Tampereen ammattikorkeakoulu

Rakennusautomaatio ja koneoppinen

Säätökaavioiden automaattinen tulkinta

Teemu Forsten

OPINNÄYTETYÖ
Joulukuu 2021

Dataosaamisen ja tekoälyn ylempi tutkinto-ohjelma, ylempi AMK

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Dataosaamisen ja tekoälyn ylempi tutkinto-ohjelma, ylempi AMK

FORSTEN, TEEMU:
Rakennusautomaatio ja koneoppiminen
Säätökaavioiden automaattinen tulkinta

Opinnäytetyö 113 sivua, joista liitteitä 1 sivu
Joulukuu 2021

Rakennusautomaatiojärjestelmällä ohjataan kiinteistöjen LVISJ-laitteistoja. Opinnäytetyön tarkoituksena oli automatisoida rakennusautomaatiojärjestelmissä käytettävien grafiikkakuvien piirtäminen kuvantunnistussovelluksen avulla. Opinnäytetyö toteutettiin case-tutkimuksena rakennusautomaatiojärjestelmiä toimittavalle Fidelix Oy:lle.

Kuvantunnistukseen käytetään tekoälyyn pohjautuvia menetelmiä. Tekoälytutkimus on tällä hetkellä painottunut vahvasti koneoppimiseen. Koneoppiminen pohjautuu pitkälti olemassa olevaan dataan ja siinä olevien säännönmukaisuuksien etsimiseen algoritmien avulla. Koneoppimisen algoritmien tuloksena muodostuu malli, jonka avulla mallinnetaan datan mukaista ilmiötä.

Tässä tutkimuksessa käytetty data koostuu säätökaavioista. Säätökaaviossa kuvataan rakennusautomaationjärjestelmään liitettävän laitteiston toimintaperiaatetta. Säätökaavioissa olevat eroavaisuudet aiheuttivat haasteita kuvantunnistusalgoritmeille. Lisäksi tutkimuksessa käytetyn datan vähyys aiheutti haasteita lopputulokseen. Säätökaavioista pystyttiin tunnistamaan kyseessä ollut laitteisto, mutta yksittäisten laitteiden semantiikkaa ei saatu selville käytetyillä menetelmillä.

Tämän tutkimuksen data oli painottunut muutaman suuren suunnittelutoimiston säätökaavioihin. Jatkotutkimusta varten on dataa kerättävä merkittävästi enemmän ja sen tulisi olla monipuolisempaa. Kuvantunnistussovelluksen kehitystä helpottaisi säätökaavioiden standardointi ja yhtenäistäminen.

Asiasanat: rakennusautomaatio, tekoäly, koneoppiminen, kuvantunnistus, säätökaavio

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree Programme in Data Expertise and Artificial Intelligence

FORSTEN, TEEMU:
Building Automation and Machine Learning
Automatic Interpretation of PI Diagrams

Master's thesis 113 pages, appendices 1 page
December 2021

The property's HVAC system is controlled by a building automation system. The purpose of the thesis was to automate the drawing of graphics images using an image recognition application. Graphic images are used in building automation systems as an interface. The thesis was carried out as a case study for Fidelix Oy, which supplies building automation systems.

Artificial intelligence-based methods are used for image recognition. Artificial intelligence research is currently strongly focused on machine learning. Machine learning is based on existing data. Complex algorithms look for regularities in the data. As a result of machine learning algorithms, a model is formed. The model is used to interpret the phenomenon according to the data.

The data used in this thesis consist of PI diagrams. The PI diagram describes the operating principle of the equipment connected to the building automation system. Differences in the PI diagrams caused challenges to image recognition algorithms. In addition, the small amount of data used in the thesis caused challenges to the outcome. The correct system could be identified from the PI diagrams, but the semantics of the individual devices could not be ascertained by the methods used.

The data from this thesis were focused on the PI diagrams drawn by a few large design companies. Significantly more data needs to be collected for further research and it should be more diverse. The development of an image recognition application would be facilitated by the standardization and unification of PI diagrams.

Key words: building automation, artificial intelligence, machine learning, image recognition, PI diagram

SISÄLLYS

1 JOHDANTO 7
1.1 Rakennusautomaatiourakoinnin maailma 7
1.2 Ongelma, tavoite ja rajaukset 8
1.3 Aikaisempi tekoälytutkimus rakennusautomaatioalalla 10
2 RAKENNUSAUTOMAATIO 13
2.1 Rakennusautomaatiojärjestelmän rakenne ja tehtävä 13
2.2 Fidelix 15
2.3 Fidelix-järjestelmän grafiikka ja ohjelmat 16
3 TEKOÄLY JA KONEOPPIMINEN 19
3.1 Johdotus tekoälyyn 19
3.2 Tekoälystä koneoppimiseen 19
3.3 Ohjattu oppiminen 22
3.3.1 Luokittelu 23
3.3.2 Luokittimen mittarointi 25
3.4 Regressio 29
3.5 Virhefunktion minimimointi algoritmeja 32
3.6 Yli- ja alisovittaminen 34
3.7 Ohjaamaton oppiminen 37
3.8 Vahvistusoppiminen 38
4 KONEOPPIMISPROJEKTIN PROSESSI 41
4.1 Datan kerääminen ja esikäsittely 42
4.1.1 Opetusdatan määrä 42
4.1.2 Datan virheiden käsittely 44
4.2 Muuttujien valinta 47
4.3 Algoritmin valinta 48
4.4 Mallin opetus 49
4.5 Mallin arviointi 50
5 NEUROVERKOT 51
5.1 Neuroverkon arkkitehtuuri 51
5.2 Perseptroni 53
5.3 Aktivointifunktiot 54
5.3.1 Sigmoid-funktio (logistinen funktio) 54
5.3.2 Hyperbolinen tangentti (tanh) 55
5.3.3 Softsign 56
5.3.4 ReLu 57
5.4 Single layer perceptron SLP 59

