OULUN AMMATTIKORKEAKOULU

Valeri Haataja

KONENÄÖN OPETTAMINEN

KONENÄÖN OPETTAMINEN

Valeri Haataja
Opinnäytetyö
Kevät 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, Ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Valeri Haataja
Opinnäytetyön nimi suomeksi: Konenäön opettaminen
Opinnäytetyön nimi englanniksi: Teaching Machine Vision
Työn ohjaaja: Jukka Jauhiainen
Työn valmistumislukukausi ja -vuosi: Kevät 2021
Sivumäärä: 41

Tämän opinnäytetyön tavoitteena oli tutkia konenäköjärjestelmää ja etsiä tunnistustarkkuutta parantavia tekijöitä. Konenäköjärjestelmä oli tarkoitettu työkoneiden osien tunnistamiseen varaosien tilaamisen helpottamiseksi ja nopeuttamiseksi. Tarve työlle tuli, kun järjestelmää oltiin ottamassa asiakaskäyttöön, mutta tunnistustulokset olivat olleet heikkoja.

Työ tehtiin tutkimalla konenäköjärjestelmän toimintaa ja etsimällä tekijöitä tunnistustarkkuuden parantamiseksi. Ensin työssä käsitellään konenäköön liittyviä aiheita, sekä kerrotaan järjestelmän toiminnasta. Tutkimista varten tehtiin paikallinen testiympäristö, jossa konenäön pystyi opettamaan ja testaamaan tarkkuutta monella tavalla.

Työn lopputuloksena saatiin tutkittua, testattua ja dokumentoitua konenäön opettamisen ja testauksen vaiheet sekä tulokset. Testien tuloksena pystyttiin toteamaan konenäköjärjestelmän toimivan hyvin, kun opetusdata on tarpeeksi laadukasta. Kuvien tunnistus parani huomattavasti rajaamalla häiriötekijöitä taustalta pois, joten jos kuvien laatu olisi standardisoitua, niin silloin tunnistustarkkuus saataisiin varmistettua.

Asiasanat: koneoppiminen, konenäkö, Docker-kontti, testaus

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Information Technology, Software Development

---

Author: Valeri Haataja
Title of thesis: Teaching Machine Vision
Supervisor: Jukka Jauhiainen
Term and year when the thesis was submitted: Spring 2021
Pages: 41

The goal of this thesis was to find factors from the machine vision system that could improve system's image classification accuracy. The machine vision system was designed to identify parts of work machines and that way to make the ordering spare parts easier and faster. The need for the thesis arose when the system was being introduced to customers, but image recognition accuracy results had been poor.

The work was done by examining the operation of the system and looking for factors to improve the recognition accuracy. This document deals with topics and concepts related to the machine vision and describes the operation of the system. Machine vision was taught and tested in many ways in the local test environment.

As a result of the work was this document which describes the steps of teaching and testing the machine vision system and the test results. Image recognition was considerably improved by delimiting interference factors from the image. By standardizing the quality of the images would ensure the accuracy of the image recognition.

Keywords: machine learning, machine vision, Docker container, testing

ALKULAUSE

Työn ohjaavana opettajana toimi Oulun ammattikorkeakoulun tietotekniikan yliopettaja Jukka Jauhiainen. Työn toimeksiantajan puolelta ohjaajana toimi projektipäällikkö Matti Asumaniemi.

Haluan kiittää Finlabsia mahdollisuudesta tämän opinnäytetyön tekemiselle. Haluan myös kiittää työn ohjaajia sekä insinöörejä Juha-Matti Tirilää ja Mika Lackmania teknisestä tuesta.

Oulussa 11.5.2021

Valeri Haataja

SISÄLLYS

SANASTO

API	Application Programming Interface, ohjelmistoissa käytettävä rajapinta.
AWS	Amazon Web Services, Amazonin kokoelma etätietojenkäsittelyresurssien palveluita.
Base64	Enkoodausmenetelmä, jota käytetään datan muuttamiseen merkkijonoiksi ja takaisin alkuperäiseksi dataksi.
Malli	Koneoppimismalli, tiedosto, joka on opetettu tunnistamaan datasta kuvioita ja piirteitä.
REST	Representational State Transfer, arkkitehtuurimalli rajapintojen toteutukseen, joka perustuu HTTP-protokollaan.
Skripti	Tietokoneohjelma, joka on kirjoitettu jollain komentokielellä ja suoritetaan ajonaikaisesti.
S3	Amazon Simple Storage Service, pilvitallennusalusta.

1 JOHDANTO

Opinnäytetyö tehtiin Finlabsille vuoden 2021 kevään aikana. Finlabs on konserni, joka on perustettu vuonna 2017. Siihen kuuluu emoyhtiö Financial labs Oy ja tytäryhtiö Labrats Oy. Yritys tekee laaja-alaisesti design-lähtöisiä ohjelmistoratkaisuja kotimaisille sekä kansainvälisille yrityksille. Finlabsilla on perustamisesta saakka ollut New Yorkissa työntekijä, mutta vuonna 2020 virallisesti perustettiin yritys Yhdysvaltoihin, jossa työskenteli työn tekohetkellä neljä työntekijää. Yhteensä työntekijöitä oli vuoden 2021 alkupuolella noin 40, joista suurin osa Oulussa. Finlabsin toimistot sijaitsevat Oulussa, Helsingissä sekä New Yorkissa. (1; 2.)

Projektin yhteistyökumppani Green Master on oululainen vuonna 2016 perustettu yritys, joka tilaa suomalaisten tarpeisiin työkoneita kiinalaiselta tehtaalta. Mallistoon kuuluvat pyöräkuormaajat, kaivinkoneet ja trukit. (3.)

Opinnäytetyön aihe syntyi, kun Finlabs oli kehittänyt konenäköjärjestelmän Green Masterin työkoneiden osien tunnistamiseen, mutta tunnistustarkkuus oli ollut heikkoa. Opetusdata eli opetettavat kuvat osista otetaan ja lähetetään mobiilisovelluksella, joka lähettää kuvat pilvitallennuspalveluun ja sitä kautta järjestelmälle. Tunnistettavat kuvat käyttäjän toimesta lähetetään WhatsApp-keskustelusovelluksella ja lähetetty kuva menee tunnistettavaksi konenäköjärjestelmään. Jos osan tunnistus onnistui, WhatsApp-keskusteluun tulee suora linkki Green Masterin verkkokauppaan, josta osan voi tilata. Opinnäytetyön aloittamishetkellä järjestelmä oli siis jo olemassa, mutta ongelmana oli järjestelmän tuottamien tunnistustulosten epätarkkuus.

Työn tavoitteena oli tutustua järjestelmän toimintaan ja etsiä sekä testata tapoja, jotka parantaisivat konenäön tunnistustarkkuutta. Opinnäytetyön tekijällä ei ollut aikaisempaa kokemusta tämänkaltaisista järjestelmistä.

2 TEKOÄLY

Tässä luvussa käydään läpi tekoälyn teoriaa, termejä sekä tekoälyn osa-alueiden eroavaisuuksia, jotka liittyvät oleellisesti tämän opinnäytetyön aiheeseen.

Termi ”tekoäly” eli Artificial Intelligence (AI) tarkoittaa yleisesti koneita tai ohjelmia, jotka pystyvät suoriutumaan tehtävistä, joihin yleensä tarvitaan ihmisen älykkyyttä (4). Näitä tekoälyn tehtäviä ja kykyjä, jotka perinteisesti liitetään ihmisen älyyn, ovat muun muassa päättely, ongelman ratkaisu, esineiden tunnistaminen, kieleen reagointi ja oppiminen (5).

Tänä päivänä käytetty tekoäly on heikkoa tekoälyä, josta käytetään myös termiä kapea tekoäly (5). Heikko tekoäly on tekoälyn muoto, jossa ohjelma tai kone keskittyy kapeaan tehtävään ja seuraa sille asetettuja sääntöjä eikä voi poiketa niistä (6). Yksi esimerkki heikosta tekoälystä on shakkia pelaava tekoäly, jossa pelin siirrot perustuvat aiemmin määritettyihin sääntöihin. Tekoäly ei siis itsessään tiedä shakista mitään, vaan se suorittaa siirrot tilanteen analysoinnin jälkeen sen mukaan, miten se on ohjelmoitu käyttäytyvän. (7.)

Vahva tekoäly on lähempänä ihmismieltä olevaa tekoälyä. Siinä tekoäly pystyy itsenäisesti ajattelemaan ja tekemään päätöksiä ilman ihmisen puuttumista asiaan. Vahva tekoäly on kuitenkin vain teoreettista, eikä sellaista ole ainakaan vielä kehitetty. (5.)

2.1 Koneoppiminen

Tekoälyn yksi keskeisin osa-alue on koneoppiminen (4). Koneoppimisessa tietojärjestelmä opetetaan hyödyntämään algoritmeja ja havaitsemaan kuvioita, jotka toistuvat datassa. Esimerkiksi sähköpostin suodatus, hakukoneen suositukset, konenäkö ja kyky tunnistaa puhetta ovat tekoälyä, joka hyödyntää koneoppimista. (8; 9.)

2.1.1 Ohjattu oppiminen

Ohjattu tai valvottu oppiminen tarkoittaa sitä, kun konetta opetetaan lisäämällä dataan jonkinlaisia tunnistetietoja. Esimerkiksi jos data sisältää valokuvia ja niihin

on lisätyt tunnisteet kertomaan, mitä kuva esittää, algoritmi kerää nämä tunnisteet pois datasta. (8.)

2.1.2 Ohjaamaton oppiminen

Jos koneet tunnistavat yhtäläisyyksiä datasta ilman tunnisteiden käyttöä, puhutaan valvomattomasta tai ohjaamattomasta oppimisesta. Tällöin algoritmi etsii ja ryhmittelee yhteen esimerkkejä, joissa on samankaltaisuutta, ilman algoritmin ohjelmointia tiettyjen datalajien havaitsemiseen. (8.)

2.1.3 Vahvistava oppiminen

Vahvistavaksi oppimiseksi kutsutaan sitä, että kone oppii yrityksien ja virheiden kautta ja osaa lopulta päättää parhaan tavan tehtävän suorittamiseen. Esimerkiksi videopelissä tekoälyllä varustettu pelihahmo voi pystyä oppimaan pelikentän vaarallisia paikkoja ja väistää ne. (8.)

