Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Miika Martikainen

Koneoppimisen hyödyntäminen ohjelmistorobotiikassa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

8.11.2019

Tekijä Otsikko Sivumäärä Aika	Miika Martikainen Koneoppimisen hyödyntäminen ohjelmistorobotiikassa 28 sivua 8.11.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Smart Systems and Software Engineering
Ohjaajat	Kimmo Sauren

Insinöörityössä oli tavoitteena luoda ohjelmistorobotiikalla ratkaisu, joka toimii ihmisen rinnalla tiedostojen ja niiden sisältämän datan keräämiseen. Samalla ratkaisu nopeuttaa työprosessin sulavuutta ja tehokkuutta.

Ohjelmistorobotiikan lisäksi työssä perehdyttiin koneoppimiseen ja siihen, miten sitä voitaisiin hyödyntää osana ohjelmistorobotiikan toteutusta. Koneoppimista päädyttiin hyödyntämään neuroverkkoihin perustuvaan tekstintunnistukseen kuvatiedostoista, joiden kopioiminen ei onnistu tavallisilla ohjelmistorobotiikan sisältämillä työkaluilla.

Insinöörityön lopputuloksena oli toimiva ohjelmistorobotti, joka suoritti työprosessin niin kuin sen oli tarkoitus. Se nopeutti tiedostojen hakemista sekä niiden käsittelyä huomattavasti aiempaan avustamattomaan työprosessiin verrattuna.

Avainsanat	Ohjelmistorobotiikka, koneoppiminen

Author Title Number of Pages Date	Miika Martikainen Utilizing machine learning in robotic process automation 28 pages 8 November 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Smart Systems and Software Engineering
Instructors	Kimmo Sauren

The purpose of this final year project was to design a robotic process automation solution that would work alongside with a human operator to increase the efficiency of collecting documents and processing the information they contain.

This project also includes the study of machine learning and how it could be used in combination with the robotic process solution, the robot ended up using neural network based character recognition to extract information from image files that the robot cannot copy with basic tools.

As a result of this thesis the product was a functioning robot that increased the efficiency of collecting and processing information considerably when comparing to the previous unassisted work process.

Keywords	robotic process automation, machine learning

Sisällys

Lyhenteet

OCR – Optical character recognition eli teknologia, jolla tunnistetaan käsinkirjoitettua tekstiä digitaalisista kuvista.

IMAP - Internet Message Access Protocol on protokolla, joka tallentaa sähköpostiviestit serverille mutta antaa käyttäjän muokata viestejä aivan, kuten ne olisi tallennettu käyttäjän tietokoneelle.

NIST - National Institute of Standards and Technology database on tietokanta, joka sisältää käsinkirjoitettuja lomakkeita neuroverkkojen kouluttamiseen.

MNIST - Modified National Institute of Standards and Technology database on muokattu versio NIST tietokannasta, joka sisältää käsinkirjoitettuja numeroita neuroverkkojen kouluttamiseen.

LSTM - Long Short-Term Memory eli keinotekoinen neuroverkko arkkitehtuuri, jota käytetään koneoppimisessa.

1 Johdanto

Insinöörityön tarkoituksena oli rakentaa ohjelmistorobotti, joka nopeuttaisi ja helpottaisi puuduttavaa toistuvaa työprosessia toimimalla ihmisen rinnalla heidän käsitellessä sähköpostissa tulevien dokumenttien tietoja. Ohjelmistorobotti ohjelmoitiin keräämään kaikki tieto, mitä dokumenteista tarvitaan, ja tallentamaan se tietokoneelle, jotta työntekijöiden on helpompaa tarkistaa tiedot ja syöttää ne tietokantaan. Koska sähköpostissa tulevissa liitteissä on myös kuvatiedostoja, tarvittiin niiden käsittelemiseen koneoppimisella toteutettu tekstintunnistustyökalu.

Insinöörityön alussa käydään läpi ohjelmistorobotiikan kehitystä ja historiaa, jotta ymmärretään, mihin sitä voidaan käyttää, sekä sen heikkouksia, vahvuuksia ja sitä, miten sen toteutus tapahtui. Tämän jälkeen käydään läpi, mitä koneoppiminen on ja miten sitä hyödynnetään ohjelmistorobotiikan rinnalla, sekä selitetään olennaisia osia koneoppimiseen liittyvistä alakategorioista.

2 Ohjelmistorobotiikka

Ohjelmistorobotiikkaa käytetään automatisoimaan työtehtäviä, jotka ovat toistuvia, eivätkä tarvitse päätösten tekemistä, esimerkiksi lomakkeista tiedon kerääminen, ja saaduilla tiedoilla toisen lomakkeen täyttäminen. Tällöin ihmisten ei tarvitse tehdä paljon aikaa vieviä toistuvia prosesseja, ja heille vapautuu aikaa suorittaa haastavampia työtehtäviä, mikä kasvattaa yrityksen tuottavuutta ja parantaa työntekijätyytyväisyyttä. Ohjelmistorobotiikka on ollut nousussa lähivuosina. Esimerkiksi finanssi-, terveys-, vakuutus- ja telealat ovat ottaneet ohjelmistorobotiikkaa käyttöön vapauttaakseen ihmisresursseja muihin työtehtäviin. (1.)

2.1 Historia

Ohjelmistorobotiikka on termi, jota alettiin käyttämään 2000-luvun alussa. Vaikka itse termi on uusi, on itse teknologiaa kehitetty jo vuosikymmenten ajan. Ohjelmistorobotiikka tuo yhteen eri teknologioita, jotka mahdollistavat työprosessien automatisoinnin. Kolme tärkeintä edeltäjää ohjelmistorobotiikan kehitykselle ovat näytön kaavintaohjelmistot, työnkulun automaatio ja hallintotyökalut ja tekoäly. (2.)

Näytönkaavinta teknologia syntyi ennen internetin kehitystä, ja sen kehitys mahdollisti vanhojen järjestelmien tietojen käytön uusissa järjestelmissä, jotka eivät olleet yhteensopivia vanhojen järjestelmien kanssa kaapimalla tiedot vanhan järjestelmän näytöltä uuteen järjestelmään. Nykypäivänä näytönkaavintaohjelmia käytetään vähentämään manuaalista työtä, kun kerätään tietoja web-sivustojen esitystapakerroksesta. (2.)

Työnkulun automaation kehitys alkoi 1920-luvulla, kun tuotteiden valmistusprosesseja muutettiin liukuhihnamaiseen muotoon, missä tuotteiden valmistusmäärät nousivat, ja oli tärkeää saada tuotanto järjestykseen. Työnkulun automaatio-ohjelmistoja kehitettiin 1990-luvulla pidemmälle sen nykypäiväiseen muotoonsa. Sitä alettiin hyödyntämään esimerkiksi tilausten käsittelyssä, sillä ohjelmisto auttaa keräämään lomakkeista tietoja, esimerkiksi asiakkaan yhteystiedot, laskun summan ja tilatun tuotteen tai palvelun, ja tal-

lentaa tiedot yrityksen tietokantaan, koska ohjelmisto poistaa ihmisen tarpeen datan keräämisessä. Sen tallentamisessa tietokantaan nopeutuu tilausten käsittely huomattavasti. (2.)

Ohjelmistorobotiikka alkoi kehittymään itsenäiseksi teknologiaksi 2000-luvun alussa. Vaikka ohjelmistorobotiikan kehitys oli nopeaa ja sen hyödyt olivat selviä, yritykset olivat epäluuloisia robottien suhteen eivätkä luottaneet niiden toimintaan. Vasta vuonna 2015 alkoi ohjelmistorobotiikka nousemaan suureen suosioon yritysten keskuudessa, robotit alkoivat kehittymään paremmiksi, ja niillä pystyttiin automatisoimaan useita asioita, missä monet yritykset pystyisivät tekemään säästöjä. Vaikka robotit ovat tällä hetkellä rajoittuneita tekemään toistuvia ja puuduttavia tehtäviä, ihmiset uskovat, että tulevaisuudessa tulee olemaan kognitiivisia robotteja. (3.)

Ohjelmistorobotiikan markkinakehitys on vuoden 2000 jälkeen kasvanut räjähdysmäisesti, kun robotit kehittyvät. Tällöin ne pystyvät suorittamaan useampia tehtäviä, ja niiden yritykselle tuottamansa arvo nousee. Gartner-tutkimuskeskuksen tekemän selvityksen mukaan ohjelmistorobotiikan markkina-arvo kasvoi 63 % vuonna 2018. (4.) Ohjelmistorobotiikan talouden on ennustettu ylittävän 8,75 miljardia dollaria vuoteen 2024 mennessä. Tämä tarkoittaa, että ihmisiä tullaan korvaamaan roboteilla yksinkertaisissa datankäsittelytehtävissä. Vaikka uusia robotteja tullaan ottamaan käyttöön ja niillä korvataan tämänhetkisiä työskenteleviä ihmisiä, Pew-tutkimuskeskuksen vuonna 2013 suorittamaan kyselyyn vastanneista asiantuntijoista 52 % on sitä mieltä, että ohjelmistorobotiikka tulee luomaan uusia työpaikkoja enemmän kuin se tulee poistamaan. (1; 5.)