5.5 MLP (multilayer perceptron) 59
5.6 RNN (Recurrent neural network) 61
5.7 CNN (Convolutional neural network) 62
5.7.1 Konvoluutiokerros (convolution layer) 63
5.7.2 Yhdistämiskerros (pooling layer) 65
5.7.3 Täysin yhdistetty kerros (fully connected layer / Dense layer) 66
5.7.4 Opeteltavien parametrien määrä CNN-verkossa 66
6 KUVIEN JA OBJEKTIEN TUNNISTAMINEN 67
6.1 Kuva datalähteenä 67
6.2 Kuvien luokittelu 69
6.3 Objektintunnistus 70
6.4 Kuvien luokittelu ja objektin tunnistus Azuressa 73
7 CASE-TUTKIMUS 75
7.1 FXT ohjelmakirjasto, periaatteet ja parametrisointi 75
7.2 Toteutuksen yleiskuvaus 76
7.3 Opetusdata 77
7.4 Mallipohjan valitseminen koneoppimisen avulla 80
7.4.1 Kuvien luokittelu 81
7.4.2 Objektin tunnistus 88
7.5 Ryhmävalintojen tekeminen koneoppimisen avulla 91
7.6 Integrointi osaksi FX-editorin toimintoja 100
8 JOHTOPÄÄTÖKSET JA POHDINTA 102
8.1 Opetusdata ja menetelmät 102
8.2 Säätökaaviot 103
8.3 Ohjelmakirjaston käytön kehittäminen 104
LÄHTEET 106
LIITTEET 113
Liite 1. FX-editor / Template Manager ML concept 113

LYHENTEET JA TERMIT

ANN	Artificial neural network, keinotekoinen neuroverkko
API	Application programming interface, ohjelmointirajapinta
CNN	Convolutional neural network, konvoluutioneuroverkko
CPU	Central processing unit, keskusyksikkö
EMR	Exact match measure
FB	Function block
GBM	Gradient Boosting Machine
IaaS	Infrastructure as a Service
IoT	Internet of things, esineiden Internet
LSTM	Long Short-Term Memory
LTO	Lämmöntalteenotto
LVI	Lämmitys, vesi ja ilmastointi
ML	Machine learning, koneoppiminen
MLP	Multilayer perceptron
MSE	Mean squered error, keskimääräisen neliösumman virhe
OOM	Out of memory
PaaS	Platform as a Service
PI-kaavio	prosessi- ja instrumentointikaavio
PR-käyrä	Precision-Recall käyrä
Precision	Tarkkuus
PSO	Particle swarm optimization, parviälyoptimointi-algoritmi
RAU	Rakennusautomaatio, Rakennusautomaatiourakoitsija
RBF	Radial Basis funktiot
Recall	Saanti
RNN	Recurrent neural network, takaisinkytketty neuroverkko
SaaS	Software as a Service
SDCA	Stochastic Dual coordinate Ascent
SGD	Stochastic gradient descent
SLP	Single layer neural network
SSE	Sum of squared errors, neliösumman virhe
SVM	Support Vector Machine, Tukivektorikone

1 JOHDANTO

1.1 Rakennusautomaatiourakoinnin maailma

Urakoinnissa on yleensä seuraavat pelurit: tilaaja, suunnittelija, valvojat ja urakoitsijat. Urakoitsijat hoitavat toteutuksen suunnitelmien mukaisesti valvojien tarkastaessa työn tulokset. Tilaajan tehtäväksi jää hankkia projektiin tarvittavat asiantuntijat, jotka hoitavat oman erikoisalansa tekniset asiat sekä muut tilaajan velvollisuudet, jos tilaajan omasta organisaatiosta ei löydy osaamista näihin tehtäviin. Tilaajan ja muiden osapuolinen väliset velvollisuudet määritellään sopimuksissa. Rakennusalalla käytetään yleisesti YSE98-sopimusehtoja. Konsultointiin ja neuvontaan keskittyneet asiantuntijayritykset käyttävät myös konsulttitoiminnan yleisiä sopimusehtoja KSE 2013.

Suomessa toimii muutamia isoja suunnittelutoimistoja, jolta löytyy myös rakennusautomaatioon erikoistunut osasta. Tämän lisäksi Suomessa on useita pienempiä toimistoja, jotka tekevät RAU-suunnittelua. Suunnittelija ja valvoja saattavat olla projektissa samasta yrityksestä sekä sama henkilö. Urakoitsijan tehtäväksi jää toteuttaa suunnitelmien mukainen järjestelmä urakoitsijan käyttämän järjestelmätoimittajan laitteilla.