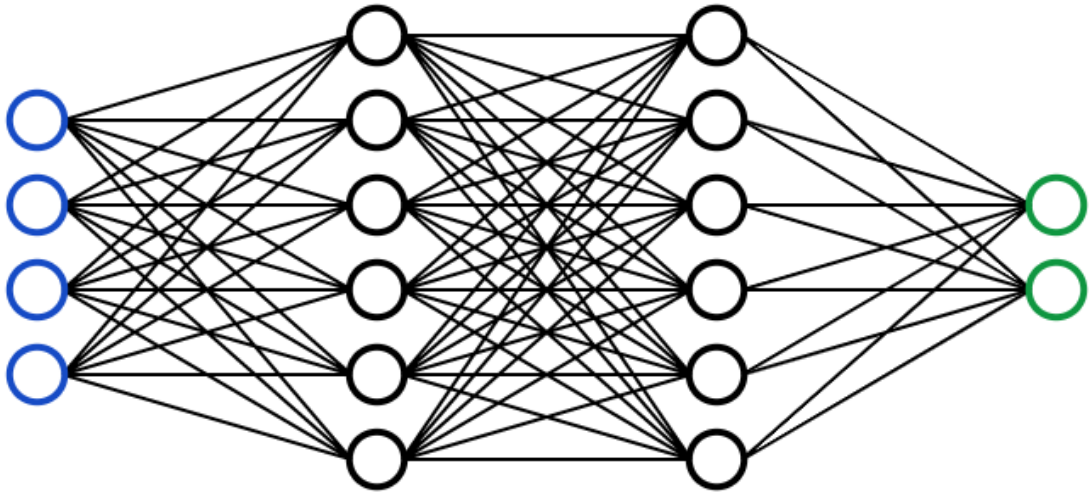
2.2 Malli

Koneoppimismalli on tiedosto, joka on opetettu tunnistamaan tietynlaisia kuvioita ja piirteitä. Malli opetetaan datasettiä eli opetusdataa vasten ja tarjotaan mallille algoritmi, jonka avulla se oppii annetusta datasta. Kun malli on opetettu, sitä voidaan käyttää käymään läpi dataa, mitä se ei ole vielä nähnyt, ja tekemään niin sanottuja ennustuksia uudesta datasta. (10.)

2.3 Neuroverkot

Neuroverkko on matemaattisten yhtälöiden verkko. Se koostuu kerroksista, joissa on solmuja eli keinotekoisia neuroneja. Verkon ensimmäinen kerros on sisääntulokerros, välissä piilokerroksia ja viimeisenä on ulostulokerros. (11.) Neuroverkko vaatii yhden tai useamman tulomuuttujan (input) ja yhtälöverkon läpikäydessä saadaan yksi tai useampi lähtömuuttuja (output) (13). Jokainen neuroverkon solmu suorittaa jonkinlaisen laskennan, joka välitetään muille, syvemmällä oleville verkon solmuille (11).

Kuvassa 1 nähdään yksinkertainen esimerkki neuroverkosta. Siinä siniset ympyrät kuvaavat tulokerrosta, mustat ympyrät piilokerroksia ja vihreät lähtökerrosta (13).



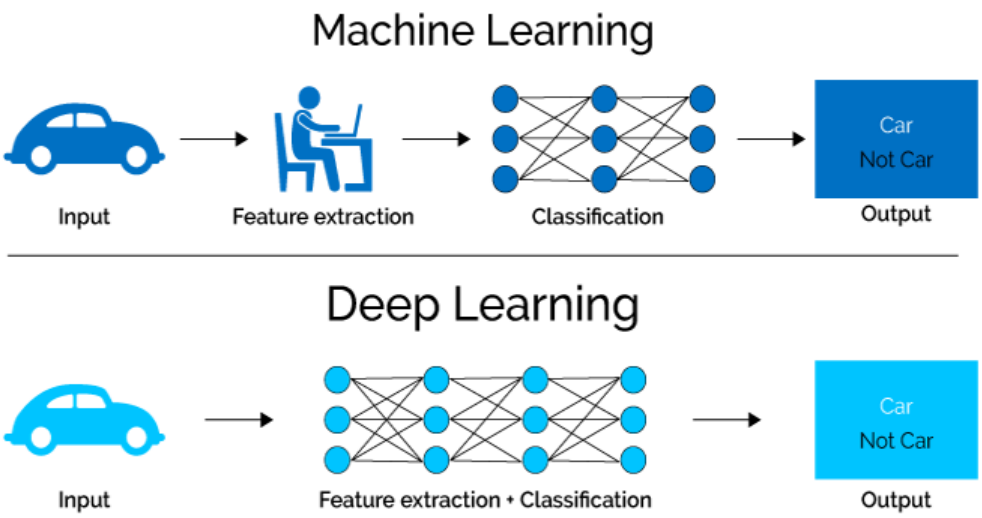
KUVA 1. Visualisointi neuroverkosta (11)

2.4 Syväoppiminen

Syväoppiminen on koneoppimisen osa-alue, jossa monikerroksiset neuroverkot oppivat suurista datamääristä, ja sen rakenne jäljittelee ihmisen aivojen rakennetta. Syväoppimisalgoritmit yrittävät saada samankaltaisia havaintoja kuin ihmiset, analysoimalla dataa annetun logiikan perusteella. Tätä varten syväoppimisessa käytetään monikerroksisia neuroverkkoja. (12.)

Koneoppiminen ja syväoppiminen eroavat toisistaan siten, että syväoppimisessa algoritmi ei tarvitse niin sanottua piirreirrotusta (feature extraction). Tämä tarkoittaa sitä, että perinteisessä koneoppimisjärjestelmässä ihmiset joutuvat määrittelemään koneoppimismallille erilaisia ominaisuuksia. Esimerkiksi jos halutaan, että koneoppimismalli tunnistaisi kuvasta auton, ihminen joutuu määrittelemään auton yleisiä piirteitä (muoto, värit, ikkunat, renkaat yms.) algoritmille inputdatana. Aivan kuten ihminen vertaa näkemäänsä objektia aiemmin nähtyyn samankaltaiseen objektiin ja sen ominaisuuksiin. Syväoppimisessa malli tunnistaa samankaltaisuuksia datasta ja tekee sen perusteella ennustuksia eli tässä esimerkissä, onko kuvassa auto vai ei. (12.)

Kuvassa 2 nähdään, että syväoppimisessa feature extraction -vaihe on osa prosessia, jossa tapahtuu myös tunnistaminen.



KUVA 2. Koneoppimisen ja syväoppimisen eroavaisuus (12)

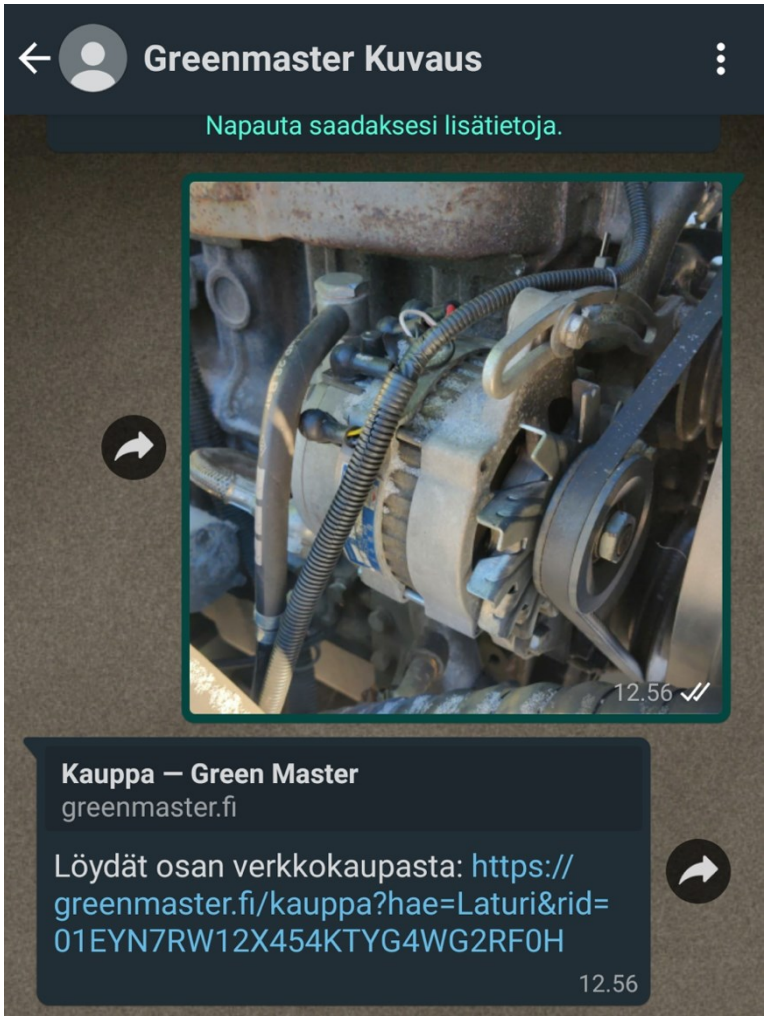
3 KONENÄKÖJÄRJESTELMÄ

Tässä luvussa käydään läpi työn kohteena olevan konenäköön perustuvan varaosien tunnistusjärjestelmän toimintaperiaatetta ja teknologioita, joita siihen oli käytetty. Aiheet liittyvät opetusdatan tuottamiseen ja muihin relevantteihin asioihin opinnäytetyön kannalta.

3.1 Lähtötilanne

Aloittaessani opinnäytetyön tekemisen konenäköratkaisu oli jo muuten toimiva kokonaisuus, tunnistustarkkuuden heittelyä lukuun ottamatta. Järjestelmä oli tarkoitettu Green Masterin työkoneiden osien tunnistamiseen, jotta osan tunnistaminen ja varaosan tilaaminen olisi helpompaa ja resursseja säästävämpää. Järjestelmän perimmäinen tarkoitus oli siis säästyä aikaa vievältä ihmisten väliseltä edestakaiselta kommunikoinnilta ja antaa koneen hoitaa osan tunnistaminen. Näin asiakas voisi tilata osan suoraan verkkokaupasta ilman pitkien katalogien selaamista.