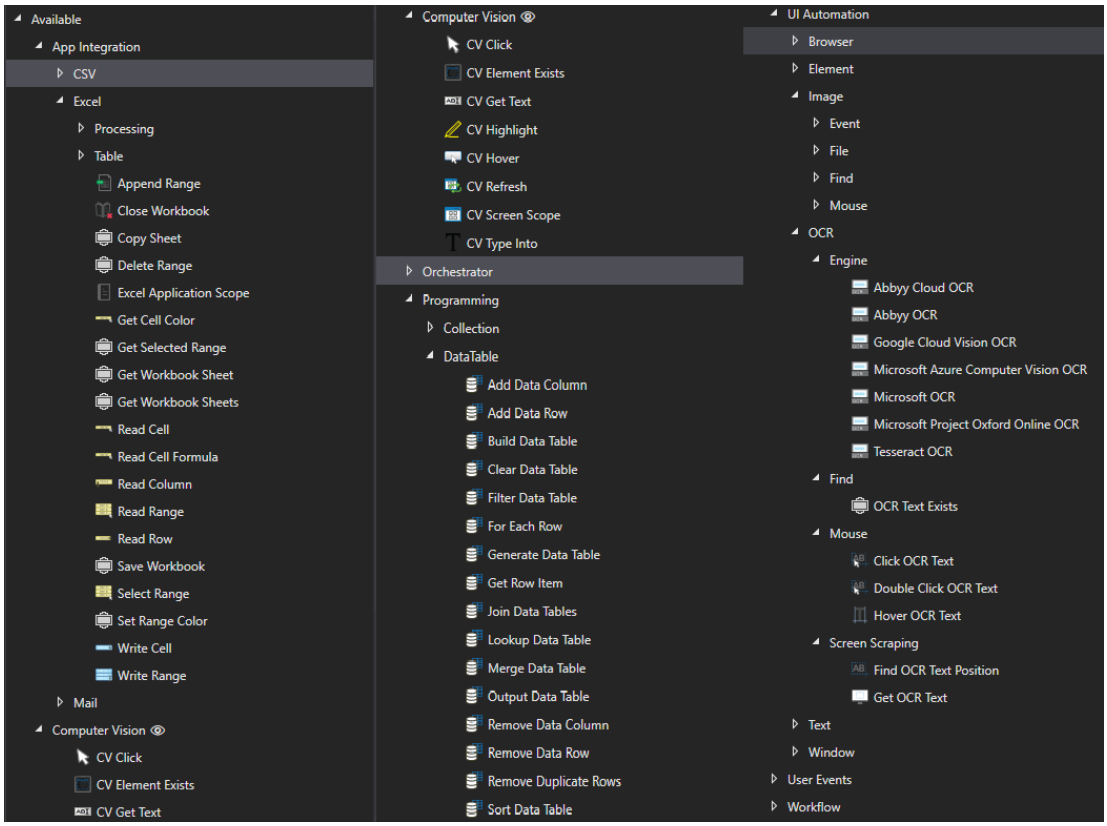
2.2 Ohjelmointi

Ohjelmistorobotiikka on kehitetty automatisoimaan esimerkiksi tiedonkäsittelyä. Robotti pystyy suorittamaan kaikki ne toiminnot, jotka ihminen voi suorittaa käyttämällä tietokoneen hiirtä ja näppäimistöä. Ohjelmistorobotti voi käsitellä tietoa huomattavasti nopeammin kuin ihminen, ja kaikissa prosesseissa, esimerkiksi useissa Microsoftin ohjelmistossa kuten Wordissä ja Excelissä ohjelmistorobotti voi suorittaa prosessit ilman, että se

häiritsee tietokoneen käyttämistä. Robotit työskentelevät tarvittaessa kellon ympäri, eivätkä tee virheitä, jos ne on ohjelmoitu oikein ja ovat ajan tasalla.

Ohjelmistorobotiikka rakennetaan yrityksen jo olemassa olevan infrastruktuurin päälle, joten siitä ei koidu järjestelmään ylimääräisiä kustannuksia, esimerkiksi tietokoneiden tai uusien kaapelointien tekemisestä. Sillä on myös mahdollista yhdistää olemassa oleva vanha järjestelmä toimimaan uuden järjestelmän kanssa keräämällä vanhalta järjestelmältä tiedot näytönkaavinta ohjelmalla uuteen järjestelmään. (5.)

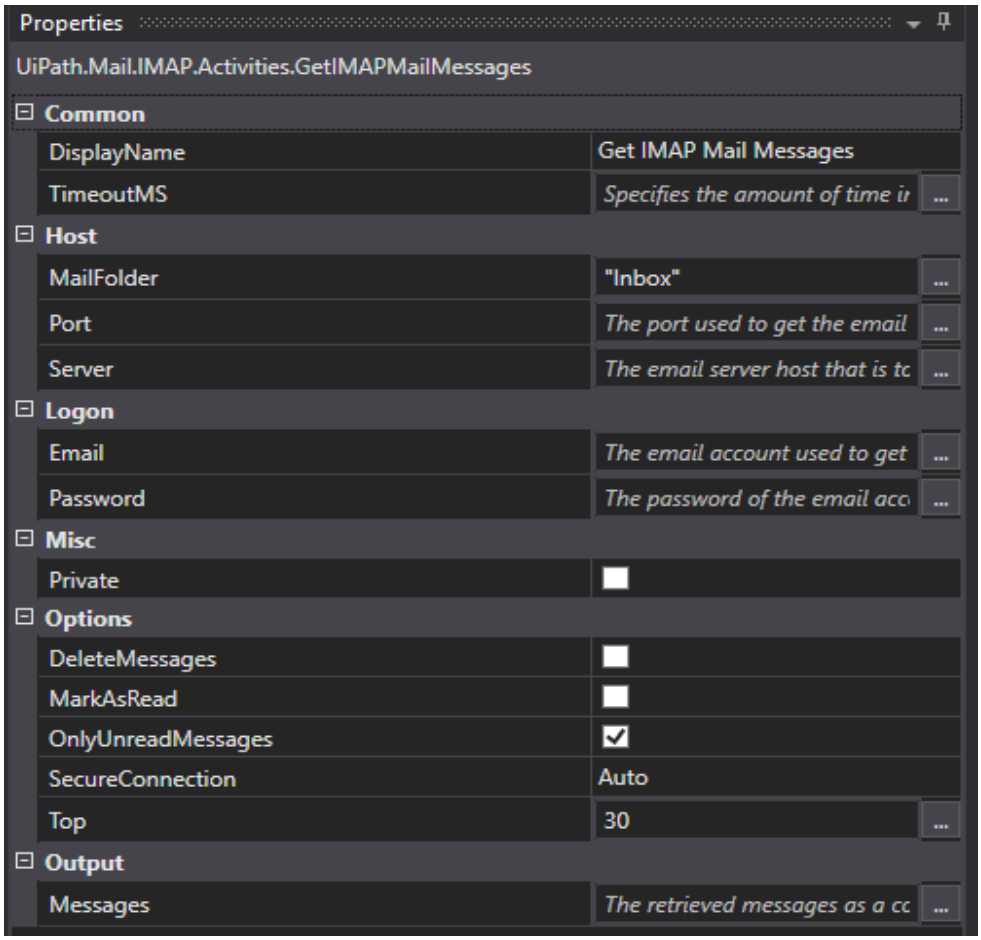
Robottien ohjelmointi tapahtuu graafisella käyttöliittymällä. Tämän ansiosta robotteja on helppo ohjelmoida, ja ohjelman kulkua on helpompi selittää ihmisille, joilla ei välttämättä ole ohjelmointitaitoja, esimerkiksi yrityksen myynti-ihmisillä. Koska ohjelmistorobotiikka on noussut suureen suosioon suurissa yrityksissä, on niille kehitetty monenlaisia ominaisuuksia ja toiminnallisuuksia. Kuvassa 1 on esimerkki siitä, mitä toiminnallisuuksia UiPath- alustalla tehtyyn robottiin voidaan lisätä.



Kuva 1. Osa UiPath-robotin toiminnallisuuksista.

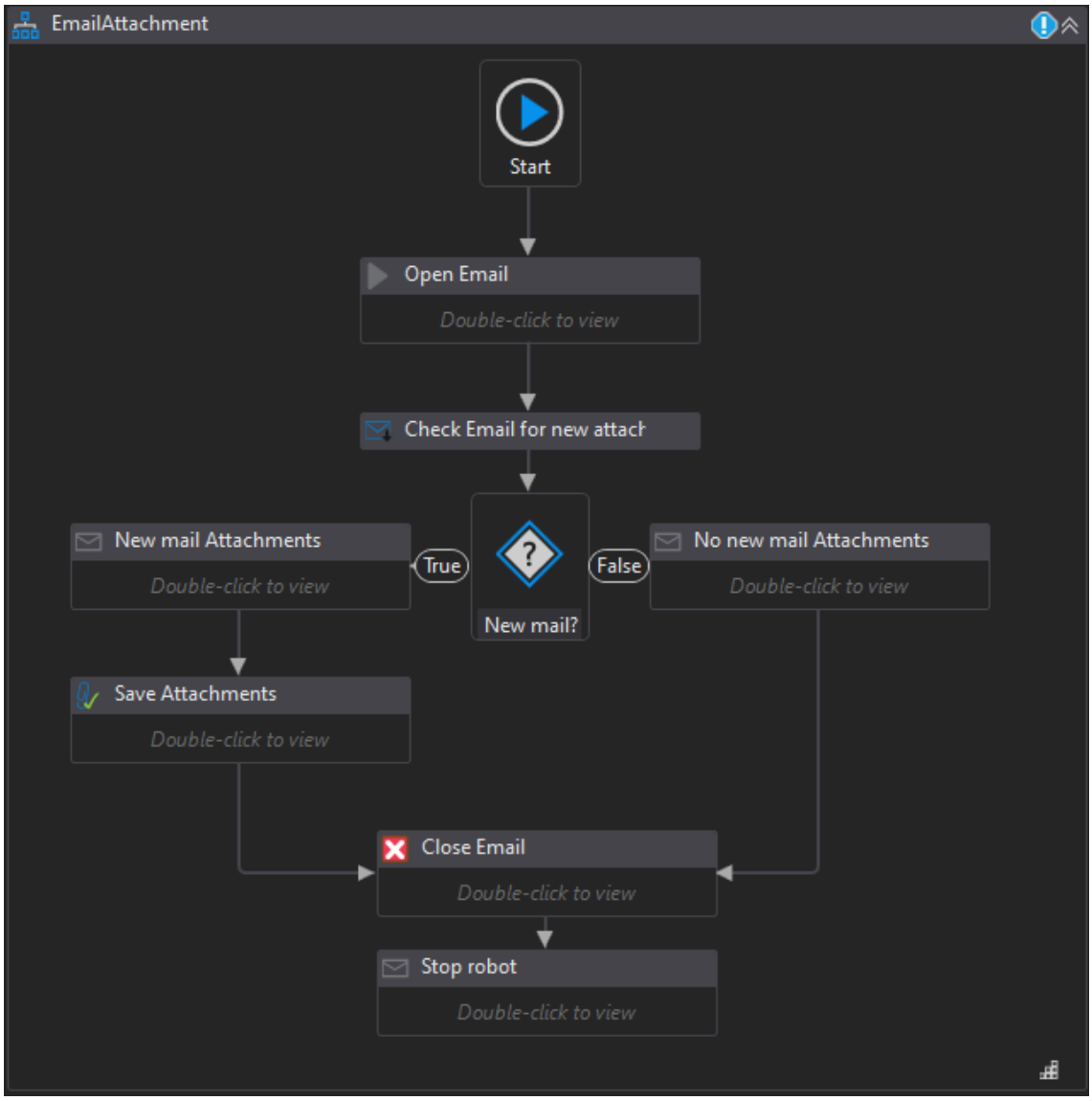
Riippuen mihin tarkoitukseen robottia ohjelmoidaan, on mahdollista rakentaa robotti vain kymmenellä eri toiminnolla. On myös mahdollista, että suuremmassa projektissa käytettyjen toiminnallisuuksien määrä nousee yli sataan.