Lähtökohtaisesti kaikkien järjestelmätoimittajien laitteet täyttävät tilaajan tekniset vaatimukset pois lukien tilanteet, jossa tilaaja on päättänyt käyttää yhden toimittajan laitteita kaikissa uusissa projekteissa. Urakoitsijoiden valinnassa eniten painoarvoa saa yleensä hinta (halvin tarjous voittaa).

RAU-urakan kustannukset muodostuvat henkilökuluista ja laitekuluista. Laitekuluihin kuuluu järjestelmälaitteet ja muut kenttälaitteet. Lähtökohtaisesti näistä syntyvät kulut ovat jokaisella urakoitsijalla samalla tasolla. Henkilökulut koostuvat järjestelmäasiantuntijoista, asennustyöstä ja projektin muita tehtäviä hoitavista henkilöistä. Asennustyöt ja projektin muut yleiset tehtävät ovat järjestelmästä riippumattomia töitä. Asiantuntijoiden tekemän työn tehostaminen on usein helpoin keino parantaa.

Suomessa on useita eri suunnittelutoimistoja, joilla kaikilla on hieman erilainen tapa esittää sama toiminnallisuus. Tästä johtuen ohjelmointi on merkittävästi työaikaa vievä vaihe. Tämä korostuu, kun toteuttava järjestelmä tehdään suunnitelmien mukaisesti. Suunnitelmista käytetään yleisesti nimitystä säätökaaviot. Järjestelmätoimittajilla ja urakoitsijoilla on käytössä vakiomalleja yleisimmistä suunnitelmista, mutta lähes aina vakiomalleja joutuu muokkaamaan kohteeseen sopivaksi.

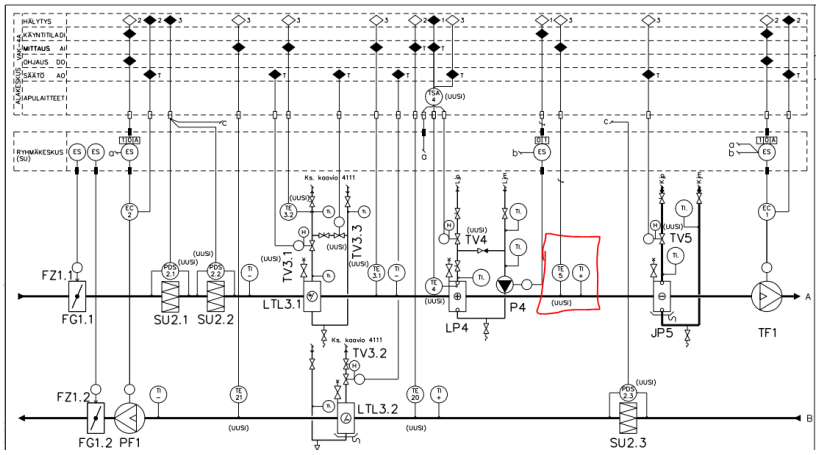
1.2 Ongelma, tavoite ja rajaukset

Tässä opinnäytetyössä tutkitaan RAU ohjelmointi- ja grafiikkatyön tehostamista koneoppimisen avulla. Projektit siirtyvät myyntiosastolta projektinhoitajalle, joka käytännössä hoitaa projektin toteutuksen luovutukseen asti. Projektiin liittyvät säätökaaviot tulee lähes aina ulkopuoliselta toimijalta, jonka jälkeen projektinhoitaja piirtää grafiikkakuvat järjestelmää varten. Grafiikkakuvat toimivat järjestelmän käyttöliittymänä. Tavoitteena on automatisoida grafiikkakuvien piirtäminen kuvantunnistussovelluksen avulla, jolloin projektinhoitajan työaikaa vapautuu muihin tehtäviin. Työnteon tehostamisen lisäksi automaattisesti generoitavat ohjelmat lisäävät toteutusten yhtenäistä ilmettä ja tätä kautta myös laatua. Generointiin käytettävien ohjelmien ollessa moneen kertaan testattuja voidaan välttää inhimilliset virheet ohjelmointityössä. Koneoppimissovelluksella olisi mahdollista tunnistaa säätökaavioista ohjelmointiin ja grafiikan tekoon vaikuttavat symbolit. Löydetyistä symboleista voitaisiin generoida automaattisesti niitä vastaavat ohjelmat ja grafiikat. Symboleiden tunnistus säätökaavioista toteutetaan kuvantunnistusalgoritmien avulla.