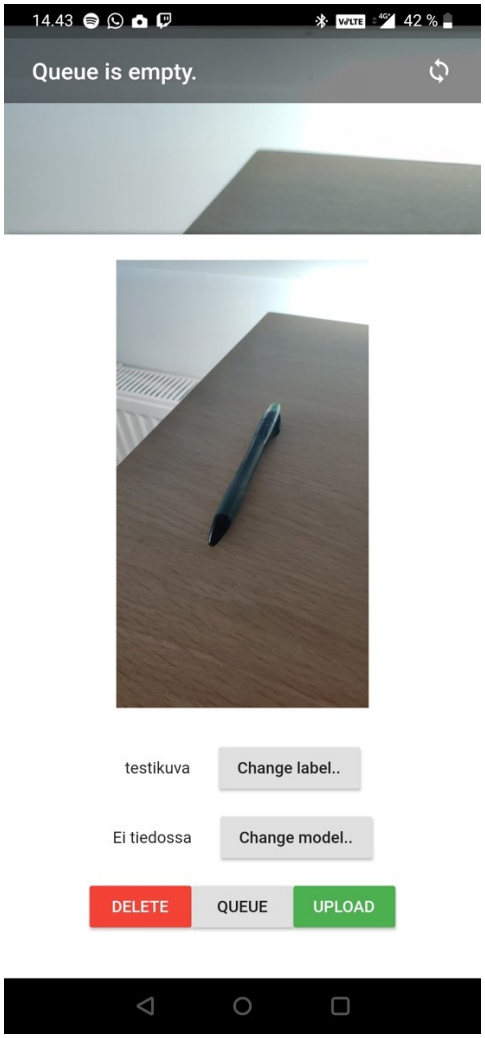
Järjestelmän toiminta menee loppukäyttäjän näkökulmasta näin: Kun työkoneen osa menee rikki ja sen tilalle tarvitaan varaosa, osasta otetaan kuva WhatsApp-keskustelusovelluksella ja lähetetään se tiettyyn numeroon. Lähetetty kuva menee konenäköohjelmaan, jossa ohjelma yrittää tunnistaa kuvassa esiintyvää osaa. Jos tunnistaminen onnistui, vastauksena tulee linkki Green Masterin verkkokauppaan, josta pääsee helposti selaamaan kyseisiä varaosia. Kuvassa 3 on kuvankaappaus WhatsApp-keskustelusta, jossa on lähetetty kuva opetetusta kategoriasta. Kuvassa on laturi ja keskusteluun on tullut linkki verkkokauppaan, josta pääsee selaamaan myynnissä olevia latureita. Tämä on nähtävissä palautuneen linkin URL-osoitteessa.



KUVA 3. Vastaus tunnistetulle kuvalle WhatsApp-keskustelussa

3.2 Opetusdatan tuottaminen

Konenäköjärjestelmän opetusmateriaali tuotetaan käyttämällä Finlabsin kehittämää mobiilisovellusta. Sillä voidaan kuvata osa joko ottamalla osasta lyhyt video tai valokuva, jonka jälkeen sovelluksessa annetaan kuvalle tunniste eli label. Tunniste annetaan osien kategorisoimista varten. Tämän jälkeen opetusdata lähetetään Amazon Web Sevice S3 -pilvitallennuspalveluun (23). Kuvassa 4 on kuvankaappaus opetusdatan tuottamista varten tehdystä mobiilisovelluksesta.



KUVA 4. Mobiilisovellus opetusmateriaalin tuottamista varten

Mobiilisovellus on siis kamerasovellus, jolla osasta voidaan ottaa opetusdataksi kuvia ja lähettää ne pilvipalveluun. Pilvestä ne saadaan ladattua konenäköjärjestelmälle, jonka voi opettaa tunnistamaan osia annetun tunnisteen perusteella. Kuville on tunnisteen lisäksi mahdollista syöttää työkoneen malli, jos se on tiedossa. Tämän avulla voidaan rajata varaosalistaa niin, että se näyttää vain varaosat, jotka käyvät kyseiseen työkoneeseen.

3.3 Konttiteknologia

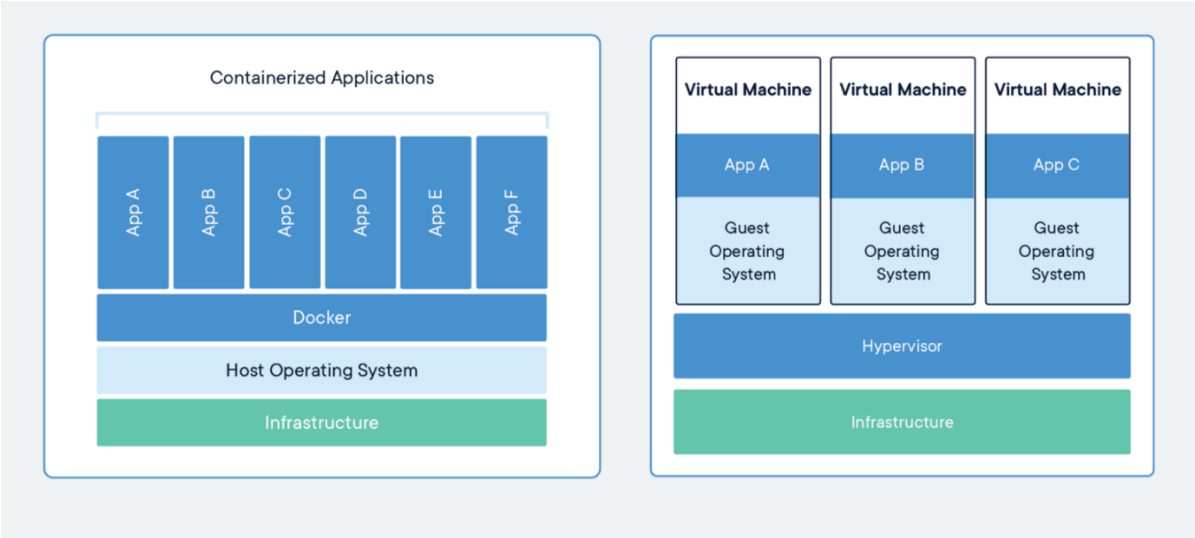
Puhuttaessa konteista ja konttiteknologiasta aiheita voi verrata logistiikkaan ja rahtikontteihin. Ennen rahtifirmoilla oli ongelma: Tavaran kuljettaminen oli hyvin vaikeaa ja kallista, koska tavara jouduttiin rekasta laivaan siirtäessä pakkamaan

aina uudelleen. Vuonna 1950 kehitettiin merikontti, joka tehosti ja nopeutti tavaran kuljettamista. Kun tavara laitettiin konttiin, tavaroiden kuljetus oli paljon halvempaa ja kontti liikkui maateitse rekassa kuin myös merellä laivassa. Ohjelmistoalalla konteissa on kysymys samasta ilmiöstä kuin esimerkissä. (14.)

3.3.1 Kontti

Kontti pakkaa ohjelmakoodin ja sen kaikki riippuvuudet, joten sovellusta voi ajaa ympäristöstä toiseen nopeasti ja luotettavasti. Eräs keskeisin teknologia tähän on Docker, jota myös käytettiin Finlabsin Green Master -projektissa. (15.)

Ohjelman ajamiseen tarvitaan paljon muitakin komponentteja, kuin vain pelkkä koodi. Koodin lisäksi tarvitaan kirjastoja, asetuksia ja lisäosia, joita ilman ohjelmaa ei voi ajaa. Virtuaalipalvelimella voisi ajaa ohjelma ympäristössä, joka tarjoaa sille tarvittavat asiat, mutta se sisältää paljon asioita, joita itse ohjelma ei tarvitse mihinkään. Virtuaalipalvelimen näköistiedoston eli imagen eli koko voikin olla noin 1 GB, joka on kontin imagen kokoon verrattuna jopa sata kertainen. (14.) Kuvassa 5 nähdään, että kontissa pakataan koodi ja riippuvuudet yhteen ja useita kontteja voidaan ajaa samalla koneella ja samalla käyttöjärjestelmällä. Kaikki toimivat silti omina prosesseina. Myös monia virtuaalipalvelimia voidaan ajaa samalla koneella, mutta jokainen virtuaalipalvelin sisältää täydellisen kopion käyttöjärjestelmästä, sovelluksesta ja tarvittavista kirjastoista, jotka vievät valtavasti tilaa. Virtuaalipalvelimen käynnistyskin on hidasta. (15.)

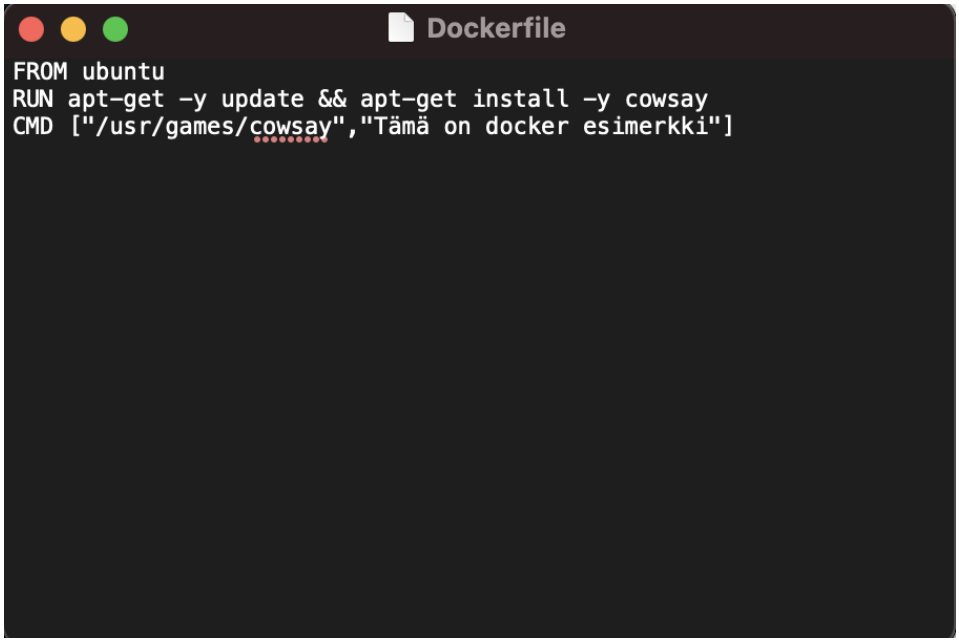


KUVA 5. Kontin ja virtuaalipalvelimen eroavaisuudet (15)

Dockerista on monenlaisia hyötyjä kehityksessä. Koska Docker standardoi eri ympäristöt saman näköisiksi, sovelluksen siirtäminen kehitysympäristöstä testiympäristöön ja sen jälkeen tuotantoon on hyvin suoraviivaista. Tästä johtuen sovelluksen kehitystyö on nopeampaa. Koodi on helpommin siirreltävissä ympäristöstä toiseen ja näin myös päivityksien ja uusien ominaisuuksien julkaisu tapahtuu ketterästi. (16.)

