sentin kolikot, ryhmässä 1 on 20 sentin kolikot, ryhmässä 2 on 50 sentin kolikot, ryhmässä 3 on euron kolikot, ryhmässä 4 on kahden euron kolikot ja ryhmässä 5 on virheelliset. Kuvassa 15 robotti on keräämässä kuvia kolikoista.

Kuva 15. Koulutusdatan keruu (10snt)



Kuva 16. Koulutusdata

Robotin ohjaimessa on automaattinen ominaisuus, joka ottaa ja tallentaa kuvat tietyn tyyppisistä kolikoista. Tällä toiminnolla käyttäjä voi tallentaa kolikkoryhmistä kuvia antamalla ryhmän tunnuksen ja syöttämällä robottiin ryhmään kuuluvia kolikoita. Laite kerää kuvia kolikoista niin kauan kunnes se keskeytetään. Otetut kuvat voidaan sitten siirtää robotin sd-kortilta koulutusympäristöön, käydä pikaisesti läpi ja varmistaa datan oikeellisuus. Tässä vaiheessa myös mahdolliset virhetilannekuvat uudelleenkategorisoidaan oikein. Kuvien järjestelyn ja varmistamisen jälkeen niiden pohjalta voidaan haluttu neuroverkko kouluttaa.

5.3 **Mallin koulutus**

Kun kuvat ovat tallennettu voidaan siirtyä seuraavaan vaiheeseen, eli kuvien ajamiseen koulutusympäristössä. Koulutusvaihe kestää usein pitkään, sillä dataa joudutaan käymään läpi useasti gradienttilaskeuma-algoritmin aikana. Koulutuksen valmistuttua koulutettu malli voidaan siirtää robottiin ja testata satunnaisten kolikkojen kanssa.

Ennen koulutusta pitää päättää verkon rakenne, tämä sisältää tiedon syötteen koon, joka robotin tapauksessa on 100x100x3 kuva (RGB), välikerrokset sekä tuloskerroksen. Verkon tarkalle rakenteelle ei ole suoraa määritystä tai ohjetta joten se usein suoritetaan arvioimalla ja kokeilemalla. Kolikkorobottia varten suunniteltiin ja touteutettiin kolme eri muotoista verkkoa. Kuten aikaisemminkin on mainittu, isommat verkot voivat analysoida ja oppia monimutkaisempia ongelmia, mutta niiden koulutus kestää huomattavasti kauemmin. Täten verkon kokoa koetetaan minimoida vastaamaan tehtävän vaatimusta. Taulukossa 2 esitetään robotin koulutettujen neuroverkkojen rakenteet ja koulutusajat. Kuvassa 17 näkyy ensimmäisen verkon koulutusprosessin loppu.

Rakennenotaatiossa muuttujat ovat suorat syötteet Tensorflow'n Layers rajapinnalle.

- Konvoluutiokerros merkitään tyylillä *C(koko, askeleet, lkm, aktivointi*).
- Litistyskerros on L.
- Tiivis kerros on T(koko, aktivointi).
- Pooling, eli kasauskerros on P(koko, askeleet).

Konvoluutiokerroksen notaatiossa koko tarkoittaa suodattimen neliön muotoisen ikkunan reunan pituutta. Esimerkiksi 5 tarkoittaisi 5x5 ikkunaa. Tiivis kerros on TensorFlown perinteinen feed-forward kerros, jossa jokainen noodi on yhteydessä kaikkiin aiemman kerroksen noodeihin. Litistyskerros on ennen Tiiviitä kerroksia vaadittu toiminto, jossa moniulotteinen data muutetaan yksiulotteiseksi. Moniulotteinen data voi olla esimerkiksi konvoluutiokerroksen suodattimet tai RGB-kuva syötekerroksessa, kaikki nämä kerrokset puretaan yhdeksi noodijonoksi johon tiivis verkko voidaan yhdistää.

Taulukko 2. Koulutetut neuroverkot, koulutusasetukset ja tulokset

ID	Rakenne	koulutusaika	Koulutus-/testausdata kappalemäärät
1	C(2,4,16,ReLu)-> C(2,8,32,ReLu)-> C(2,4,64,ReLu)->L->T(30, softmax)->T(6, linear)	1h 10min	577/577
2	C(10,2,32,ReLu)-> P(2,2)->C(5,2,32,ReLu)->	-epäonnistui (6h+ ja tarkkuus alle 20%)	577/577

	p(2,2)->C(2,2,64,ReLu)->L->T(6, softmax)		
3	C(2,2,32,ReLu)->P(2,2)->C(2,8,32,ReLu)->C(2,4,64,ReLu)->L->T(32, softmax)->T(6, linear)	3h 13min	577/577

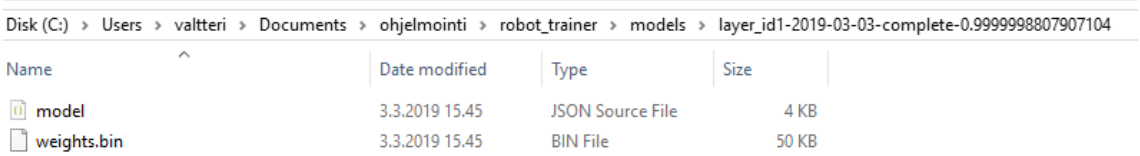
```
01:12:47 - total accuracy: 0.9799998998641968 - trainiing batch: 159 - 499/500 epoch ended
loss:  0.0002479091053828597 number 0.0002479091053828597
accuracy 0.9999998807907104 number
best accuracy 0.9999998807907104
--------------------end of loop 159----------------------
training done, doing debug:
actual labels [ 4, 2, 4, 2, 1 ]
predictions [ 4, 2, 4, 2, 1 ]
training data num 577
testing data num 577
PS C:\Users\valtteri\Documents\ohjelmointi\robot_trainer>
```