Koska robottien ohjelmointi tapahtuu graafisesti ja kaikki toimintojen suorittamiseen käytetty koodi on piilotettu niin sanottujen rakennuspalikoiden sisään ja on poissa silmistä, on itse ohjelman toiminnan seuraaminen helppoa. Jokaiselle toimenpiteelle voidaan määrittää parametrejä, mikä on helppoa, koska ohjelmasta näkee, mitä parametrejä kullekin toimenpiteelle voidaan syöttää. Kuvassa 2 on esimerkki IMAP-viestin hakutoiminnon määrittelemiseen.



Kuva 2. IMAP-toiminnallisuuden parametrit.

Uipathissa on kaksi eri tapaa yhdistää robotin toiminnallisuuksia. Kuvasta 2 näkyy flowchart-tapa. Etuna tällaisessa rakennustavassa on, että selittäminen, mitä robotti tekee, on helppoa. Robotin kasvaessa suuremmaksi tämä rakennustapa selkeyttää robotin korjaus- ja muutostöitä, mikäli niitä tarvitsee tehdä.

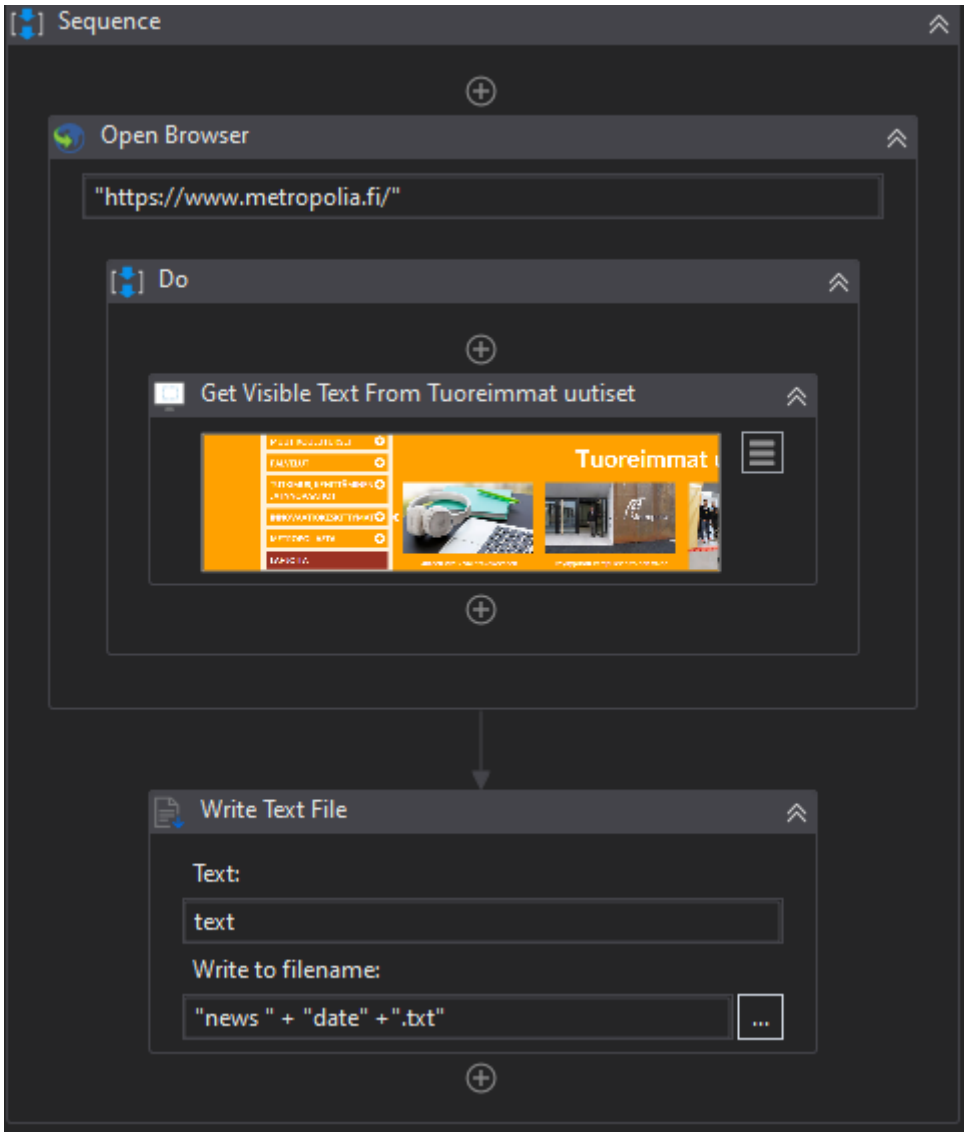


Kuva 3. Flowchart-esimerkki.

Kuvan 3 esimerkki ohjelma avaa sähköpostin, tarkistaa, onko sähköpostissa uusia liitteitä, ja tekee valinnan sen perusteella, onko sähköpostiin tullut uusia liitteitä vai ei. Jos

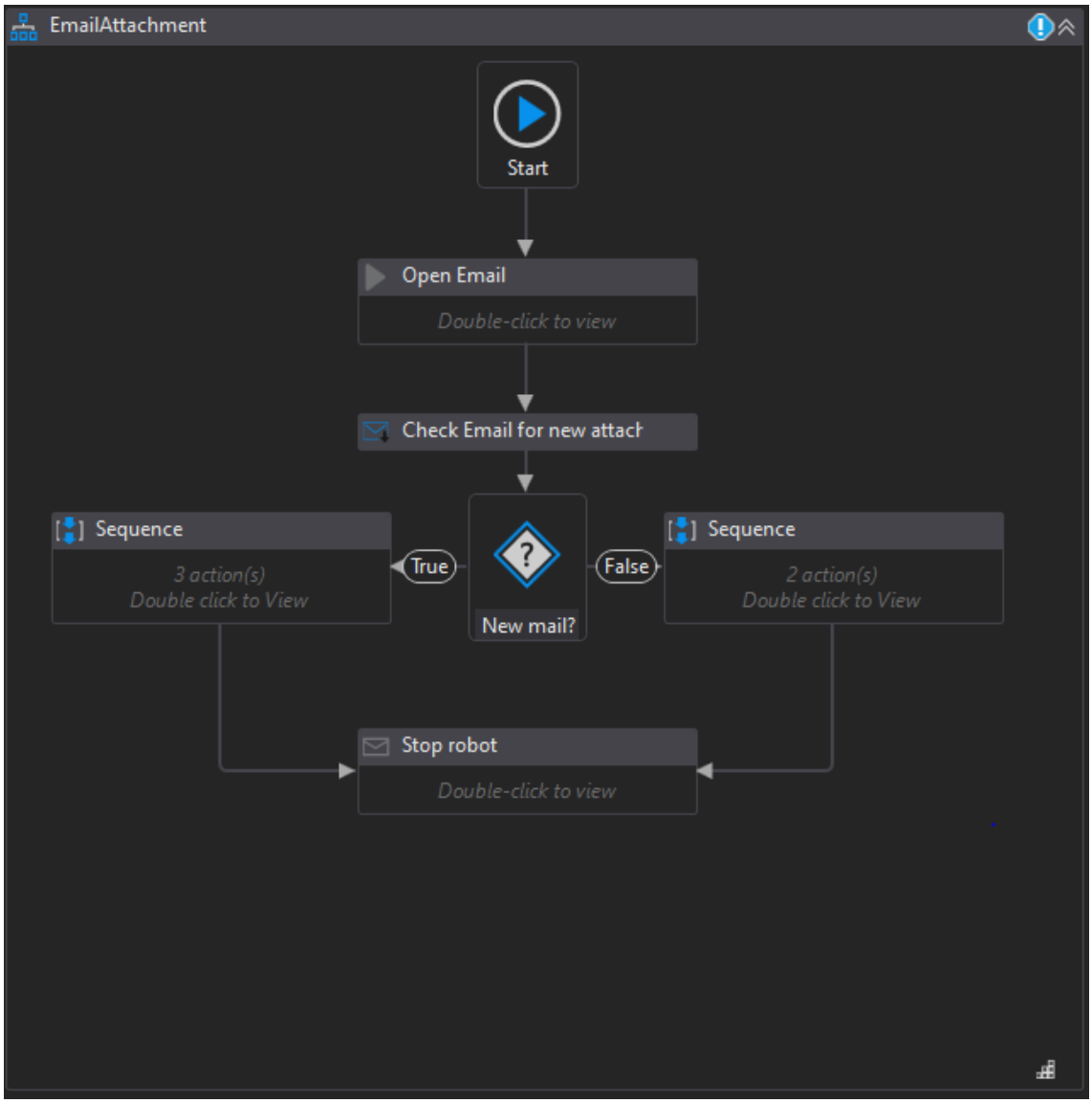
sähköpostissa on uusia liitteitä, se ilmoittaa käyttäjälle, että liitteet tallennetaan, sammuttaa sähköpostin ja lopettaa suorittamisen. Jos sähköpostissa ei ollut liitteitä, robotti ilmoittaa siitä käyttäjälle, sammuttaa sähköpostin ja lopettaa suorittamisen.