3.3.2 Imagen ajaminen

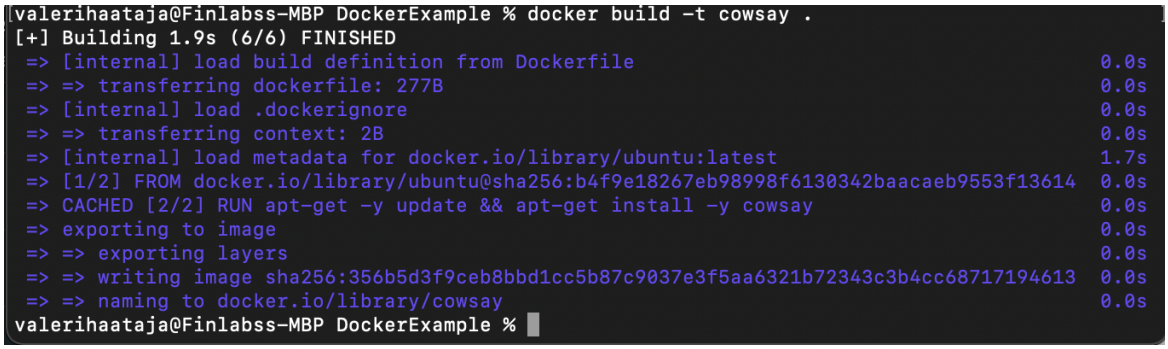
Docker sisältää erilaisia komponentteja, joista yksi on jo aiemmin mainittu Docker-image. Se on erillinen kevyt suoritettava ohjelmapaketti, joka sisältää sovelluksen suorittamiseen tarvittavat asiat eli muun muassa koodin, ajonaikaiset työkalut ja tarvittavat kirjastot (15). Imagen rakentamiseen käytetään Docker-tiedostoa eli Dockerfileä. Se on tekstitiedosto, joka sisältää kaikki komennot imagen rakentamista varten. (19.)



KUVA 6. Dockerfile-esimerkki

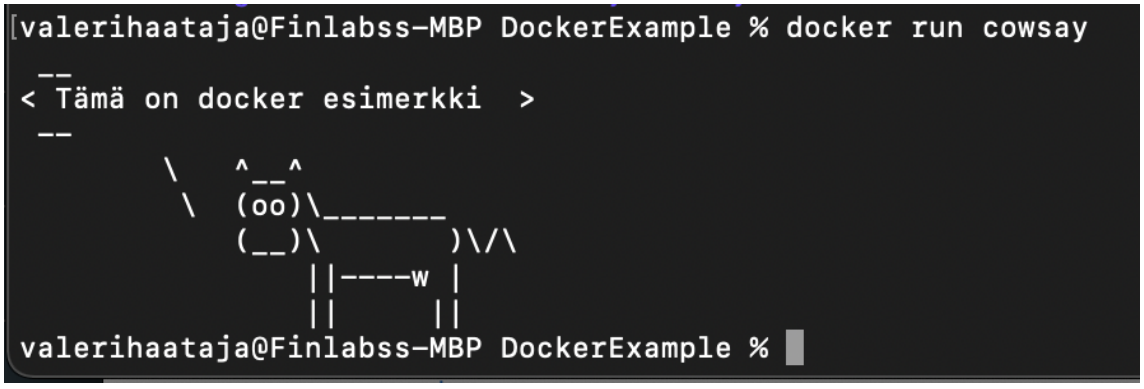
Kuvassa 6 näkyy, että Docker-tiedosto alkaa komennolla FROM, joka määrittää Docker-imagen pohjana käytettävää imagea. RUN ajaa halutut komennot ja tässä tapauksessa asennetaan cowsay -niminen ohjelma. Viimeisellä rivillä oleva CMD:n tarkoitus on tarjota oletukset ajettavalle kontille. Tässä esimerkissä Docker-tiedostossa annetaan parametrina viesti. (19.)

Image rakennetaan Docker-tiedostosta komennolla ”docker build”, joka ajaa Dockerfilen rivi riviltä. Kuvassa 7 rakennetaan Docker-image ja annetaan sille nimeksi cowsay lisäämällä komennon perään ”-t cowsay”.



KUVA 7. Docker-imagen rakentaminen

Kun Docker-image on rakennettu, se voidaan ajaa kontissa (19). Docker-image ajetaan antamalla komento ”docker run”, jonka perään lisätään imagen id tai nimi. Tässä esimerkissä komento ”docker run cowsay” käynnistää kontin, jossa image ajetaan (kuva 8).



KUVA 8. Imagen ajaminen kontissa

3.4 Tensorflow

Tensorflow on Googlen kehittämä avoimen lähdekoodin Python-kirjasto numeerista laskentaa ja koneoppimista varten. Kehittäjä voi rakentaa Tensorflow’lla datavuokaavioita ja rakenteita määrittelemään kuinka data liikkuu kaavion läpi ottamalla inputit moniulotteiseksi taulukoksi, jota kutsutaan Tensoriksi. Tensorflow’n ominaisuuksia käytetään Python-ohjelmointikielellä ja solmut ja tensorit ovat Pyt-

hon-objekteja. Itse matemaattiset operaatiot kuitenkin suoritetaan C++ -binäärei-nä korkean suorituskyvyn vuoksi. Python vain ohjaa liikennettä ja yhdistää operaatiot yhteen. Tensorflow-sovelluksia voidaan ajaa monissa kohteissa, kuten esimerkiksi pilvipalvelussa, mobiililaitteilla ja paikallisesti tietokoneella suorittimella tai näytönohjaimella. (17.)

Suurin yksittäinen hyöty Tensorflow'sta koneoppimisen kehittämiseen on se, että kehittäjä voi keskittyä sovelluksen kokonaislogiikkaan. Tämä tarkoittaa sitä, että ei tarvitse keskittyä algoritmien toiminnan yksityiskohtiin tai keksiä oikeita tapoja liittää toimintojen lähtöjä toisten tuloihin. Tensorflow huolehtii yksityiskohdista. (17.)

3.5 Amazon Web Services

Amazon Web Services eli AWS on skaalautuva pilvialusta, joka toimii satojen tuhansien yrityksien tukena ympäri maailmaa. AWS:n palvelimia on Yhdysvalloissa, Euroopassa, Brasiliassa, Singaporessa, Japanissa ja Australiassa. Sen palveluihin kuuluu muun muassa datantallennus-, Web-ylläpito- ja tietokantapalveluja. (22.)

3.5.1 S3-pilvitallennuspalvelu

S3 on Amazon Web Servicen tallennuspalvelu, johon voi tallentaa kaikenlaista dataa. Yleisesti sitä käytetään muun muassa web-sivujen, sovelluksien ja varmuuskopioiden tallentamiseen. S3-palvelu luo sekä tallentaa kopiot kaikista S3-objekteista automaattisesti. Tämän avulla tallennetut tiedot palvelussa ovat aina käytettävissä ja suojattuina virheiltä sekä uhilta. Kuten muitakin AWS-palveluita, S3-palveluakin käytetään AWS-hallintakonsolin kautta. (23.) Tässä opinnäytetyössä esitelty konenäköjärjestelmä käyttää S3-palvelua opetusdatan tallentamista varten.

3.5.2 ECS-orkestrointipalvelu konteille

Amazon Elastic Container Service eli Amazon ECS on konttien orkestrointipalvelu. ECS:n hyötyjä ovat mahdollisuus palvelimettomaan laskentaan konteille, koska se tukee AWS Fargate -palvelua. Toiseksi ECS on skaalautuva, ja sillä

pystyy nopeasti ja helposti käynnistään tuhansia kontteja. Eräs ECS:n käyttötapauksista on koneoppiminen, jossa voidaan käyttää AWS-syväoppimiskontteja mallien opettamiseen ja tuotantoon laittamiseen. (24.)

3.6 Big Transfer -mallikokoelma

Big Transfer (BiT) on kokoelma valmiiksi opetettuja malleja. Nämä kuvaluokittelumallit on opetettu suuria datasettejä ja tiettyjä neuroverkkoarkkitehtuureja vasten. Näitä malleja ovat esimerkiksi bit/m-r50x1, jota käytettiin myös tässä opinnäytetyössä käsiteltävässä järjestelmässä. Malli m-r50x1 on opetettu ImageNet-21k-datasettiä vasten ja ResNet-arkkitehtuuria hyödyntäen. (25.) Imagenet on yleisesti konenäköjärjestelmissä käytetty kuvatietokanta, jossa kunkin hierarkian solmu on kuvattu sadoilla ja tuhansilla kuvilla (26). Esimerkiksi malli m-r50x1 on opetettu tunnistamaan tuhansia luokkia. Koska nämä BiT-mallit osaavat tunnistaa erilaisia geneerisiä piirteitä kuvista, voidaan mallin opetusta jatkaa omalla datalla ja saada se tunnistamaan omia haluttuja kategorioita. Tähän liittyy termi transfer learning.

Transfer learning on koneoppimismenetelmä, jossa jotain tehtävää varten kehitettyä mallia käytetään uudelleen toisen tehtävän mallin lähtökohtana. Tätä menetelmää käytettäessä täytyy muistaa valita malli, joka on opetettu samankaltaiseen tarkoitukseen. (27.) Kuten tässä järjestelmässä, BiT-malli m-r50x1 on opetettu tunnistamaan valtavasti erilaisia kuvia, joten sitä voidaan käyttää mainiosti tässä käyttötarkoituksessa.