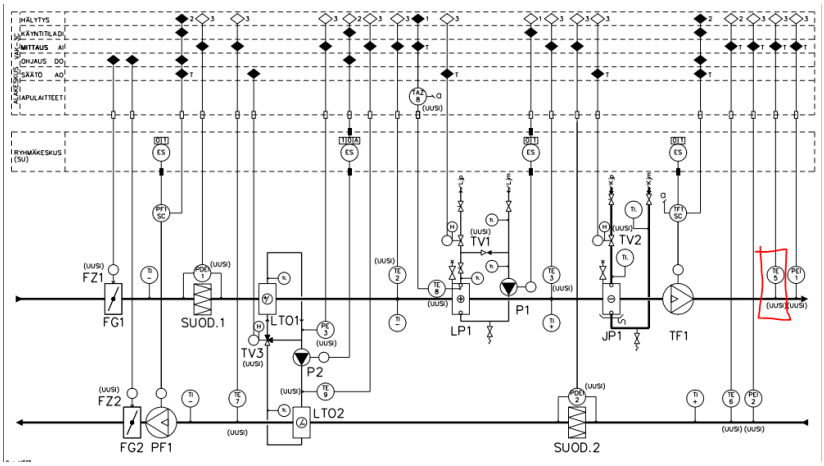
Säätökaavio on tekninen dokumentti, joka koostuu yleensä kahdesta osasta: PI-kaaviosta sekä toimintaselostuksesta. PI-kaaviossa esitetään järjestelmän LVI-prosessi sekä instrumentointi. Säätökaaviossa PI-kaavioon on lisätty nuottiviivasto. Nuottiviivasto on I/O-pisteiden esitystä varten. Säätökaaviolla tarkoitetaan tässä opinnäytetyössä dokumentin kaavio-osuutta. Toimintaselostus sisältää järjestelmän toiminnallisen kuvauksen sanallisessa muodossa ja saattaa sisältää viittauksia muihin dokumentteihin. Säätökaavioissa käytetään samoja symboleita (esim. lämpötila) useissa paikoissa. Symbolin merkitys on riippuvainen symbolin

sijainnista. Jokaisella laitteella on lisäksi yksilöllinen tunnus (positio), mutta suunnittelutoimistojen välillä on käytössä eri positiointikäytäntöjä. Myös saman toimiston suunnitelmien välillä voi olla eroavaisuuksia kohteiden välillä. Kuvioissa 1 ja 2 löytyy sama symboli / positio, mutta säätökaaviossa ne ovat eri kohdissa. Tällöin niillä on täysin eri merkitys toteuttavassa ohjelmassa.

Toimintaselostuksessa kerrotaan järjestelmän toiminta sanallisesti. Järjestelmien PI-kaavio voi olla identtinen, mutta toimintaselostukset voivat olla erilaiset. Eroavuus voi olla näennäinen, jolloin luetunymmärtämisen kautta sama ohjelma sopii molempiin järjestelmiin. Toisaalta selostuksessa voi olla myös merkittäviä tekijöitä, jolloin vakio-ohjelmaa joudutaan muokkaamaan selostuksen mukaiseksi. Toimintaselostuksen tulkitseminen on rajattu tämän opinnäytetyön ulkopuolelle.



KUVIO 1. Säätökaaviomalli, lämpötila-anturi TE5, lämmityspatterin jälkeinen lämpötila



KUVIO 2. Säätökaaviomalli, lämpötila-anturi TE5, tuloilman lämpötila

Opinnäytetyön pohjalta on tarkoitus muodostaa menetelmä, jolla säätökaavioista pystytään tunnistamaan ohjelmointiin ja grafiikkaan vaikuttavat symbolit. Löydetyistä symboleista muodostaan kutsukoodi, joka syötetään ohjelmakirjastoa käyttävään ohjelmaan. Ohjelma generoi kutsukoodin perusteella säätökaavion mukaiset ohjelmat ja grafiikat. Koneoppimisella tavoitellaan ohjelmointityön tehostamista kustannusten osalta sekä ohjelmointituotosten laadun yhtenäistämistä. Opinnäytetyön rajaus on koneoppimissovelluksen vaatimusten määrittely sekä yleisen ohjelmakehyksen suunnittelu. Käytännön kokeita kuvantunnistuksen kanssa tehdään rajatulla määrällä säätökaavioita.

1.3 Aikaisempi tekoälytutkimus rakennusautomaatioalalla

Aikaisempaa tutkimusta rakennusautomaation säätökaavioiden tulkitsemista tekoälyn avulla ei ole saatavilla. Tutkimus on keskittynyt PI-kaavioiden symboleiden tunnistamiseen ja varsinainen käyttötarkoitus on vanhojen kaavioiden digitalisointi. Tutkimuksissa on löydetty menetelmiä tunnistaa symbolit, mutta uniikkia semantiikka ei olla pystytty tunnistamaan. Esimerkiksi lämpötila-anturit tunnistetaan, mutta tunnistetuista antureista ei pystytä varmuudella sanomaan, mikä on raitisilma-anturi.

Nurminen ym. (2019) tutkivat vanhojen paperisten PI-kaavioiden digitalisoimista koneoppimisen keinoin. Tutkimuksessa luotiin objektin tunnistus malli, joka pystyi tunnistamaan vanhoista kuvista muun muassa venttiileitä ja pumppuja. Tutkimuksen opetusaineisto generoitiin käyttämällä suunnittelutoimiston symbolikirjastoa. Generoidut kuvat sisälsivät symboleita eri asennoissa ja kokoisina. Luokkia lukumäärä oli rajattu seitsemään. Objektin tunnistukseen käytettiin YOLO-algoritmia. Tutkimuksen tuloksena todetaan, että YOLO soveltuu PI-kaavioiden symboleiden tunnistamaan, joskin pienien symboleiden tunnistus oli haastavaa. (Nurminen ym. 2019.)

Kim ym. (2021) tutkivat valmiiksi kuvatiedostoina olevien PI-kaavioiden muuntamista suunnitteluohjelmiston käyttämään formaattinen. Opetusaineistossa oli yhteensä 74 eri luokkaa. PI-kaavioista tunnistettiin lisäksi tekstiä. Tutkimuksen tu-

loksena kuvista saatiin tunnistettua symbolit noin 97 % ja teksti 92 % tarkkuudella. Tutkimuksen tuloksena saatiin taulukko tunnistetuista symboleista. Symboleiden väliset yhteydet ja riippuvuudet eivät selviä taulukosta. (Kim ym. 2021.)