Toinen tapa on niin sanottu sequence, joka sopii yksinkertaisille roboteille, jotka suorittavat toiminnot aina samalla tavalla, eivätkä muutu suorituskerrasta toiseen. Kuvassa 4 on esimerkki, miten sequence voi avata nettiselaimen, mennä sivustolle ja valita tekstin, joka tallennetaan tekstitiedostoon, mikä nimetään news + päivämäärä nimellä.



Kuva 4. Sequence-esimerkki.

Nämä kaksi tapaa voidaan myös yhdistää, jolloin robotin kasvaessa sen rakenne pysyy selkeänä ja voidaan rakentaa monimutkaisempia robotteja, jotka ovat silti helppolukuisia ja siistejä. Kuvassa 5 on toiminnaltaan täysin sama robotti kuin kuvassa 3.



Kuva 5. Esimerkki flowchartin ja sequencen yhdistämisestä.

Vertaamalla kuvan 3 ja 5 robotteja huomaa, että jo pienenkin robotin kaaviossa voi huomata eron yhdistämällä kummatkin rakennusmetodit. Jos monipuolisen robotin rakenta-

miseen käytettäisiin pelkästään flowchart-metodia, joutuisi kaaviota liikuttamaan edestakaisin näytöllä, jotta sitä voisi seurata. Jos koko robotti rakennettaisiin sequence-metodilla, tulisi kyseisestä laatikosta niin sotkuinen, ettei sitä enää edes haluaisi lukea.

2.3 Käyttötarkoitus

Projektissa rakennettiin kaksi ohjelmistorobottia, joista toinen ohjelmoidaan helpottamaan sähköpostista liitteiden hakemista ja niiden tallentamista tietokoneelle. Toista robottia käytetään automatisoimaan kuvatiedostoista tekstin muuntamiseen tietokoneen ymmärrettävään muotoon käyttäen OCR-toiminnallisuutta

Ensimmäinen robotti, jonka tarkoituksena on hakea ja tallentaa liitteitä sähköpostista, kysyy käyttäjältä hakukriteerin, joka on tässä tapauksessa sana, joka esiintyy sähköpostiviestin otsikossa. Kun robotti on saanut hakukriteerin, se ottaa yhteyden sähköpostin palvelimeen IMAP-protokollalla ja käy läpi lukemattomaksi merkatut sähköpostit, jotka täyttävät hakukriteerit ja lataa niissä mahdollisesti olevat liitteet, jonka jälkeen se tallentaa ne tietokoneelle ennalta määriteltyyn paikkaan ja sammuttaa prosessin.

Toinen robotti avaa kansion, mihin aiempi robotti tallensi liitteet ja käy kansion läpi. Se ottaa kaikki kuvatiedostot ja syöttää ne OCR-ohjelmaan, joka kerää kuvista tekstin ja numerot ja tallentaa ne merkkijonomuuttujaan, minkä robotti kirjoittaa tekstitiedostoon ja tallentaa sen, jotta käyttäjä voi kopioida ja tarkistaa tiedot.

2.4 Ongelmat

Ohjelmistorobotiikan suurin ongelma on sen staattisuus. Ohjelmistorobotit seuraavat niille ohjelmoituja sääntöjä eivätkä kykene ajattelemaan tai tekemään valintoja niiden ulkopuolella. Hyvänä esimerkkinä voidaan käyttää verkkosivuilta päivittäin dataa hakevaa robottia. Jos verkkosivuja päivitetään ja sen HTML-koodi muuttuu, ei robotti välttämättä löydä enää halutun datan sijaintia sivustolla, koska näytönkaavinta perustuu HTML-koodista tiedon hakemiseen. Tässä tapauksessa robotti joudutaan ohjelmoimaan

uudestaan. Vaikka ohjelmistopäivityksistä tai sivujen uudistumisesta tulisi ilmoitus, on mahdollista, että sitä ei huomata ja robotti suorittaa sille ohjelmoidun prosessin väärin. Näytönkaavinta ohjelmistot eivät myöskään pysty hakemaan tietoa esimerkiksi kuvista, tai mistään, mikä ei ole suoraan maalattavissa hiirellä tai näppäimistöllä.

Ulkopuolisten virheiden sieto on toinen ongelma. Robotit hakevat tietoa esimerkiksi yrityksen tietokannoista tai sähköpostista, koska datan hakeminen tapahtuu ulkoisista lähteistä. Siinä voi kestää niin pitkään, että robotti ei tiedä, mitä sen pitäisi tehdä ja kaatuu. Tällaiset virheet voidaan korjata lisäämällä robotin kulkuun odotusaikoja, mutta se hidastaa robottien toimintaa myös silloin, kun ulkopuolisia ongelmia ei olisi.

Yllä mainittujen ongelmien takia ohjelmistorobotiikka vaatii paljon aikaa itse robotin kehittämiseen ja hienosäätämiseen, mutta myös robottien ylläpitoon joudutaan käyttämään huomattavia määriä työtunteja.

3 Koneoppiminen

Koneoppiminen on tekoälyn osa-alue, joka mahdollistaa ohjelmistojen kehittymisen niille syötetyn datan perusteella. Koneoppimisessa jäsennetään suuria määriä dataa käyttämällä algoritmeja. Mitä enemmän dataa algoritmille syötetään, sitä tarkemman ennusteen kyseinen algoritmi pystyy antamaan sille syötetystä datasta.

Koneoppimista käytetään nykypäivänä muun muassa sähköposteissa, mainonnassa, lääketieteessä ja monien älylaitteiden kehityksessä. Jokaiselle varmasti tuttua puuhaa sähköpostissa on merkata viesti roskapostiksi. Kun tämän toimenpiteen toistaa tarpeeksi monta kertaa, oppii sähköposti itse poistamaan samankaltaiset viestit roskapostikansioon. (6.)

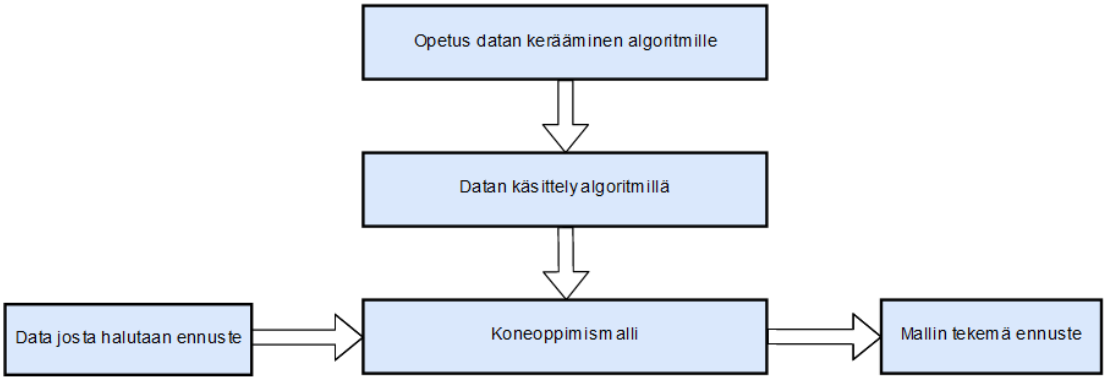
Mainonnassa koneoppimisen käytön huomaa parhaiten internettiä selatessa. Jos selaa verkkokauppojen tuotteita, huomaa, että hetken selaamisen jälkeen alkaa verkkokauppa suosittelemaan tuotteita samoista kategorioista, joita on juuri selannut. Myös tavallisesti

vieraillut verkkosivut voivat sisältää mainoksia, jotka on räätälöity käyttäjästä kerättyjen tietojen mukaan. (6.)

Lääketieteessä tunnetuin koneoppimisen esimerkki on IBM Watson. Watsonille voi esittää kysymyksen, se tulkitsee kysymyksen ja etsii sopivaa vastausta vertailemalla kuvailtuja oireita useista lähteistä. Tämän jälkeen Watson generoi mahdollisen vastauksen ja kerää lisää dataa, mikä vahvistaa arviota. Lopulta Watson antaa vastauksen käyttäjälle ja on oikeassa 71-prosenttisesti annetuista diagnooseista. (7.)