4 KONENÄKÖJÄRJESTELMÄN TUTKIMINEN

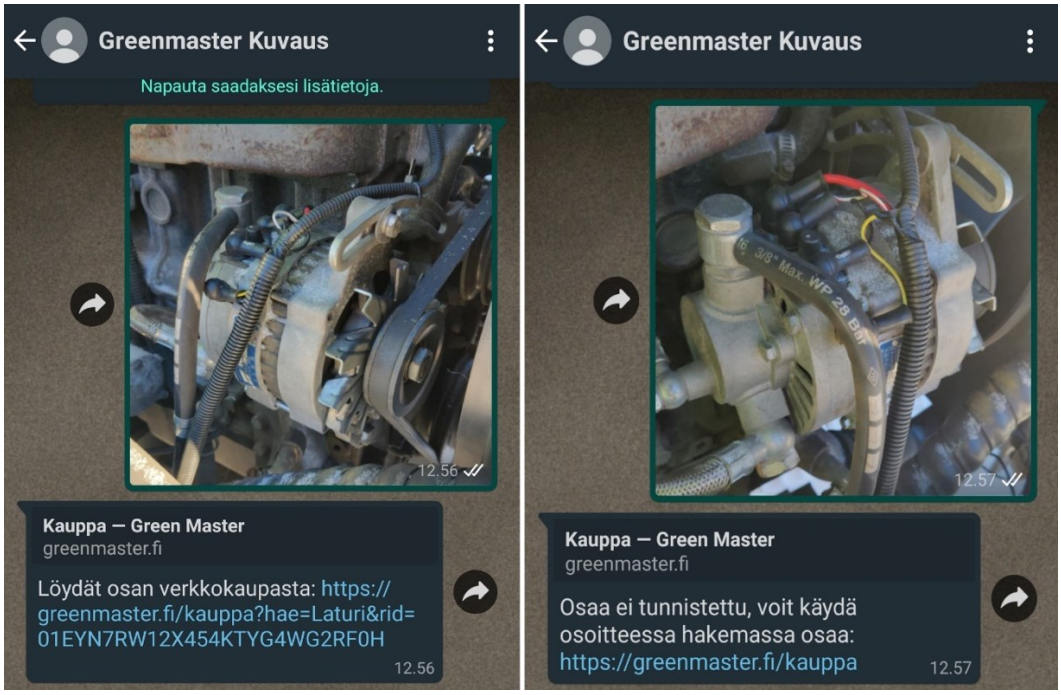
Tässä luvussa käsitellään saatuja havaintoja järjestelmään tutustumisesta ja käydään läpi käytäntöä siitä, kuinka opetus Docker-kontissa tapahtuu.

4.1 Kuvien lähetys järjestelmään WhatsApp-sovelluksella

Järjestelmässä käyttäjän ja järjestelmän välissä käytetään WhatsApp-keskustelusovellusta. WhatsApp olikin varmasti looginen vaihtoehto, koska se on hyvin suosittu. Itsessään keskusteluominaisuus, joka palauttaa linkin, on tehty käyttämällä Twilio-palvelua, joka tukee natiivisti muun muassa SMS- ja WhatsApp-keskusteluja (18). Kun kuva lähetetään, se menee REST API:lle, joka on yhteydessä Docker-imageen, jossa opetettu malli on ajossa. Tämä Docker-image on ajossa Amazon Web Servicen ECS -palvelussa. API palauttaa keskusteluun vastauksen sille lähetetyn kuvan perusteella.

4.2 Havainnot järjestelmästä

Järjestelmän tutkiminen alkoi, kun kävin Green Masterin hallilla kokeilemassa, mihin järjestelmä tällä hetkellä kykenee. Järjestelmän testauksen puolesta oli valitettavaa, että varaosat olivat muualla, joten jouduin testaamaan ainoastaan kuvaamalla suoraan työkoneessa kiinni olevia osia. Heti ensimmäinen havainto testailussa oli se, että samaa osaa kuvatessa eri kuvakulmasta tulos oli poikkeava. Järjestelmä tunnisti siinä olevan osan ja palautti oikean linkin, mutta toiselta puolelta kuvatessa osaa ei tunnistettu, vaikka molemmat kuvat olivat muutoin hyviä ja osat olivat keskellä kuvaa (kuva 9). Tämä herätti kysymyksiä, joten tutkin AWS S3-bucketista, minkälaista opetusmateriaalia oli käytetty tämän osan opettamiseen. Opetusdatan määrä oli melko vähäistä, vain noin parikymmentä kuvaa tästä osasta, ja jotkut kuvista olivat huonolaatuisia tai osa ei ollut kokonaan kuvassa.



KUVA 9. WhatsApp-vastaukset kuville

Yllä olevassa havainnossa käytettiin sen hetkistä järjestelmän tilaa, ja opetus näille osille oli tehty AWS:ssä aiemmin. Kuitenkin opetus oli mahdollista suorittaa myös tietokoneella käyttäen prosessoria tai näytönohjainta, joten täytyi tutustua järjestelmän eri osiin tarkemmin.

4.3 Makefile

Järjestelmän eri toimintoja käytettiin komentorivillä Makefilessä määritettyjen toimintojen perusteella make-ohjelmalla. Makefilen avulla voidaan päättää, mitkä ohjelman osat käännetään uudelleen. Make-apuohjelma täytyy asentaa tietokoneelle, jonka jälkeen sitä voidaan käyttää komentoriviltä komennolla ”make”. (21.)

Esimerkiksi tässä konenäköjärjestelmässä ennen opettamisen aloittamista uudella opetusdatalla täytyi rakentaa uusi Docker-image, joka tehtiin antamalla tässä tapauksessa komento ”make build-train”. Makefile on siis itsessään kätevä tapa kerätä kasaan skriptejä ja make-ohjelmalla voidaan suorittaa niitä.

4.4 Docker-imagen rakentaminen

Kun opetusdataa oli tuotettu, järjestelmän pystyi opettamaan tunnistamaan annettuja kategorioita. Mallin opettaminen oli mahdollista Docker-kontissa tai natiivisti ilman Dockeria. Itse opetuksen pystyi suorittamaan paikallisesti tietokoneella tai AWS:ssä.

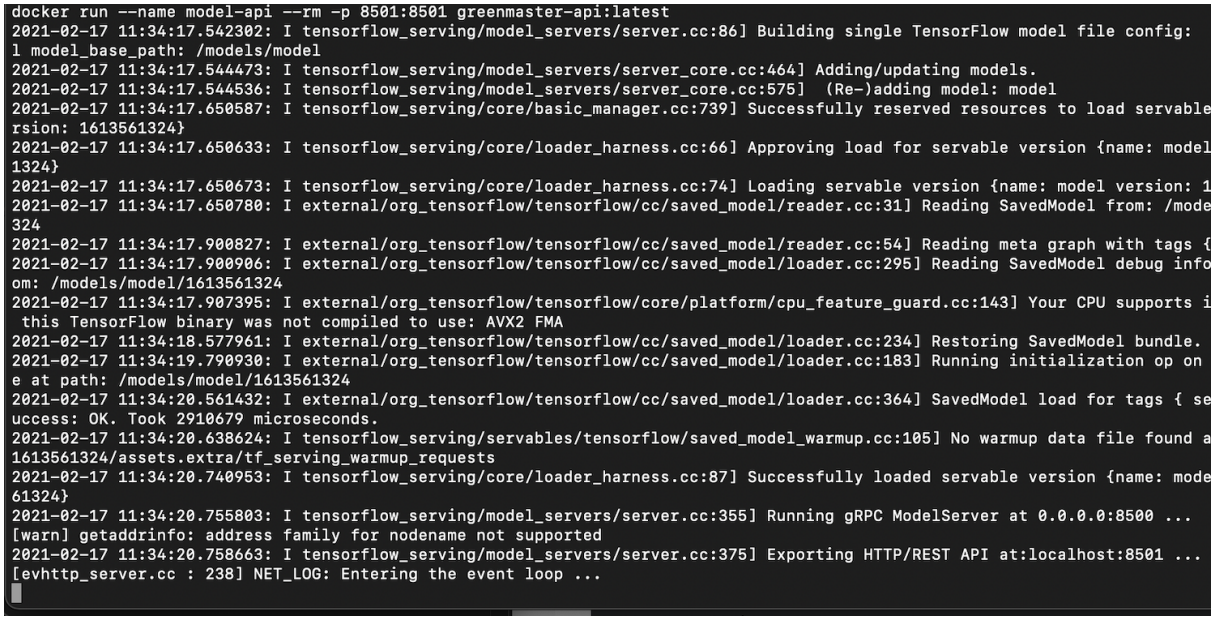
Dockeria käytettäessä ennen opetuksen aloittamista täytyi rakentaa Docker-image. Tässä järjestelmässä sen saavuttamiseksi ensin täytyi ajaa make-komento, joka latasi valmiiksi opetetun mallin TensorFlowHubista sekä latasi opetettavat kuvat AWS S3 -palvelusta. Lopuksi komento rakensi uuden Docker-imagen. Docker-imageja voi tarkastella joko Docker Desktop -ohjelmalla tai komentorivillä komennolla ”docker images” (kuva 10).

```
(tfenv) valerihaataja@Finlabss-MBP gm-service % docker images
REPOSITORY                   TAG       IMAGE ID       CREATED        SIZE
greenmaster-api              latest    6bacba88cfe9   8 days ago     383MB
greenmaster-train-cpu-copy   latest    d19376d13509   9 days ago     2.93GB
cowsay                       latest    356b5d3f9ceb   3 weeks ago    146MB
docker-whale                 latest    356b5d3f9ceb   3 weeks ago    146MB
greenmaster-train-gpu-copy   latest    3b69ea47090f   7 weeks ago    6.27GB
```

KUVA 10. Lista Docker-imageista

4.5 SavedModel

Kun malli oli opetettu, lopputuloksena oli TensorFlow SavedModel -formaatissa oleva malli. Tässä formaatissa oleva malli voi vastaanottaa kuvia, jotka ovat base64-enkoodattuja. Seuraavaksi täytyi saada lähetettyä näitä base64-enkoodattuja kuvia tunnistettavaksi. Tämän saavuttamiseksi täytyi rakentaa ja ajaa Docker API, jonka kautta pystyttiin antamaan neuroverkolle tunnistettavia kuvia ja saamaan vastauksena tunnistamisen tarkkuus. Käytännössä tämä tarkoittaa sitä, että kun valmis malli on ajossa Docker-kontissa, sille voi lähettää kuvia rajapinnan välityksellä. API oli valmis, kun komentoriville tulostui ”*NET_LOG: Entering the event loop* …” (kuva 11).



KUVA 11. Komentorivi kun API on valmis

Ympäristön toiminnan testaamista varten oli tehty valmiiksi skripti, joka lähetti API:lle base64-enkoodatun kuvan. Jos vastaus rajapinnalta saapui, voitiin todeta ympäristön toimivan.

5 MALLIN OPETUS UUDELLA DATALLA

Ennen opinnäytetyön aloittamista pohdimme työn toimeksiantajan kanssa mahdollisia tapoja yrittää parantaa tunnistustarkkuutta. Eräs näistä asioista oli kuvien muokkaaminen, esimerkiksi rajaamalla tai tekemällä niille perspektiivimuunnoksia. Koska aikaisempien havaintojen perusteella opetusdatan laatu oli huonoa, oli ensimmäinen asia miettiä paljonko, ja minkälaista opetusdata kannattaisi olla.

Green Masterin työkoneiden osat olivat muualla, joten jouduin miettimään, mitä muuta voisin järjestelmälle opettaa. Ongelma oli työkoneen osien kanssa pitkälti se, että ne ovat hyvin saman näköisiä toistensa kanssa ja moottorin läheisyydessä on hyvin paljon samankaltaisuutta. Päädyin tulokseen, että opetan koneen tunnistamaan käsityökaluja, koska niitä oli saatavilla ja monet niistä muistuttavat hyvin paljon toisiaan. Työkalujen kategorisointi ja koneen opettaminen niillä olikin hyvin mielenkiintoinen lähtökohta testausprosessin aloitukselle.