Kuvankäsittely tekniikoihin pohjautuvaa putkistojen tunnistusta kaaviosta on tutkittu useissa tutkimuksessa. Menetelmät pohjautuvat pääosin suodattimien tai rastereiden käyttöön. (Yu ym. 2019; Arroyo 2016; Kang, Lee & Baek 2019; Tan, Chen & Tan 2016.) Kyseiset menetelmät ovat käyttökelpoisia siinä tapauksessa, että grafiikkakuvat generoidaan täysin säätökaavion pohjalta ilman valmista mallipohjaa.

Tekoälytutkimus rakennusautomaatioalalla on keskittynyt rakennusten energian käytön optimointiin. Rakennusten energiatehokkuutta pystytään parantamaan älykkäillä ohjausalgoritmeilla. Toteutus jakaantuu pääsääntöisesti kahteen osaan; rakennuksessa toimivaan automaatiojärjestelmään ja pilvipalveluissa toimivaan koneoppimisympäristöön. Koneoppimisympäristön sovellus laskee rakennukselle optimaaliset verkostojen asetusarvot, jotka välitetään automaatiojärjestelmään toteutettavaksi. Pohjimmiltaan sovellukset optimoivat lämmitysenergian käyttöä. Laskentaan käytetään rakennuksesta saatavia lämpötilamittauksia sekä säätietoja. Käytössä olevat algoritmit pohjautuvat koneoppimisen tekniikoihin ja algoritmeihin, jota ovat mm. neuroverkot (*Neural Networks*), tukivektorikone (*Support Vector Machines*) sekä Radial Basis funktiot (RBF) (Moradzadeh ym. 2020; Wortmann ym. 2017).

Lämmitysenergian optimointi suoritetaan siten, että sisäilman olosuhteet pysyvät hyväksyttävinä. Käytännössä sisäilman olosuhteiden annetaan vaihdella tietyllä välillä esimerkiksi 21–22°C. Lämmitysenergian optimoinnissa tulee huomioida kaikki verkoston alueella olevat huoneet ja lämpötilat. Pienin mahdollinen lämmitysenergia löydetään käyttämällä PSO-algoritmia (*particle swarm optimization*) (Deng & Wang 2018). PSO-algoritmi on optimointitekniikka, joka kuuluu parviälyalgoritmien joukkoon. PSO soveltuu hyvin yhden muuttujan optimointiin. PSO-algoritmin lopetusehtona voidaan pitää, että kaikki lämpötilat ovat hyväksytyissä rajoissa. (Kusiak, Li & Tang 2010.) Optimoitava muuttuja lämmitysenergian käytössä on verkoston menoveden lämpötila. Neuroverkon ja PSO-

algoritmin käytöllä energiaa säästetään keskimäärin 7–10 % (Deng & Wang 2018; Kusiak, Li & Tang 2010).

Singaporessa tehdyssä tutkimuksessa optimoitiin jäähdytysenergian käyttöä huomioiden sisäilman lämpötilan mukavuus. Energian optimointi ja mukavuus ovat usein vastakkaisia tavoitteita. Esitetty ratkaisu pohjautuu ANN-neuroverkkojen hyödyntämiseen. Tutkimuksen mallirakennuksessa päästiin hyviin tuloksiin jäähdytysenergian ja jäähdytykseen käytetyn sähkön vähentämisessä sekä samalla sisäilman mukavuus parani. (Yang ym. 2020.)

Keskeinen haaste koneoppimismallien käyttöönotossa on rakennusautomaatiojärjestelmien pisteiden linkitys laskentamalleihin. Linkitys on pääosin tehty manuaalisena työnä, koska rakennusautomaatiojärjestelmät eivät yleensä sisällä yksiselitteistä pisteitä kuvailevaa metatiedostoa. Metatiedot on usein koodattu pisteiden tunnukseen tai pistetekstiin. Tätä ongelmaa on lähetty ratkaisemaan koneoppimisen keinoin. (Mishra ym. 2020; Wang ym. 2018.)