Kuva 17. Neuroverkon (ID 1) koulutusohjelman lopputulos

Tuloksista selviää, että verkon rakenteen pienelläkin muutoksella on suuria vaikutuksia sen koulutusvaikeuteen ja soveltuvuuteen haluttuun tehtävään, esimerkiksi verkoilla 2 ja 3 on vain hyvin pieniä rakenteellisia eroja, mutta lopputulos on aivan erilainen. Parhaiten tehtävästä silti suoriutui verkko 1, lopputulos oli tarkka ja koulutusaika suhteessa muihin erittäin lyhyt. Pienet erot koulutusajassa eivät ole merkittäviä, sillä prosessi sisältää aina jonkin verran satunnaisuutta valitun koulutus- ja testausdatan vuoksi. Stokastisessa gradienttilaskeumassa käytettävä data vaihtelee satunnaisesti, joten saman mallin koulutusnopeus voi eri ajokerroilla vaihdella huomattavasti. Toisaalta suuret erot kuten verkkojen 1 ja 3 välillä, joissa ero on melkein kolminkertainen, ei voida yksinään pistää satunnaisuuden varaan.

5.4 Mallin testaus robotissa

Mallin koulutusohjelma tuottaa jokaista verkkoa kohden oman kansion, joka sisältää kaksi tiedostoa: verkon rakennekuvauksen ja yksittäisten yhteyksien painot (Kuva 18). Tämän kansion voi siirtää robotin ohjaimeen, jonka jälkeen robotti käyttää kyseistä verkkoa kolikoiden järjestelyyn.



Kuva 18. Koulutetun neuroverkon tallennetut tiedot

Mallin testaus robotissa tapahtuu syöttämällä satunnaisia kolikoita robottiin ja antamalla sen järjestää ne omiin lokeroihinsa ottamalla niistä kuvan ja hyödyntämällä sille annettua neuroverkkoa. Lopullinen verkon toimivuus ja tunnistustarkkuus siis on ihmisvalvojan katsottavissa. Mahdollisesti nämäkin kolikot voitaisiin kuvata ja tallentaa ennusteidensa kanssa mahdolliseksi jatkokoulutustiedoksi jos tarkkuutta joskus haluttaisiin parantaa.

Mallit 1 ja 3 testattiin robotissa, ja ne toimivat erinomaisesti ilman virheellisiä tunnistuksia. Jopa virhetilan tunnistus ja automaattinen sammutus toimivat moitteetta. Robotti tunnisti kolikot oikein myös vaihtuvissa valotuksissa, joita koulutusmateriaali ei sisältänyt.

6 YHTEENVETO JA TULOKSET

Yhteenvetona voidaan todeta, että työ onnistui tavoitteessaan kouluttaa neuroverkko tunnistamaan kolikoita, ja ajamaan kyseistä verkkoa kevyessä päätelaitteessa (tässä tapauksessa Raspberry Pi). Lisätyöllä ja ajalla verkon rakenteen tutkimista olisi mahdollisesti voinut jatkaa ja optimoida koulutusajan lyhentämiseksi, sekä lisäominaisuuksien kehittämistä kuten jatkokouluttamista uusille kolikkotyypeille tai tarkkuuden lisäystä uudella datalla.

Omasta kokemuksesta neuroverkkojen sovellus muuten ohjelmallisesti hankalan ongelman ratkaisemiseksi oli suhteellisen helppo operaatio. Neuroverkot olivat kolikoiden kuvapohjaiseen tunnistukseen erittäin hyvä ja helppo työväline. Näen että tämä teknologia on erittäin hyödyllinen hankalasti kuvailtavien ongelmien kuten kuvantunnistuksen työväline.

LÄHTEET

Adeshpande3. (n.d.) *Beginner's guide to understanding convolutional neural networks*. GitHub. Haettu 6.2.2019 osoitteesta https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Bromley, T. (2018). *Making a Neural Network, Quantum.* Medium. Haettu 28.1.2019 osoitteesta https://medium.com/xanaduai/making-a-neural-network-quantum-34069e284bcf

Colah. (n.d.) *Colah's blog: understanding LSTM networks*. GitHub. Haettu 6.2.2019 osoitteesta http://colah.github.io/posts/2015-08-Understanding-LSTMs/

CS321n. (n.d.) *CS231n Convolutional Neural Networks for Visual Recognition*. GitHub. Haettu 6.2.2019 osoitteesta http://cs231n.github.io/convolutional-networks/

Gupta, D. (2017) *Fundamentals of deep learning – Activation Functions and When to Use Them?.* Analytics Vidhya. Haettu 28.1.2019 osoitteesta https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/

Harvey, M. (2017). *Let's evolve a neural network with a genetic algorithm.* coastline automation. Haettu 28.1.2019 osoitteesta https://blog.coast.ai/lets-evolve-a-neural-network-with-a-genetic-algorithm-code-included-8809bece164

Ml4a. (n.d.) How neural networks are trained. GitHub. Haettu 28.1.2019 osoitteesta https://ml4a.github.io/ml4a/how_neural_networks_are_trained/

Tensorflow. (n.d.) *TensorFlow*. Haettu 28.1.2019 osoitteesta https://www.tensorflow.org/

Tensorflow. (n.d.a) *Training on images: Recognizing Handwritten Digitswith a Convolutional Neural Network.* Haettu 6.2.2019 osoitteesta https://js.tensorflow.org/tutorials/mnist.html