5.1 Testiympäristö

Python-ohjelmoinnissa usein on tapana luoda jokaiselle projektille oma virtuaaliympäristö. Tämä tarkoittaa sitä, että luodaan jokaiselle sovellukselle itsenäinen hakemistopuu, johon voidaan asentaa Pythonin tietty versio ja pelkästään halutut kirjastot. Näin vältytään ristiriidoilta kirjastojen versioissa ja saadaan sovelluksille oma eristetty ympäristö, jossa on haluttu Python-versio sekä ainoastaan kirjastot, joita ohjelman suoritukseen tarvitaan. (20.) Tässä projektissakin Python-versiota joutui alentamaan, koska Tensorflow ei tukenut uusinta versiota.

Koska halusin tehdä kaiken paikallisesti tietokoneella, muokkasin sen saavuttamiseksi Makefileä. Poistin sieltä S3-buckettiin liittyvät skriptit, ja loin omaan hakemistoon kansiot. Kansiosta haettiin opetettavat kategoriat ja sinne syötettiin kuvat, jotka haluttiin tunnistettavaksi. Lisäksi Makefilessä oli muuttuja *NUM_OF_EPOCHS,* joka määritti epochien määrän. Epoch on koneoppimisessa käytetty termi, joka ilmaisee algoritmin koko datasetin läpikäyntien määrän (28). Tässä tapauksessa epochien määrä oli oletuksena vain kaksi. Jos epocheja on vain kaksi, malli ei opi tunnistamaan kuvia kovin hyvin, joten muutin arvon kahdesta 25:een.

5.2 Testauksessa käytetty opetusdata

Työn toimeksiantaja halusi järjestelmää testattavan niin, että opetettaisiin se laadukkaalla opetusdatalla, jota on riittävä määrä. Lisäksi tunnistettavien kuvien ottaminen tapahtuisi jonkun muun toimesta. Kun joku muu kuin itse opetusdatan kuvannut ottaa tunnistettavat kuvat, esineen kuvakulma tai tausta poikkeaa melko varmasti opetusdatasta. Tämän avulla testi vastaisi jokseenkin tosielämän tilannetta, kun asiakas ottaa tunnistettavia kuvia eikä tiedä, millaista opetusdata on. Tässä testauksessa projektipäällikkö otti tunnistettavat kuvat. Jos olisin itse opettanut sekä ottanut testikuvat, olisi niihin varmasti alitajuisestikin tullut samankaltaisuutta, joka suoraan vaikuttaa esineen tunnistamiseen. Tällaisessa tapauksessa kone tunnistaisi esineen ongelmitta. Testin tarkoitus oli siis simuloida tilanne tosielämästä, kun asiakas ottaisi kuvan osasta ties millaista taustaa vasten tai kuvakulmaa käyttäen.

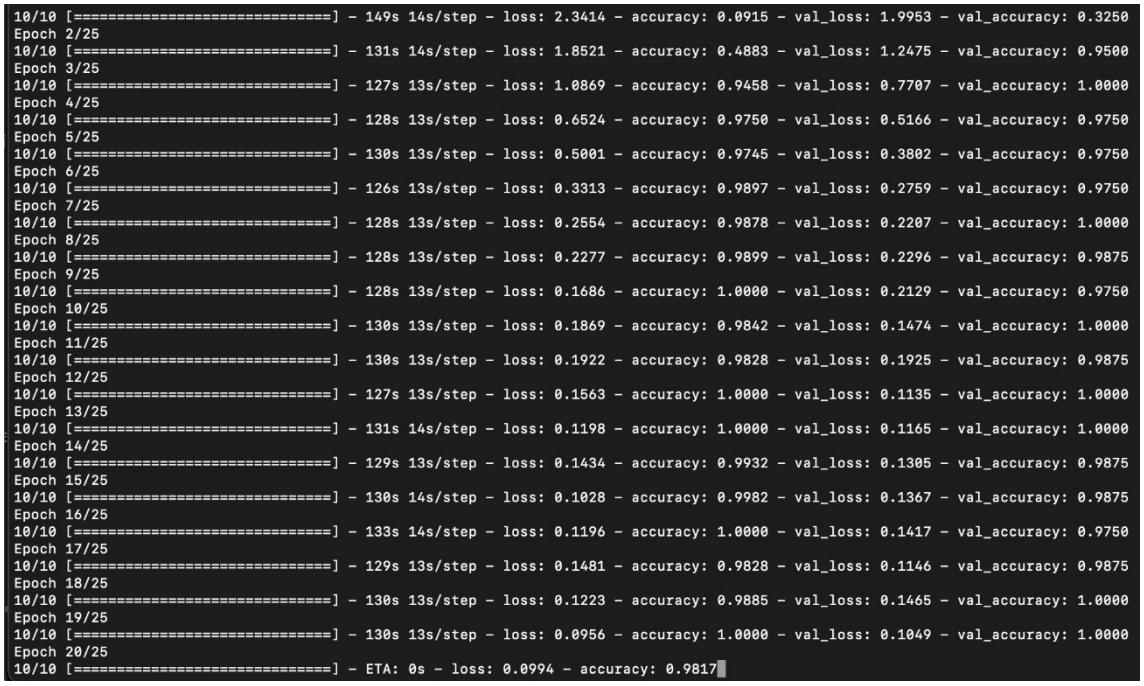
5.2.1 Kategoriat

Testaamista varten valitsin yksitoista kategoriaa käsityökaluista. Kategoriat olivat vasara, jakoavain, lenkkiavain, ruuvimeisseli, pihdit, viila, kuusiokolo, taltta, hohtimet, työntömitta ja ruuvi. Opetusdataksi kuvattiin jokaisesta kategoriasta noin 20 kuvaa valkoista taustaa vasten hyvässä valaistuksessa. Jokaista opetettavaa työkalua kuvattiin eri kulmista ja jokaisen kategorian opetusdatasta tuli tasalaatuista. Opetusdatan laadun olikin oltava hyvää ja tasaista, koska vain näin pystyi testaamaan koneen todellisesta kuvantunnistustarkkuutta. Kuvat piti seuraavaksi laittaa omiin kansioihinsa ja nimetä kansiot sen mukaan, mitä kuvia se sisälsi. Kuvassa 12 on kuudesta eri kategoriasta yhdet opetusdatakuvat.

KUVA 12. Opetettuja kategorioita

5.2.2 Mallin opettaminen

Opetusdatan kuvaamisen, kuvien kategorisoimisen ja Docker-imagen rakentamisen jälkeen mallin opettaminen voitiin tehdä Makefilen skriptien mukaisesti. Tässä tapauksessa malli opetettiin käyttämällä tietokoneen prosessoria. Kuten aiemmin mainittiin, epochien määrä oli muutettu kahteenkymmeneen viiteen ja opetus on itsessään melko raskas prosessi, joten mallin opettaminen vei useita tunteja suorittimella tehtynä. Kuvassa 13 on kuvankaappaus komentoriviltä, kun opetus on käynnissä 25 epochilla.



KUVA 13. Mallin opetus

Opetuksen valmistuessa lopputuloksena oli SavedModel-formaatissa oleva malli, jolle pystyi alkaa antamaan base64-enkoodattuja kuvia tunnistettavaksi.

5.3 Tunnistettavat kuvat

Testin tavoitteena oli siis saada selville konenäön kykyä tunnistaa kuvia, jotka joku opetusdatasta tietämätön on ottanut. Tätä varten projektipäällikkö otti kuvia työkaluista, koska hän ei ollut tietoinen, millaista opetusdataa oli käytetty. Hän myös tarkoituksella kuvasi työkaluja muun muassa erilaisia taustoja vasten, tarkoituksena tarjota hämäävää dataa neuroverkolle. Kuvia tuli yhteensä kaikista kategorioista 54 kappaletta.

5.4 Kuvien lähetys neuroverkolle

Mallin opetuksen jälkeen kuvia voitiin antaa neuroverkolle tunnistettavaksi. Docker-API:n rakentaminen tehtiin tässä vaiheessa. Kun se oli rakennettu ja laitettu ajoon Docker-konttiin, neuroverkolle pystyi lähettämään base64-enkoodattuja kuvia. Koska Docker-rajapinta on REST API, sille voitiin lähettää tietoja HTTP-pyynnöllä. Pyyntöä varten olisi voinut käyttää esimerkiksi curlia, joka on komentorivityökalu URL-syntaksilla määritettyjen tietojen siirtämistä varten,

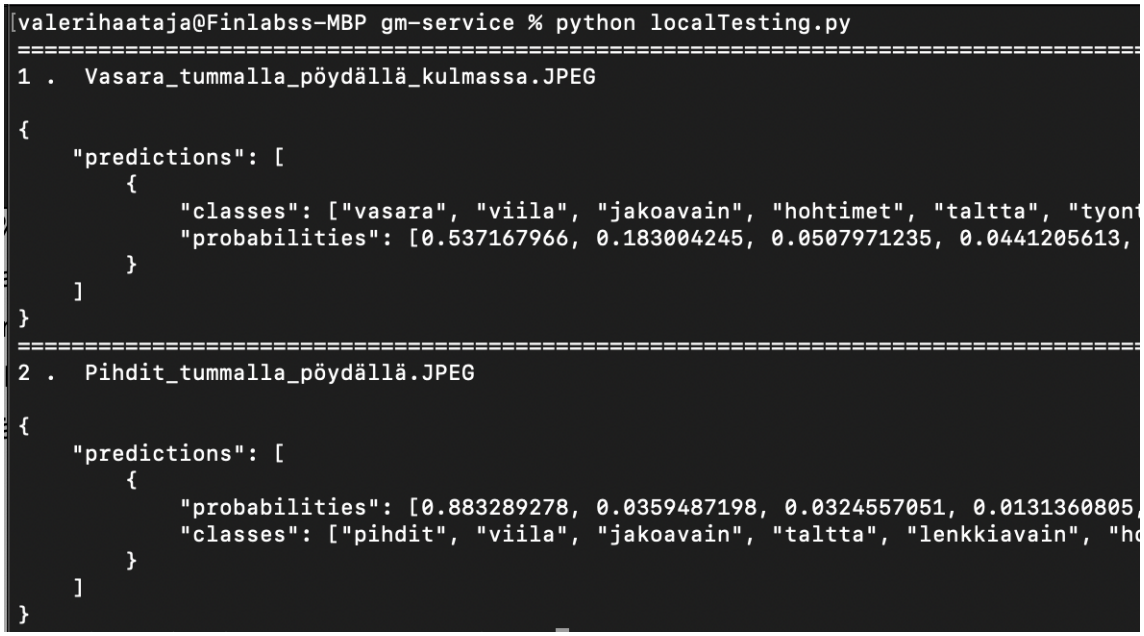
mutta koska järjestelmä oli Python-ympäristössä, päätin tehdä pienen ohjelman Pythonilla kuvien base64-enkoodausta ja HTTP-pyyntöjen lähettämistä varten. Pythonille oli kirjastot olemassa tarvittavia toimintoja varten. Kuvassa 14 on Python-ohjelma, joka hakee kuvat annetusta polusta, base64-enkoodaa ne ja lähettää neuroverkolle.

```
import requests
import base64
import json
import os

directory = '/Users/valerihaataja/Development/Pictures'
counter = 0
for file in os.listdir(directory):
     filename = os.fsdecode(file)
     if filename.endswith(".jpg") or filename.endswith(".png") or
         filename.endswith("jpeg") or filename.endswith("JPEG"):
         picturePath = os.path.join(directory, filename)
         counter = counter + 1
         print("===" * 50)
         print(counter,". ",filename)
         with open(picturePath, 'rb') as datafile:
            data = base64.b64encode(datafile.read()).decode("UTF-8")
            payload = json.dumps({'instances': [{'b64': data}]})
            r = requests.post('http://localhost:8501/v1/models/model:predict',
                headers={"Content-Type": "application/json"}, data=payload)
            data_dict = json.loads(r.text)
            print("")
            print(r.text)
```