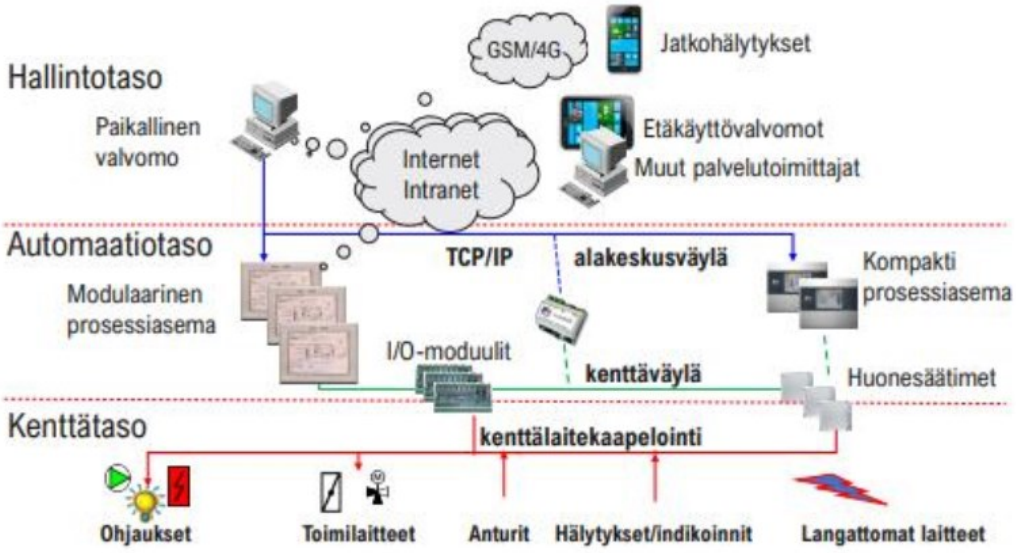
2 RAKENNUSAUTOMAATIO

2.1 Rakennusautomaatiojärjestelmän rakenne ja tehtävä

Kiinteistöihin on asennettu monenlaisia LVI- ja sähköteknisiä järjestelmiä. Näitä järjestelmiä ovat mm. ilmanvaihtokoneet, lämmitysverkosto ja valaistus. Järjestelmien tulee olla hallittavissa ja ohjattavissa. Tätä tehtävää kiinteistöissä hoitaa rakennusautomaatiojärjestelmä.

Rakennusautomaatiojärjestelmän keskeiset ominaisuudet ovat varmistaa käyttäjien mukavuus ja turvallisuus sekä samanaikaisesti minimoida kiinteistön energiankulutus. Mukavuus määritellään optimaalisilla sisäilman olosuhteilla, joita ovat lämpötila ja sisäilmanlaatu (CO^2 ja kosteus). Lisäksi työpisteiden ja yleisten tilojen riittävä valaistus lasketaan mukaan mukavuustekijöihin. (Hayduk, Kwasnowski & Mikòs 2016.)

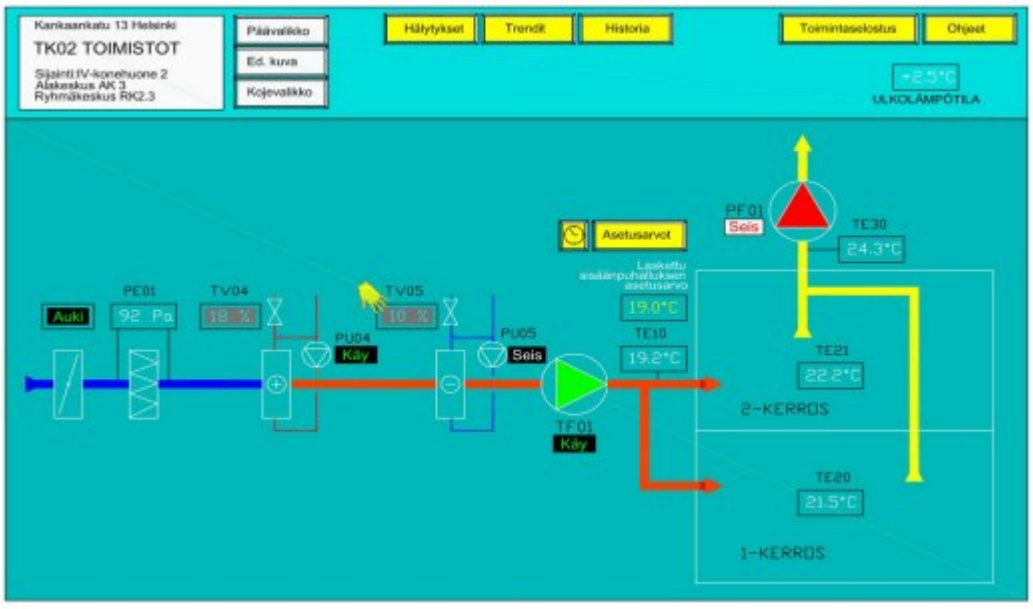
Kuviossa 3 on esitetty perinteisen rakennusautomaatiojärjestelmän rakenne. Rakennusautomaatiojärjestelmä voidaan jakaa hierarkkisesti kolmeen tasoon: hallinta-, automaatio- ja kenttätaso (Spangar & Sandström 2018, 59).



KUVIO 3. Rakennusautomaation hierarkia (Spangar & Sandström 2018, 60)

Hallintotasolla on käytännössä valvomotietokone, joka toimii keskitettynä käyttöliittymänä ja kerää historiadataa sekä tarvittaessa laatii raportteja. Automaatiotasolla tarkoitetaan alakeskuksia, joihin kenttätason laitteet on kytketty. Automaatiotaso hoitaa myös prosessiohjausta. Nykypäivänä on yleisesti käytössä myös ylätason valvomo-ohjelmistot, joilla voidaan hallita ja analysoida useita kiinteistöjä samanaikaisesti. Nämä ohjelmistot ovat pääsääntöisesti toteutettu pilvipalveluina.