3.1 Toimintaperiaate

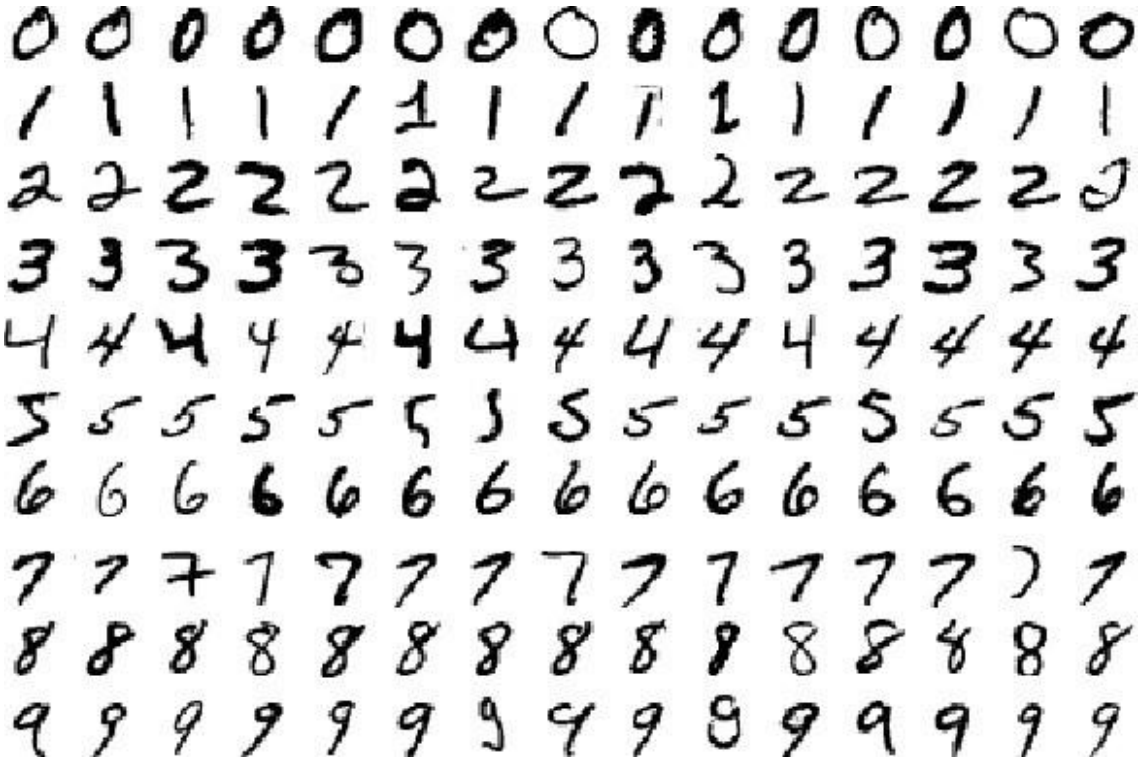
Koneoppiminen perustuu datan analysointiin ja sen sisältämän tiedon hyödyntämiseen määrätyn tehtävän suorittamisessa. Kuvasta 6 käy ilmi, miten koneoppimisen toiminta voidaan jakaa kolmeen osaan: datan keräämiseen, opettamiseen ja datan käsittelyyn.



Kuva 6. Koneoppimismallin toteutus ja käyttö.

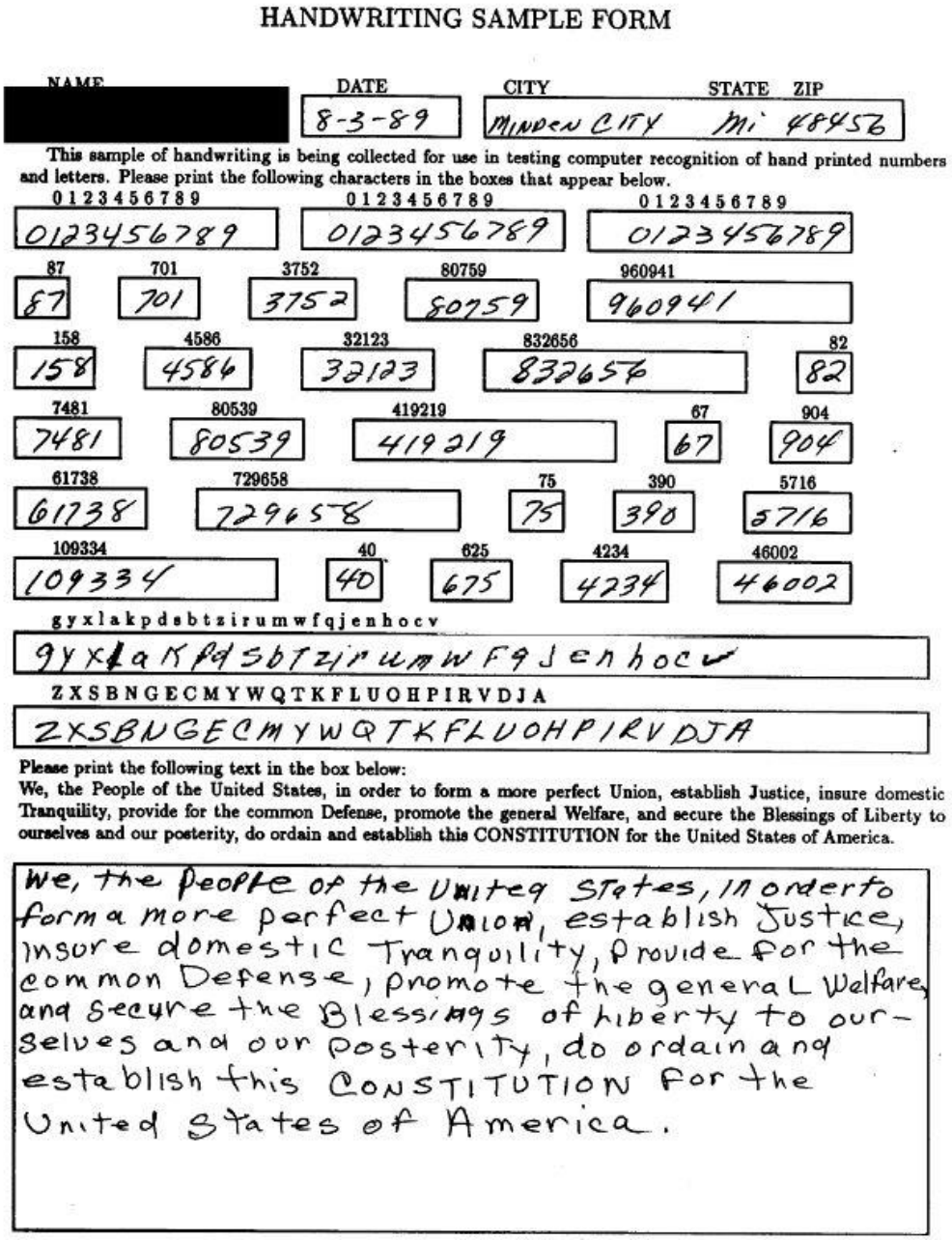
Kuten kuvasta 6 nähdään, koneoppimiseen tarvitaan dataa, joka toimii opetusmateriaalina, ja algoritmi, joka käy datan läpi ja muodostaa sen pohjalta itse koneoppimismallin, jolle voidaan syöttää opetusmateriaalin jälkeen dataa, josta halutaan ennuste.

Tärkein näistä vaiheista on datan kerääminen opetusmateriaaliksi. Mitä enemmän dataa algoritmi voi käydä läpi, sitä enemmän yhtenäisyyksiä se voi löytää datasta, ja sitä tarkempia kyseisen koneoppimismallin ennusteet ovat. Esimerkkinä voidaan käyttää tekstintunnistusalgoritmia, Algoritmille syötetään tuhansia eri tapoja kirjoittaa eri kirjaimia ja numeroita. Kuvassa 7 on esimerkki MNIST-datapaketista.



Kuva 7. Pieni otos MNIST-data paketista.

MNIST-data paketti sisältää 60 000 kuvassa 6 näkyvää käsin kirjoitettua numeroa algoritmin opettamiseen ja 10 000 esimerkkiä opetetun algoritmin testaamiseen. Kyseisellä opetusdatapaketilla on saatu opetettua konvoluutioneuroverkko tunnistamaan numerot 99.7%:n tarkkuudella. (8.) Tekstin opettamiseen algoritmille on olemassa datapaketti, johon on kerätty käsinkirjoitettuja lomakkeita 3600 eri kirjoittajalta. Ne sisältävät yhteensä 810 000 merkkiä. Kuvassa 8 on esimerkki yhdestä lomakkeesta. (9.)



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

0123456789 0123456789 0123456789

87 701 3752 80759 960941

87 701 3752 80759 960941

158 4586 32123 832656 82

158 4586 32123 832656 82

7481 80539 419219 67 904

7481 80539 419219 67 904

61738 729658 75 390 5716

61738 729658 75 390 5716

109334 40 625 4234 46002

109334 40 675 4234 46002

gyxlakpdsbtzirumwfqjenhocv

gyxlakpdsbtzirumwfqjenhocv

ZXSBNGECMYWQTKFLUOHPIRVDJA

ZXSBNGECMYWQTKFLUOHPIRVDJA

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Kuva 8. Esimerkki NIST-databasessa olevasta käsinkirjoitetusta lomakkeesta. (9.)

Kun opetusdataa on kerätty tarpeeksi, se pitää syöttää algoritmille, jota halutaan opettaa. Koneoppimiseen voidaan käyttää lähes mitä vain ohjelmointikieltä, mutta suosituimpia ohjelmointikieliä githubin suorittaman selvityksen mukaan ovat

- Python
- C++
- Javascript
- Java
- C#.

Jokaiselle listatulle kielelle on tehty kirjastoja, jotka helpottavat algoritmien rakentamisessa. Eniten käytetty kieli koneoppimiseen liittyvissä repositoreissa oli python ja syynä tälle voi olla, että Pythonille on rakennettu suuria määriä koneoppimisille ja datatieteelle tarkoitettuja kirjastoja. (10.)

3.2 Neuroverkot

Ihminen pystyy helposti tunnistamaan kirjoitettuja merkkejä. Tämä johtuu siitä, että ihmisen aivoissa on V1-niminen primäärinen näköaivokuori, joka sisältää 140 miljoonaa neuronia, jotka käsittelevät silmien kautta aivoihin päätyvän datan. Vaikka ihminen pystyykin tunnistamaan käsinkirjoitettuja merkkejä helposti, on vaikeaa kirjoittaa algoritminen tietokoneohjelma, joka pystyy päättelemään, mikä numero tai kirjain on kyseessä. (11.)