KUVA 14. Python-ohjelma tunnistustuloksia varten

Ohjelma tulostaa jokaisen kuvan jälkeen vastauksen komentoriville, jossa näkyy kuvan nimi sekä vastaus neuroverkolta. Vastaus tulee JSON-muodossa ja siinä on nähtävissä annetun kuvan todennäköisyydet sekä kaikki opetetut kategoriat. Todennäköisyys esitetään asteikolla 0–1, joten voi ajatella, että esimerkiksi 0,50 tarkoittaa 50 %:n todennäköisyyttä. Kun kuvassa esiintyvä esine tulee ensimmäisenä vaihtoehtona, vaikka pienemmälläkin todennäköisyydellä, voitaisiin silti todeta, että se on onnistuneesti tunnistettu. Kuvassa 15 on näkymää, kun Python-ohjelma suoritettiin.



KUVA 15. Vastaukset komentorivillä

Kaikki tunnistettavat kuvat tuli nimetä niin, että komentoriviltä katsoessa tiesi, mitä mikäkin kuva esittää. Lisäsin sitten kaikki kuvat hakemistoon ja ajoin ohjelman, jonka jälkeen otin tulokset jokaisesta ylös.

6 TULOKSET

Monet tunnistettavat kuvat poikkesivat opetusdatan kuvista hyvinkin paljon. Tämä tehtiin testitarkoituksessa, jotta nähtiin, mihin kaikkeen opetettu malli tarttui kuvia tunnistaessa. Esimerkki tallaisesta tapauksesta on kuvassa 16, jossa on vasemmalla tunnistettava kuva ja oikealla yksi opetusdatan kuvista. Kuten on huomattavissa, tässä käytettiin kokonaan eri jakoavainta tunnistettavana, sekä häiriötekijäksi lisättiin muitakin työkaluja taustalle.

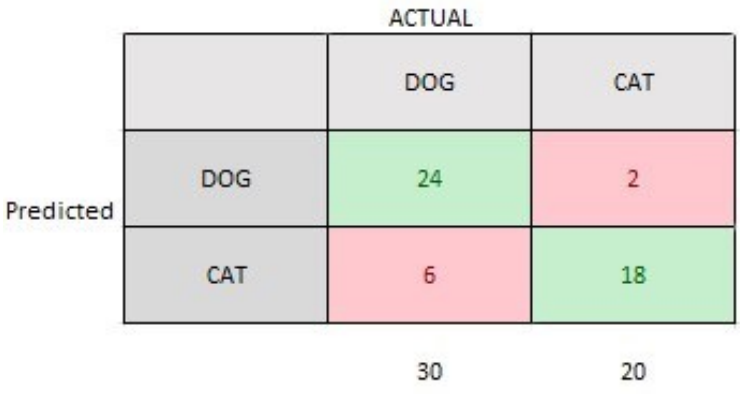
KUVA 16. Tunnistettavaksi lähetetty kuva ja opetusmateriaalin kuva

Yllä olevassa tapauksessa neuroverkko palautti kategorian ”jakoavain” ensimmäisenä vaihtoehtona todennäköisyydellä 0,46. Vaikka tässä kuvassa käytettiin eri jakoavainta ja taustalla oli häiriötekijöitä, kone tunnisti työkalun hyvin. Voi siis todeta, että kone tunnisti jakoavaimen muodon ja pinnan piirteistä.

6.1 Sekaannusmatriisi

Koneoppimisen tuloksien analysointiin käytetään usein sekaannusmatriisia eli confusion matrixia. Se on taulukko, jota voidaan käyttää luokittelumallin suoritus-

kyvyn testaamiseen testidatalla, joiden todelliset arvot tiedetään. Sekaannusmatriisissa luokitellaan oikeat positiiviset ja negatiiviset arvot eli ne, joissa mallin ennuste on pitää myös paikkaansa. Lisäksi matriisissa luokitellaan väärät positiiviset ja negatiiviset arvot, jotka eivät pidä paikkaansa. (29.) Kuvassa 17 on yksinkertainen esimerkki sekaannusmatriisista, jonka testidatassa on 30 koiraa ja 20 kissa.



KUVA 17. Sekaannusmatriisi esimerkki

Mallin todellinen tarkkuus saadaan laskemalla oikein luokitellut yhteen ja jakamalla se testidatan kokonaismäärällä (30). Yllä olevassa esimerkissä tarkkuus on siis 42/50, eli 0,84. Sekaannusmatriisilla voidaan laskea tarkkuuden lisäksi paljon muutakin. Näitä ovat muun muassa väärät luokitukset, todellisten positiivisten ja negatiivisten osuus, väärien positiivisten ja negatiivisten osuus ja luotettavuus. (29.)

6.2 Tunnistustarkkuus

Tein opetetuista kategorioista sekaannusmatriisin tuloksien tarkastelua varten (kuva 18). Kuten siitä on nähtävissä, kaikkien työkaluista otettujen kuvien tunnistamisen jälkeen neuroverkko palautti oikean kategorian ensimmäisenä vaihtoehtona 47 kuvassa. Kuvia oli yhteensä 54, joten malli luokitteli seitsemän kuvaa väärin, mikä tekee mallin tarkkuudeksi 47/54, eli 0,87. Todennäköisyydet olivat kuitenkin pieniä monissa kuvissa, vaikka luokittelu menikin oikein. Tällainen tilanne voisi johtaa tosielämässä siihen, että kone saattaisi palauttaa väärän kategorian, varsinkin silloin, jos kategorioita olisi enemmän. Mallin kokonaistarkkuus eli 0,87 oli kuitenkin todella hyvä.

Sekaannusmatriisista näkee myös helposti, mihin luokkiin konenäkö sekoitti kuvia. Esimerkiksi luokissa jakoavain, lenkkiavain, ruuvimeisseli ja taltta, konenäkö sekoitti kuvia viilaan. Hohtimien kohdalla on nähtävissä, että yksi kuva neljästä oli mallin mielestä lenkkiavain. Matriisin jokaisen rivin loppuun on laskettu yhteen jokaisen kategorian ennustuksien yhteenlasketut määrät, jotka sisältävät siis väärin sekä oikein menneet ennustukset. Jokaisen sarakkeen lopussa on todelliset määrät. Väärin menneet ovat punaisella pohjalla ja oikein menneet vihreällä pohjalla.

	Hohtimet	Jakoavain	Kuusiokolo	Lenkkiavain	Pihdit	Ruuvi	Ruuvimeisseli	Taltta	Viila	Työntömitta	Vasara	Yhteensä
Hohtimet	3	0	0	0	0	0	0	0	0	0	0	3
Jakoavain	0	4	0	0	0	0	0	0	0	0	0	4
Kuusiokolo	0	0	3	0	0	0	0	0	0	0	0	3
Lenkkiavain	1	0	0	5	0	0	0	0	0	0	0	6
Pihdit	0	0	0	0	3	0	0	0	0	0	0	3
Ruuvi	0	0	0	0	0	4	0	0	0	0	0	4
Ruuvimeisseli	0	0	0	0	0	0	7	0	0	0	0	7
Taltta	0	0	0	0	0	0	0	3	0	0	0	3
Viila	0	1	0	1	0	0	2	2	6	0	0	12
Työntömitta	0	0	0	0	0	0	0	0	0	5	0	5
Vasara	0	0	0	0	0	0	0	0	0	0	4	4
Yhteensä	4	5	3	6	3	4	9	5	6	5	4	54

KUVA 18. Kategorioiden sekaannusmatriisi

Mallin ennustuksissa eniten pieleen meni kategoria ”viila”. Kuten yllä olevasta matriisista on nähtävissä, mallin mielestä viila esiintyi kahdessatoista kuvassa, vaikka todellinen määrä oli vain kuusi.

Kuten mainittu, sekaannusmatriisin avulla voidaan laskea erilaisia arvoja. Voidaan esimerkiksi laskea, kuinka usein malli oli väärässä. Se saadaan laskemalla ensin kaikki väärin menneet ennustukset yhteen ja jakamalla summa datan kokonaismäärällä. Tässä tapauksessa väärin menneitä ennustuksia oli 7 ja kuvia oli yhteensä 54, mikä tekee 7/54, joka on 0,129 (29).

6.3 Kuvien rajaus

Tosielämässä työkoneiden osien tunnistamista häiritsi erilaiset taustat sekä kuvissa ilmaantuvat muut osat. Tällaisessa tilanteessa voisi siis parantaa tarkkuutta, jos hämäävä tausta olisi rajattu pois kuvasta.

Tämän testauksen tarkoitus oli siis yrittää parantaa tunnistustuloksia rajaamalla kuvia. Kone tunnisti eli antoi oikean kategorian alkuperäisistä kuvista 0,87:n tarkkuudella, joten rajasin kaikki väärin luokitellut kuvat. Rajaamisen jälkeen kone tunnisti kaikki kuvat, joten tässä vaiheessa pystyi jo toteamaan kuvien rajauksien auttavan hyvin paljon tunnistamisessa.