Automaatiotasolla hallitsevana laitteena on alakeskus, joka sisältää I/O-moduulit ja keskusyksikön (CPU). Keskusyksikössä on järjestelmän ohjelmat ja se toimii käyttöliittymänä järjestelmään. Rakennusautomaatiojärjestelmän käyttöliittymää varten ohjattavista järjestelmistä laaditaan prosessikuvat (kuvio 4), joihin tehdään niin kutsutut pistekiinnitykset. Pistekiinnityksen tehdään grafiikkakuvan objekteihin, jolloin objekti saa arvonsa pistetietokannassa kiinnityksen mukaisesta pisteestä. Prosessikuvien katselu voi vaatia oman ohjelman tai katselu voi tapahtua web-selaimen kautta. Järjestelmän ohjelmointi tehdään yleensä graafisella vuokaavio- tyyppisellä esityksellä, suoraan ohjelmakoodina tai näiden yhdistelmällä. Tämä riippuu järjestelmätoimittajasta ja käytettävistä ohjelmista.



KUVIO 4. Rakennusautomaatiojärjestelmän prosessikuva (Sahala 2018, 129)

Kenttätasolla sijaitsevat varsinaiset laitteet, joita automaatiotasolta ohjataan. Klassisessa automaatiojärjestelmässä kenttälaitteet eivät itsessään sisällä suu-

rempaa älykkyyttä. Tämän päivän järjestelmissä kenttä- ja automaatiotasojen välinen raja on alkanut hämärtyä. Markkinoille on tullut IoT-antureita (Internet of things), jotka pystyvät kommunikoimaan suoraan hallintotason valvomon kanssa. Toinen esimerkki on maalämpöpumppu, joka sisältää prosessisäätimen ja joukon kenttälaitteita. Automaatiojärjestelmän näkökulmasta tämä on kuitenkin keskusyksikön alapuolella oleva laite (järjestelmä järjestelmän sisällä). Perinteiset kenttälaitteet kytketään I/O-moduuleihin, jotka voidaan jakaa neljään eri tyyppiin DI, DO, AI ja AO.

- DI – digitaalinen sisääntulo, digital input (0 tai 1).
- DO – digitaalinen ulostulo, digital output. Käytännössä rele, jolla voi olla vaihtokärki (NO,NC).
- AI – analoginen sisääntulo, analog input. Analogisten viestien sisääntulomoduuli. Sisään tuleva viesti voi olla resistiivinen- (esim. NTC10, PT1000, Ni1000), virta- (0/4-20mA) tai jännitemittaus (0/2-10V). Mittaus muunnetaan A/D-muuntimella digitaaliseen muotoon.
- AO – analoginen ulostulo, analog output. Analoginen ohjausviesti, joka on jänniteviesti 0/2-10V tai virtaviesti 0/4-20mA.

Kuvassa 3 on esitetty rakennusautomaatiojärjestelmän rakenne, jossa alakeskukset ovat yhteydessä toisiinsa alakeskusväylän kautta. Yleisin alakeskusväylän protokolla on Ethernet TCP/IP. Kenttäväylissä on enemmän kirjoja järjestelmien välillä. Tyypillisiä kenttäväyliä ovat mm. Modbus, BACnet, Profibus, Dali ja KNX. Yleisimmät rakennusautomaatiojärjestelmät Suomessa ovat Trend, DEOS-AG, Fidelix, Siemens Desico ja Schneider Electric EcoStruxure.

2.2 Fidelix

Fidelix on vuonna 2002 perustettu suomalainen rakennusautomaatiojärjestelmää kehittävä ja urakoiva yritys. Fidelix-järjestelmällä urakoi Fidelixin lisäksi laajayhteistyökumppaniverkosto. Fidelixin aluekonttorit ja yhteistyökumppanit kattavat koko Suomen. Ruotsin markkinoilla Fidelix on ollut mukana perustamisesta asti. Ruotsissa toimii myös osa tuotekehityksen henkilöistä, sekä oma myynti- ja tukiorganisaatio, jotka auttavat paikallisia yhteistyökumppaneita. Suomen ja Ruotsin ulkopuolella toimintaa on verrattain vähän, mutta se on kasvamaan päin. Fidelix

on Suomessa markkinajohtaja ja tavoittelee pohjoismaiden markkinajohtajuutta lähivuosien aikana (Fidelix n.d.).

Joulukuussa 2020 pohjoismaalainen talotekniikkayritys Assemblin osti koko Fidelix Oy:n osakekannan. Yrityskauppa vaati kilpailu- ja kuluttajaviraston hyväksynnän. Yrityskauppa hyväksyttiin ehdollisena elokuussa 2021 ja saatiin vietyä loppuun asti syyskuussa 2021. Fidelix jatkaa toimintaa omalla nimellään itsenäisenä liiketoimintayksikkönä.

2.3 Fidelix-järjestelmän grafiikka ja ohjelmat

Fidelix-järjestelmän prosessiohjelmointi tehdään IEC61131-3 standardin mukaisella ohjelmointikielellä. Puhekielessä ohjelmista, joka on tehty IEC61131-3 standardin mukaisella kielellä, käytetään nimitystä ”IEC-ohjelma” tai ”PLC-ohjelma”. Kielen syntaksi muistuttaa C-kieltä. Perusominaisuuksiltaan IEC-kieli on varsin rajattu ja suppea verrattuna ylemmin tason ohjelmointikieliin. Fidelix-järjestelmän ohjelmointia varten täytyy IEC:tä laajentaa ottamalla käyttöön järjestelmäfunktiot. Järjestelmäkirjasto sisältää tarvittavat funktiot kommunikointiin pistetietokannan kanssa. Tyypillisiä järjestelmäfunktioita ovat mm. SetDigitalPointF ja GetDigitalPointF, jolla pystytään lukemaan ja kirjoittamaan pistetietokantaan kokonaislukumuuttujan arvo. Reaalilukuja varten on omat Set/GetAnalogPointF-funktiot. Kirjoittaminen ja lukeminen pistetietokannasta tapahtuu pistetunnuksen kautta. Järjestelmäfunktiot on kuvattu Fidelix-ohjelmointidokumentaatioissa.