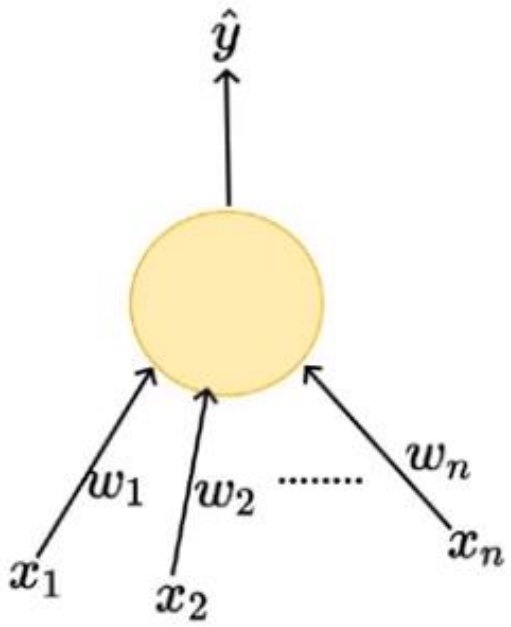
Koska neuroverkostot ovat erittäin monimutkaisia ja niiden käyttötarkoituksista riippuen voidaan rakentaa usealla eri tavalla, käydään työssä läpi vain niiden perustoimenpiteet ja itse neuronien toiminta.

Ensimmäinen keinotekoinen neuroni (McCulloch Pitts neuron) kehitettiin vuonna 1943, kun neurobiologian tutkijat Warren McCulloch ja Walter Pitts saivat idean hyödyntää tietokonetta imitoimaan ihmisen aivoja ratkaistaessa matemaattisia ongelmia. (12.)

Tietokoneella käytetyt keinotekoiset neuroverkostot rakennetaan erilaisiin funktioihin pohjautuvista neuroneista. Työssä käydään läpi perceptron, joka oli ensimmäinen tietokoneelle kehitetty keinotekoinen neuroni, ja sigmoid-neuroni, jota käytetään työssä opetetussa OCR-ohjelmistossa. Vaikka perceptron neuroverkkoja ei nykyisin käytetä paljoa, sen toiminnan ymmärtäminen helpottaa käsittämään sigmoidi-verkkoja. (11.)

Perceptron-neuroni

Frank Rosenblatt kehitti McCullock-Pitts-neuronista uuden mallin vuonna 1958, jossa kaikille sisään tuleville arvoille määritellään painoarvot. Neuronin nimeksi tuli perceptron, ja myöhemmin 1960-luvulla Minsky ja Papert kehittivät neuronin toiminnan sen nykyiseen muotoonsa. (15.) Neuronit perustuvat laskentaan, jossa niille määritellään sisääntulot, joiden perusteella se laskee todennäköisimmän vastauksen ja syöttää sen ulostulosta seuraavalle neuronille tai ulos neuroverkosta. Kuvassa 9 on esimerkki perceptronista.



Kuva 9. Malli perceptronista (15).

Kuvan 9 esimerkistä nähdään, että perceptronille voidaan antaa monta sisääntuloa, mutta sillä on aina vain yksi ulostulo Ŷ. Kuvan sisääntuloissa on myös x-arvojen lisäksi w, joka on kyseisen sisääntulon painoarvo ulostulon kannalta. Perceptronin ulostulo

määritellään sisääntuloon syötetyn arvon, painoarvon ja ennalta määritellyn raja-arvon perusteella b, kuten kuvassa 10 olevasta kaavasta voidaan huomata. (11.)

Perceptronin perusyhtälö määritellään seuraavasti:

$$\Sigma_J w_J x_J > b$$

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x - b \leq 0 \\ 1 & \text{if } w \cdot x - b > 0 \end{cases}$$

Kuva 10. Yksinkertainen perceptronin ulostulon yhtälö (11.)

Kuvan 9 kaavassa x on sisääntuloon syötetty arvo, ja w on sille annettu painoarvo. Näitä arvoja verrataan arvoon raja-arvoon b. Jos b määritellään suureksi, kasvaa todennäköisyys, että perpectroni antaa ulostuloksi arvon 1. (11.)

Perceptronien ongelmana on, että ne voivat antaa ulostulossa vain arvon 0 tai 1, ja pienetkin muutokset sisääntulevissa arvoissa tai painoarvoissa vaikuttaa lopputulokseen huomattavasti. Tämä asia korjattiin kehittämällä sigmoid-neuronit. (11.)

Sigmoid-neuroni

Sigmoidisen neuronin voi esittää kuvan 8 mallin mukaisesti. Erona perceptroniin on se, että sisääntulon ja ulostulon arvot voivat olla esimerkiksi 0,250 tai 0,550. Tämän mahdollistaa funktion derivoitavuus. Sigmoidin funktio määritellään seuraavasti:

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

Funktion arvo z korvataan perceptronin yhtälöllä, jolloin sigmoidin kaavaksi saadaan:

$$\frac{1}{1 + e^{(-\sum w_i x_i - b)}}$$

Aivan kuten perceptroninkin kaaviossa, myös sigmoidiset neuronit laskevat ulostulon summaamalla sisääntulo arvot x ennalta määriteltyihin painoarvoihin w.

3.3 Konvoluutioneuroverkon toiminta

Konvoluutioneuroverkoissa on useita kerroksia. Ensimmäinen kerros neuroverkolla on syöttökerros, johon lähetetään dataa. Syöttökerroksen jälkeen on jokin määrä niin sanottuja piilokerroksia, eli konvoluutio ja yhdistämiskerroksia, ja lopuksi on täysin yhdistetty kerros. Eri kerrosten toiminnan selkeyttämiseksi käydään seuraavaksi läpi työssä neuroverkostojen toiminnan ymmärtämiseen käytetyn OCR-ohjelman lähdekoodia. Alla olevasta esimerkkikoodista 1 näkyvät kirjastot, joita ohjelma käyttää.

```
import cPickle
import gzip
import numpy as np
import theano
import theano.tensor as T
```

Esimerkkikoodi 1. Ohjelman sisältämät kirjastot.

Koska neuroverkot pohjautuvat matemaattisiin yhtälöihin ja niiden prosessointiin, on tärkeää, että käytetään oikeita kirjastoja, esimerkiksi numpya, ja theanoa.

Numpy on pythonille kirjoitettu kirjasto, joka mahdollistaa moniulotteisten taulukoiden ja matriisien käsittelyn. Se sisältää myös useita matemaattisia funktioita, joilla voidaan hyödyntää kyseisiä taulukoita ja matriiseja. (13.)

Theano on pythonille kirjoitettu kirjasto, jolla voidaan määritellä, optimoida ja arvioida matemaattisia lausekkeita, jotka sisältävät moniulotteisia matriiseja. Theanolla voidaan hyödyntää tietokoneiden näytönohjaimia prosessoreiden sijaan, minkä ansiosta sillä voidaan suorittaa data intensiivisiä tai monimutkaisia matemaattisia yhtälöitä nopeasti. (14.)

Data, jota neuroverkoille syötetään niiden opetusvaiheessa, on erittäin tärkeää tarkkojen lopputulosten saamiseksi algoritmilla. Tähän käytetään aiemmin mainittua MNIST-datapakettia, joka on havaittu hyväksi opetusmateriaaliksi. Esimerkkikoodi 2 sisältää datan lataamisen.

```
def load_data_shared(filename="../data/mnist.pkl.gz"):
f = gzip.open(filename, 'rb')
training_data, validation_data, test_data = cPickle.load(f)
f.close()
```

Esimerkkikoodi 2. MNIST-datapaketin lataaminen ohjelmalle.

Konvoluutioneuroverkostot rakentuvat erilaisista kerroksista sen mukaan, mitä niillä halutaan tehdä. Pienten kuvien kuten esimerkiksi MNIST-datapaketin sisältämät 28x28 pikselin kuvat ovat mustavalkoisia. Ne on mahdollista käsitellä tavallisella täysin yhdistetyllä neuroverkolla, missä kaikki aiemman kerroksen neuronit on yhdistetty seuraavan kerroksen neuroneihin, koska syöttökerroksentulo jälkeen neuroneilla olisi vain 784 painoarvoa.