Kuitenkin noiden kuvien tunnistustarkkuudet olivat osittain hyvin pieniä, ja sen takia tein uuden testin. Siinä hyväksyin kuvan tunnistetuksi ainoastaan silloin, jos se oli saanut yli 0,60:n todennäköisyyden. Yleisesti ottaen yli 0,60:n todennäköisyys on hyvin korkea tulos ja yleensä konenäköjärjestelmissä riittäisikin, että oikea kategoria tulisi ensimmäisenä vaihtoehtona. Tämä jo saavutettiin edellisen testin kuvien rajauksien jälkeen. Kuitenkin näitä ”todella hyvin” tunnistettuja, yli 0,60:n todennäköisyyden saaneita kuvia oli 32 kappaletta eli mallin tarkkuus oli vain 0,59. Rajasin kaikki kuvat, jotka saivat tulokseksi alle 0,60 niin, että työkalu oli kuvan keskellä ja taustaa oli rajattu mahdollisimman paljon pois.

Kuvien rajausten jälkeen lähetin kuvat neuroverkolle tunnistettavaksi ja tulos oli mielestäni erittäin hyvä. Ennen rajauksia yli 0,60:n saaneita kuvia oli 32 ja rajauksien jälkeen tulos oli 47. Tämä tarkoittaa, että uusia tällä todennäköisyydellä tunnistettuja kuvia tuli lisää 15 kappaletta. Kokonaistuloksena tunnistustarkkuudeksi saatiin siis tässä tapauksessa 0,87, eli tarkkuus parani lähes kolmekymmentä prosenttia.

Vaikka opetusmateriaali ja tunnistettavaksi otetut kuvat työkaluista poikkesivat toisistaan, kuvien tunnistustarkkuus oli kuitenkin hyvä. Yksi kuvista, jossa kone ei tunnistanut siinä esiintyvää työkalua, sai todennäköisyyden 0,05, joka on äärimmäisen huono tulos. Tämä varmaankin johtui siitä, että kuvassa esiintyvä työkalu on hyvin kaukana, koska rajauksen jälkeen tämän kuvan todennäköisyys nousi 0,05:stä 0,79:ään (kuva 19).

KUVA 19. Yksi tunnistamattomista kuvista

Testin tulos, jonka tarkoitus oli nähdä, paljonko kuvan rajaus vaikuttaa tunnistukseen oli hyvä. Ensimmäisessä tapauksessa, jossa jätettiin tunnistustarkkuuden arvot huomiotta, rajauksien jälkeen tulos parani 7 kuvaa eli kaikki kuvat tunnistettiin.

Kuvien, joiden kohdalla tarkkuuden täytyi ylittää 0,60, parannus oli 15 kuvaa eli 27,78 %. Tämä on hyvä tulos, joten voi siis todeta, että tässä tapauksessa kone tunnisti esineen piirteet ja muodot hyvin tarkasti.

Konenäköjärjestelmä toimi hyvin silloin, kun se oli opetettu laadukkaalla opetusdatalla ja opetusta itsessään oli suoritettu tarpeeksi. Lisäksi kuvien tunnistus parani huomattavasti, kun niitä rajattiin. Jos tunnistettavissa kuvissa esine olisi mahdollisimman keskellä kuvaa, tunnistustarkkuus olisi varmasti hyvä. Myös Green Masterin työkoneiden varaosien tunnistustarkoituksessa voisi todeta järjestelmän

pystyvän tunnistamaan osia hyvin, jos opetusdatan määrään ja laatuun panostettaisiin enemmän.

7 YHTEENVETO

Konenäköjärjestelmän tunnistustulokset olivat olleet huonoja Green Masterin työkoneiden osien tunnistamisessa. Opinnäytetyön päämääränä oli tutkia järjestelmän toimintaa ja etsiä sekä testata konenäön tunnistustarkkuutta parantavia tekijöitä.

Lopputuloksena saatiin hyödyllistä tietoa järjestelmän jatkoa ajatellen. Järjestelmä todettiin testien perusteella toimivaksi. Hyviin tuloksiin päästiin, kun opetusdatan laatu oli riittävällä tasolla. Lisäksi tunnistettavat kuvat pitäisi olla kuvattu niin, että taustaa olisi kuvassa mahdollisimman vähän. Tutkimisen ja testien perusteella voisikin sanoa, että järjestelmän ollessa asiakaskäytössä opetusdatan laatu ja määrä pitäisi olla standardisoitua tasaisen tunnistustarkkuuden varmistamiseksi.

Koneoppiminen ja lähes kaikki käsitellyistä aihealueista olivat minulle uusia ennen opinnäytetyön aloittamista, joten niiden opettelussa menikin melko paljon aikaa. Opinnäytetyön aihe oli mielestäni todella mielenkiintoinen, ja työ oli kokonaisuudessaan erittäin opettavainen kokemus. Tämän kaltaiselle järjestelmälle voisi olla rajattomia mahdollisuuksia, koska sen voi opettaa tunnistamaan käytännössä mitä tahansa. Työ sisälsi paljon teknologioita, joita tulen varmasti tarvitsemaan ja käyttämään tulevaisuudessa.

LÄHTEET

1. At Finlabs, we help our clients take full advantage of new and evolving technologies to make their ideas a reality. Finlabs. Saatavissa: https://finlabs.fi/about/. Hakupäivä 14.12.2020.

2. Pulkkinen, Markku 2021. Founder, Finlabs. Haastattelu. Saatavissa: Oma arkisto. Haastateltu 23.2.2021.

3. Green Masterin Tarina. Green Master. Saatavissa: https://greenmaster.fi/meista/. Hakupäivä 15.12.2020.

4. What is Artificial Intelligence? 2020. Builtin. Saatavissa: https://builtin.com/artificial-intelligence. Hakupäivä 28.12.2020.

5. Artificial Intelligence (AI). 2020. IBM. Saatavissa: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence. Hakupäivä 30.12.2020.

6. Weak Artifical Intelligence (Weak AI). 2020. technopedia. Saatavissa: https://www.techopedia.com/definition/31621/weak-artificial-intelligence-weak-ai . Hakupäivä 3.1.2021

7. Mitä on tekoäly? FabricAI. Saatavissa: https://fabricai.fi/mita-on-tekoaly/. Hakupäivä 3.1.2021.

8. Viitaila, Mikko. Tekoälyn perusteet: koneoppiminen, työn tulevaisuus ja hyvä vai paha tekoäly. 2020. Microsoft. Saatavissa: https://pulse.microsoft.com/fi-fi/business-leadership-fi-fi/na/fa2-tekoalyn-perusteet-koneoppiminen-tyon-tulevaisuus-ja-hyva-vai-paha-tekoaly/. Hakupäivä 1.1.2021.

9. Mitä tekoäly on? Skycode Oy. Saatavissa: https://xn--tekoly-eua.info/mita_tekoaly_on/. Hakupäivä 1.1.2021

10. Radich, Quinn & Cowley Eric 2019. What is a Machine learning model?. Microsoft. Saatavissa: https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model. Hakupäivä 20.1.2021.

11. McCullum, Nick 2020. Deep Learning Neural Networks Explained in Plain English. FreeCodeCamp. Saatavissa: https://www.freecode-camp.org/news/deep-learning-neural-networks-explained-in-plain-english/. Hakupäivä: 25.1.2021

12. Oppermann, Artem 2019. What is Deep Learning and How does it work?. 2019. Medium. Saatavissa: https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac. Hakupäivä 20.1.2021.

13. Shin, Terence 2020. All Machine Learning Models explained in 6 Minutes. Medium. Saatavissa: https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a. Hakupäivä 21.1.2021.

14. Wallenius, Niklas 2019. Konttiteknologia – mitä kontit ovat ja mitä hyötyä niistä on? Saatavissa: https://niklaswallenius.fi/konttiteknologia-mita-hyotya/. Hakupäivä 12.2.2021.

15. What is a Container? Docker. Saatavissa: https://www.docker.com/resources/what-container. Hakupäivä 23.2.2021.

16. Wallenius, Niklas 2020. Mikä on Docker ja mitä hyötyä siitä on? Saatavissa: https://niklaswallenius.fi/mika-on-docker/. Hakupäivä 12.2.2021.

17. Yegulalp, Serdar 2019. What is TensorFlow? The machine learning library explained. InfoWorld Saatavissa: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html. Hakupäivä 22.2.2021.

18. Conversations API. 2021. TWILIO INC. Saatavissa: https://www.twilio.com/conversations-api. Hakupäivä 23.2.2021.

19. Dockerfile reference. Docker docs. 2021. Docker Inc. Saatavissa: https://docs.docker.com/engine/reference/builder/. Hakupäivä 14.3.2021.

20. Virtual Environments and Packages. 2021. Python Software Foundation. Saatavissa: https://docs.python.org/3/tutorial/venv.html. Hakupäivä 23.3.2021.

21. Learn Makefiles. Makefiletutorial.com. Saatavissa: https://makefiletutorial.com/. Hakupäivä 5.4.2021.

22. About AWS. 2021. Amazon Web Sevices. Saatavissa: https://aws.amazon.com/about-aws/. Hakupäivä 5.4.2021.

23. Amazon S3. 2021. Amazon Web Services. Saatavissa: https://aws.amazon.com/s3/. Hakupäivä 5.4.2021.

24. Amazon Elastic Container Service. 2021. Amazon Web Services. Saatavissa: https://aws.amazon.com/ecs/. Hakupäivä 5.4.2021.

25. Bit/m-r50x1. 2021. TensorFlow Hub. Saatavissa: https://tfhub.dev/google/bit/m-r50x1/1. Hakupäivä 5.4.2021.

26. ImageNet. 2021. Stanford Vision Lab. Saatavissa: http://www.image-net.org/. Hakupäivä 5.4.2021.

27. Brownle, Jason 2017. A Gentle Intoduction to Transfer Learning for Deep Learning. Machine Learning Mastery. Saatavissa: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. Hakupäivä 5.4.2021.

28. Bell, Daniel J & Gaillard, Frank. Epoch (machine learning). Radiopaedia. Saatavissa: https://radiopaedia.org/articles/epoch-machine-learning. Hakupäivä: 5.4.2021.

29. Simple guide to confusion matrix terminology. 2014. Data School. Saatavissa: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. Hakupäivä 24.4.2021.

30. Koneoppimismallien tulokset. 2020. Microsoft. Saatavissa: https://docs.microsoft.com/fi-fi/dynamics365/finance/finance-insights/confusion-matrix. Hakupäivä 24.4.2021.