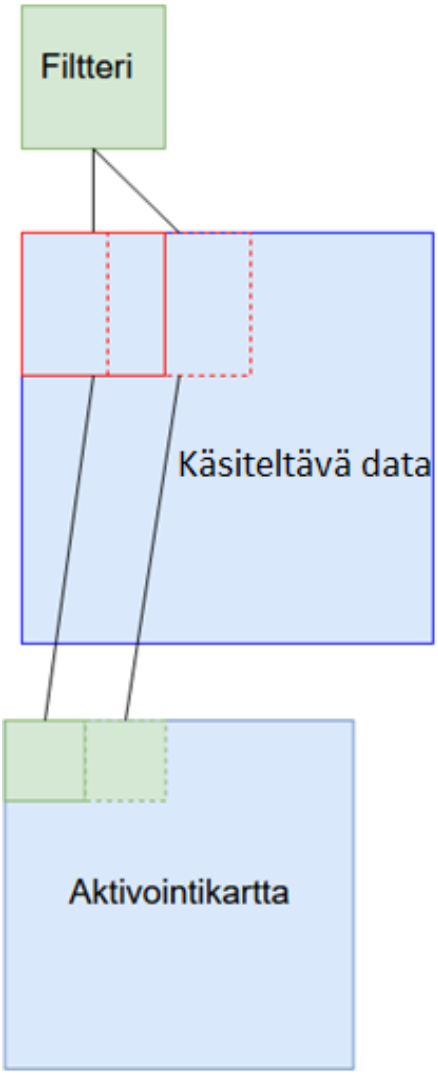
Mutta suuremmissa kuvissa, esimerkiksi 300x300x3-kuva, eli 300x300 pikseliä ja 3 väriä, kasvaisi painoarvojen määrä jo 270 000. Tällaisen määrän käsittelemiseen tietokoneella menisi erittäin pitkään edetä neuroverkossa. Tämän takia suurempien kuvien käsittelemiseen neuroverkoilla käytetään usein eri kerroksia, jotka suorittavat kuville erilaisia toimenpiteitä ja toimivat eräänlaisina filttereinä. Esimerkkikoodissa 3 määritellään yhdistetty konvoluutio- ja yhdistämiskerros. (15.)

```
def __init__(self, filter_shape, image_shape, poolsize=(2, 2),
                 activation_fn=sigmoid):

        self.filter_shape = filter_shape
        self.image_shape = image_shape
        self.poolsize = poolsize
        self.activation_fn=activation_fn

        n_out = (filter_shape[0]*np.prod(filter_shape[2:])/np.prod(poolsize))
        self.w = theano.shared(
            np.asarray(
                np.random.normal(loc=0, scale=np.sqrt(1.0/n_out), size=fil-
ter_shape),
                dtype=theano.config.floatX),
            borrow=True)
        self.b = theano.shared(
            np.asarray(
                np.random.normal(loc=0, scale=1.0, size=(filter_shape[0],)),
                dtype=theano.config.floatX),
            borrow=True)
        self.params = [self.w, self.b]

    def set_inpt(self, inpt, inpt_dropout, mini_batch_size):
        self.inpt = inpt.reshape(self.image_shape)
        conv_out = conv.conv2d(
            input=self.inpt, filters=self.w, filter_shape=self.filter_shape,
            image_shape=self.image_shape)
        pooled_out = downsample.max_pool_2d(
            input=conv_out, ds=self.poolsize, ignore_border=True)
        self.output = self.activation_fn(
            pooled_out + self.b.dimshuffle('x', 0, 'x', 'x'))
        self.output_dropout = self.output
```

Esimerkkikoodi 3. Konvoluutio- ja yhdistämiskerroksen määrittäminen.

Konvoluutiokerros käy kuvan läpi 3x3 pikselin tarkkuudella käyttäen filttereitä. Se siirtyy aina yhdellä pikselillä eteenpäin kuvassa ja muodostaa niin sanotun aktivointikartan, joka sisältää kuvassa olevia ominaisuuksia.

Kun konvoluutiokerros käy kuvaa lävitse, se vertaa kuvan 3x3-pikseliä sen 3x3-filtteriin, ja muodostaa tiedon perusteella kartan, josta voidaan päätellä, mitä kyseisessä kuvan kohdassa on. Siinä voi olla esimerkiksi viiva, jos kyseessä on tekstintunnistus. Kuvassa 11 on havainnollistettu tapahtumaa.



Kuva 11. Konvoluutiokerroksen toiminta havainnollistettuna.

Yhdistämiskerros nimensä mukaisesti yhdistää asioita. Sen tarkoituksena on pienentää käsiteltävien parametrien määrää neuroverkossa. Yleensä näitä kerroksia laitetaan neuroverkoissa jokaisen konvoluutiokerroksen väliin.

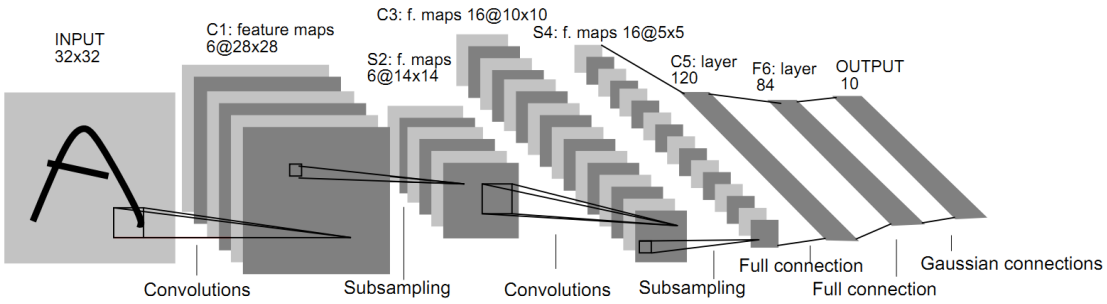
Ohjelman yhdistämiskerros käyttää niin sanottua maksimifunktiota ja tekee sen 2x2-alueelle. Tämä tarkoittaa, että se ottaa neljä neuronia, valitsee niistä suurimman arvon ja päästää sen läpi seuraavaan kerrokseen. Tämä toimenpide laskee tarvittavaa laskentatehoa tulevassa kerroksessa 75 %. (15.)

Viimeinen ja tärkein kerros konvoluutioneuroverkoissa on täysin yhdistetty neuroverkosto. Esimerkkikoodissa 4 määritellään täysin yhdistetty kerros.

```
def __init__(self, n_in, n_out, activation_fn=sigmoid, p_dropout=0.0):
        self.n_in = n_in
        self.n_out = n_out
        self.activation_fn = activation_fn
        self.p_dropout = p_dropout
        self.w = theano.shared(
            np.asarray(
                np.random.normal(
                    loc=0.0, scale=np.sqrt(1.0/n_out), size=(n_in, n_out)),
                dtype=theano.config.floatX),
            name='w', borrow=True)
        self.b = theano.shared(
            np.asarray(np.random.normal(loc=0.0, scale=1.0, size=(n_out,)),
                       dtype=theano.config.floatX),
            name='b', borrow=True)
        self.params = [self.w, self.b]

    def set_inpt(self, inpt, inpt_dropout, mini_batch_size):
        self.inpt = inpt.reshape((mini_batch_size, self.n_in))
        self.output = self.activation_fn(
            (1-self.p_dropout)*T.dot(self.inpt, self.w) + self.b)
        self.y_out = T.argmax(self.output, axis=1)
        self.inpt_dropout = dropout_layer(
            inpt_dropout.reshape((mini_batch_size, self.n_in)), self.p_drop-
out)
        self.output_dropout = self.activation_fn(
            T.dot(self.inpt_dropout, self.w) + self.b)
```

Esimerkkikoodi 4. Täysin yhdistetyn neuroverkkokerroksen määrittäminen

Tämä kerros on niin sanottu tavallinen neuroverkko toisin kuin konvoluutiokerroksen neuronit, jotka ottavat pienen määrätyn osan aiemman kerroksen ulostuloista, täysin yhdistetyn kerroksen jokainen neuroni yhdistyy jokaiseen aiemman kerroksen ulostuloon. Tämän kerroksen jälkeen kaikki käsitelty data johdetaan viimeisten neuroneiden läpi, jotka päättävät lopullisen lopputuloksen, mitä kuvassa on, ja syöttävät vastauksen ulostuloon. Konvoluutioneuroverkko koostuu usein monesta edellä mainitusta kerroksesta. Alla olevassa kuvassa 12 on esimerkki 1990-luvulla kehitetystä LeNet-5-verkosta.



Kuva 12. LeNet-5-konvoluutioneuroverkko.

LeNet-5 on yksinkertainen konvoluutioneuroverkko. Se sisältää syöttökerroksen, johon esimerkissä on laitettu käsin kirjoitettu kirjain A. Syöttökerroksen jälkeen tulee ensimmäinen konvoluutiokerros C1, joka tekee sille syötetystä kuvasta 6 aktivointikarttaa, jotka ovat 28x28x6-kokoisia. Konvoluutiokerros C1 syöttää tiedot yhdistyskerrokseen S2, joka yhdistää 28x28x6-aktivointikartoista 4 pikselin arvot yhdeksi, jotta se saa 6 kappaletta 14x14x6-pikselin karttaa. Tämä toimenpide toistetaan toisen kerran toisilla parametreillä kerroksissa C3 ja S4, jonka jälkeen tiedot lähetetään täysin yhdistettyyn konvoluutiokerrokseen C5, jonka neuronit yhdistyvät kaikkiin S4-kerroksen 400 neuroniin. C5-kerroksessa lasketaan 120 1x1-kokoista aktivointikarttaa. F6-kerros on tavallinen täysin yhdistetty kerros: se sisältää 84 neuronia, jotka yhdistyvät aiemman kerroksen kaikkiin 120 neuroniin. Lopulta neuroverkko loppuu output-kerrokseen, missä neuroverkon eniten uskoma numero valitaan painoarvojen perusteella, ja se syötetään ulostulosta ulos.

Neuroverkon opettaminen tapahtuu syöttämällä sille testidataa. Termiä epoch käytetään kuvaamaan yhtä kokonaista opetuskierrosta. Esimerkkikoodissa on tähän asti käytetty MNIST-datapaketin sisältämiä valmiiksi tehtyjä kuvia. Projektissa kuitenkin koitettiin opettaa neuroverkkoa tavallisilla dokumenteista skannatuilla kuvilla, jotka muunnettiin mustavalkoisiksi, Esimerkkikoodeissa 5 ja 6 käydään läpi itse neuroverkon kouluttaminen ja testaaminen.

```
def SGD(self, training_data, epochs, mini_batch_size, eta,
validation_data, test_data, lmbda=0.0):
training_x, training_y = training_data
validation_x, validation_y = validation_data
test_x, test_y = test_data
```

Esimerkkikoodi 5. Neuroverkon opettaminen gradientti-menetelmällä.

Gradientti-menetelmä SGD (Stochastic Gradient Descent) on optimointitekniikka kone- ja syväoppimiselle. Se valitsee koko oppimateriaalista määrätynsuuruisen osan kuvia, ja syöttää ne neuroverkolle. Jokaisen opetuskierroksen välissä se muuttaa datan painoarvoja, kunnes päästään haluttuun lopputulokseen. (11.)

```
best_validation_accuracy = 0.0
for epoch in xrange(epochs):
for minibatch_index in xrange(num_training_batches):
iteration = num_training_batches*epoch+minibatch_index
if iteration % 1000 == 0:
print("Training mini-batch number {0}".format(iteration))
cost_ij = train_mb(minibatch_index)
if (iteration+1) % num_training_batches == 0:
validation_accuracy = np.mean(
[validate_mb_accuracy(j) for j in xrange(num_validation_batches)])
print("Epoch {0}: validation accuracy {1:.2%}".format(
epoch, validation_accuracy))
```

Esimerkkikoodi 6. Neuroverkon testaaminen halutulla datalla.

Algoritmin tarkkuuden testaaminen tapahtuu lataamalla ohjelmaan kuvia datasetistä, joka sisältää myös oikean vastauksen siihen, mitä kuvassa pitäisi olla. Ohjelma käy kuvat läpi ja vertailee niitä oikeisiin vastauksiin ja ilmoittaa onnistumisprosentin käyttäjälle.

3.4 Hyödyntäminen

Insinöörityössä koneoppimista hyödynnettiin keräämään tietoa kuvatiedostoista, joihin ohjelmistorobotiikan näytönkaavintatyökalut eivät onnistu. Tähän tarkoitukseen käytettiin konvoluutio-neuroverkostoa, jonka tarkoituksena oli käydä kuvatiedostot läpi, poimia niistä oleelliset tiedot, ja tallentaa ne koneen ymmärrettävään muotoon.

Työssä päädyttiin käyttämään konvoluutio-neuroverkostoa, jolle opetettiin eri numeroita ja kirjaimia 28x28 pikselin mustavalkoisista kuvista, näiden kuvien sisältämien merkkien tunnistamiseen ohjelma onnistui hyvin. Ongelmana tällä neuroverkolla oli se, että kun sille syötettiin kuva, jossa oli yhden merkin sijaan tekstiä ja useita numeroita lähekkäin, ei ohjelma onnistunut erottamaan kuvasta selkeästi eri merkkejä, ja ulostuleva teksti oli sekasotkuista ja käyttökelvotonta. Ohjelman toiminta ei parantunut huomattavasti, vaikka sille syötettiin enemmän opetusdataa, ja huomattiin, että sillä on hankalaa muuntaa kuvatiedostojen tekstiä käytettävään muotoon. Ei ollut realistista opettaa verkkoa, koska opetusdatan määrä, mitä sille voitiin tehdä, oli rajoitettu olemassa olevien tiedostojen määrän vuoksi.

Koska konvoluutioneuroverkon opettaminen halutun lopputuloksen saamiseksi ei ollut realistista, päädyttiin projektissa käyttämään UiPathiin sisäänrakennettua OCR-ohjelmistoa, joka on valmiiksi opetettu tunnistamaan kuvista tekstiä. Koska konvoluutioneuroverkon opettaminen epäonnistui, tutkittiin, minkä takia sen tarkkuus ei parantunut suurten kuvien tunnistamisessa, vaikka se tunnistaakin hyvin pieniä kuvia, joissa on vain yksi merkki. Selvisi, että konvoluutioneuroverkot ovat hyviä työkaluja, jos tahdotaan tunnistaa kuvista esimerkiksi eläimiä, pieniä määriä merkkejä, jotka eivät ole lähekkäin, tai muita yksittäisiä kappaleita. Konvoluutioneuroverkot keräävät kuvasta filttereiden avulla ominaisuuksia ja kuvioita. Esimerkiksi jos algoritmille syötetään 100 kuvaa, missä on lintuja, oppii se tunnistamaan, että kuvassa on lintu, koska se tietää, että linnuilla on sulkia ja ne näyttävät tietynlaiselta.

Konvoluutioneuroverkot eivät sovellu kokonaisten kirjoitettujen lauseiden tunnistamiseen, koska sen neuronit eivät muista, mitä ne käsittelivät viimeksi eivätkä täten onnistu yhdistämään erinäisiä kirjaimia toisiinsa muodostaakseen lauseita.

Työn senhetkisessä vaiheessa ei ollut enää realistista opetella uutta neuroverkkoarkkitehtuuria, mutta vaihtoehtoja konvoluutioverkoston korvaamiseksi silti etsittiin jatkokehitystä varten. Tutkiessa eri neuroverkkojen käyttötarkoituksia huomattiin, että esimerkiksi Google käyttää toistuvia neuroverkkoja Google Translate-palvelussa. (16.)

Toistuvissa neuroverkoissa neuronit muistavat aiemman arvon, mikä niillä oli, ja käyttävät tätä arvoa laskiessaan seuraavaa arvoa. Tämän ansiosta, kun verkosto tunnistaa kuvissa olevaa tekstiä, se osaa yhdistää kirjaimet sanoiksi ja lisätä välilyönnit sanojen väliin ulostulevaan tekstiin. Toistuvat neuroverkot on kehitetty käsittelemään peräkkäistä dataa. Tämä tarkoittaa, että kun sille syötetään kuva, missä on lause, se käy kuvan lävitse samalla tavalla kuin konvoluutioneuroverkko, mutta jokaisen kirjaimen kohdalla se käyttää aiempaa kirjainta painoarvojen määrittämiseen. (16.)

Koska uuden arkkitehtuurin ja ohjelmiston opetteleminen ei enää ollut mahdollista, löytöjä, joita tehtiin, suunniteltiin käytettäväksi mahdollisessa jatkokehityksessä.

4 Yhteenveto

Insinöörityössä käytiin läpi ohjelmistorobotiikan hyviä ja huonoja puolia ja miten sen huonoja puolia voidaan tukea koneoppimisella. Työn tavoitteena oli toteuttaa ohjelmistorobotti toimimaan ihmisen valvonnassa ja automatisoimaan puuduttavia työtehtäviä kuten tiedostojen lataamista ja tekstin poimimista kuvista. Lisäksi työssä perehdyttiin neuroverkostojen toimintaan ja niiden opettamiseen.

Ohjelmistorobotiikan toteuttaminen työtehtävään onnistui, ja tuloksena työprosessi ihmiselle helpottui. Myös neuroverkostot ja niiden toiminta tulivat tutuksi työn aikana, vaikka itse neuroverkoston opettaminen epäonnistui puutteellisen opetusmateriaalin takia. Ohjelmistorobotissa käytettiin lopulta avoimen lähdekoodin valmiiksi koulutettua robottia.

Työssä tehty ohjelmistorobotti kehitettiin senhetkisten tarpeiden mukaan toimivaksi kokonaisuudeksi, ja jos ohjelmistoa kehitetään tulevaisuudessa, voidaan harkita valmiiksi koulutetun OCR-ohjelman vaihtamista itse koulutettuun LSTM-arkkitehtuuriin pohjautuvalla OCR-ohjelmistolla.

Lähteet

1 Thothadri, Arvind. 2018. Why You Should Consider a Career in RPA. Verkkoaineisto <https://www.automationanywhere.com/blog/changing-the-world-with-automation/why-you-should-consider-a-career-in-rpa>. Luettu 15.8.2019.

2 Ostdick, Nick. 2016. The Evolution of Robotic Process Automation (RPA): Past, Present, and Future. Verkkoaineisto <https://www.uipath.com/blog/the-evolution-of-rpa-past-present-and-future>. Luettu 2.8.2019.

3 Welsh, John. 2019. What The History Of RPA Technology Says About Its Future. Verkkoaineisto https://globalpayrollassociation.com/blogs/technology/what-the-history-of-rpa-technology-says-about-its-future>. Luettu 21.8.2019.

4 Gartner Says Worldwide Robotic Process Automation Software Market Grew 63% in 2018. 2019. Verkkoaineisto. EGHAM, U.K. <https://www.gartner.com/en/newsroom/press-releases/2019-06-24-gartner-says-worldwide-robotic-process-automation-sof>. Luettu 13.7.2019.

5 What is Robotic Process Automation? 2019. Verkkoaineisto <https://www.uipath.com/rpa/robotic-process-automation>. Luettu 2.7.2019.

6 Hurwitz, Judith & Kirsch, Daniel. 2018. Machine Learning for dummies. 111 River St.Hoboken. John Wiley & Sons, inc.

7 Wang, James. 2017. How Much Artificial Intelligence Does IBM Watson Have? Verkkoaineisto <https://ark-invest.com/research/ibm-watson>. Luettu 12.6.2019.

8 Lecun, Yann THE MNIST DATABASE. Verkkoaineisto <http://yann.lecun.com/exdb/mnist/>. Luettu 8.10.2019.

9 NIST Special Database 19. 2019. Verkkoaineisto <https://www.nist.gov/srd/nist-special-database-19>. Luettu 11.10.2019.

10 Elliot, Thomas. 2019. The State of the Octoverse: machine learning. Verkoaineisto <https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/>. Luettu 18.6.2019.

11 Nielsen, Michael. 2019. Neural Networks and Deep Learning.E-kirja.

12 Kumar, Niranjan. 2019. McCulloch Pitts Neuron – Deep Learning Building Block. Verkkoaineisto <https://medium.com/hackernoon/mcculloch-pitts-neuron-deep-learning-building-blocks-7928f4e0504d>. Luettu 22.9.2019.

13 Numpy Reference. 2019. Verkkoaineisto <https://numpy.org/devdocs/reference/index.html>. Luettu 22.9.2019.

14 Theano. 2017. Verkkoaineisto <http://deeplearning.net/software/theano/>. Luettu 22.9.2019.

15 CS231n Convolutional Neural Networks for Visual Recognition. Verkkoaineisto <http://cs231n.github.io/convolutional-networks/>. Luettu 20.9.2019.

16 Kostadinov, Simeon. 2017. Verkkoaineisto <https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7>. Luettu 7.11.2019.