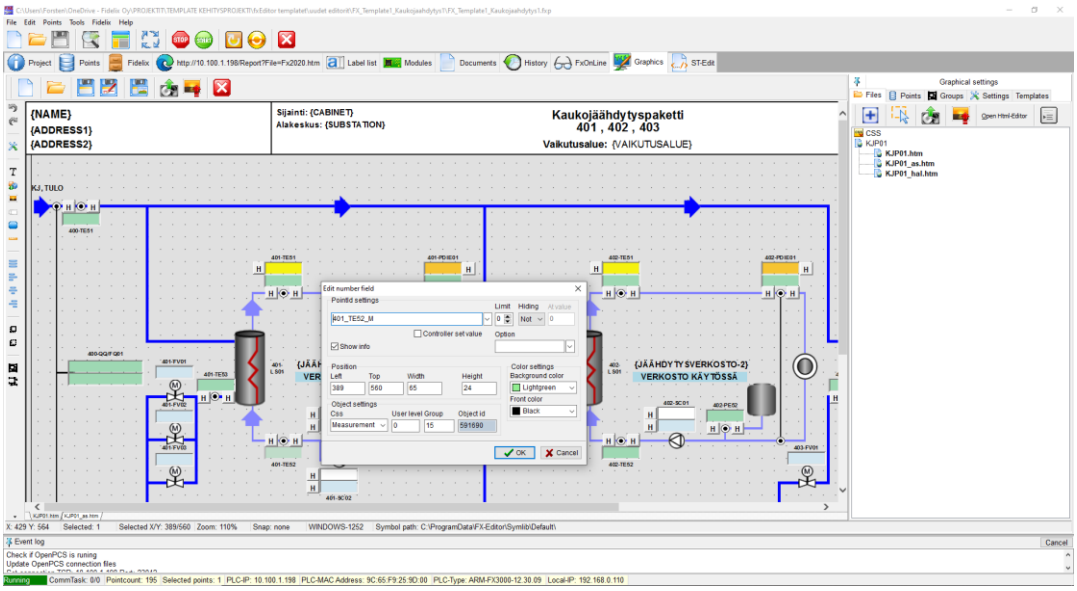
IEC-ohjelmointia voidaan standardin mukaisesti tehdä viidellä eri kielellä (taulukko 1). Ohjelmointikielistä kaksi on tekstipohjaisia ja loput kolme ovat graafisia ohjelmointikieliä. (Devasia, 2021; Heinz & Tiegelkamp, 2010, 23–25.)

TAULUKKO 1. IEC 61131-standardin mukaiset ohjelmointikielet (Devasia, 2021)

Ohjelmoinkieli	Lyhenne	
Ladder Diagram	LD	Graafinen
Function Block Diagram	FBD	Graafinen
Structured Text	ST	Teksti
Instruction List	IL	Teksti
Sequential Function Chart	SFC	Graafinen

Fidelix-järjestelmän ohjelmointi onnistuu kaikilla taulukon x tavoilla, mutta pääosa järjestelmää ohjelmoivista henkilöistä käyttää struktuurista tekstiä (ST). ST tarjoaa suurimman vapauden ohjelmien kustomointiin. Fidelix-järjestelmätoimittaja tarjoaa valmisohjelmansa ST-muodossa. Valmisohjelmat paketoidaan usein funktioblokkien (FB) sisään, jolloin pääohjelma koostuu useista FB-kutsuista.

Fidelix-järjestelmän käyttöliittymää käytetään verkkoselaimen kautta. Prosessikuvat luodaan grafiikkaeditorilla, joka muodostaan kuvasta HTML-sivun. Pohjimmiltaan HTML-sivut ovat tekstimuotoisia ja niitä pystyy muokkaamaan tekstieditorin kautta. Käytännössä tämä ei ole tarpeen tai edes suotavaa. HTML-sivujen muokkaus tapahtuu grafiikkaeditorissa (kuva 1).



KUVA 1. Kuvakaappaus FX-editorin grafiikkaeditorista

Grafiikkakuva koostuu objekteista, jotka voivat olla tyypiltään esimerkiksi teksti, kuva tai symboli. Objekteille voidaan määritellä, että ne saavat arvonsa pistetietokannasta. Pistetunnuksen eri arvoilla voidaan näyttää symbolista eri versio tai tekstikentän sisältö voi muuttua. Pistetietokantaan tehtävien arvojen nouto tapahtumaa käyttöjärjestelmän toimesta automaattisesti pistetunnuksen määrittelyn kautta. Pistetunnus on määritelty objektin FdxPntID-parametriin (kuva 2). Objektille voidaan määritellä myös ryhmätunnus, jota mallipohjien käyttöohjelma (Template manager) käyttää hyväksi.

```
<INPUT id="ID591690" class="Measurement" style="POSITION: absolute; LEFT: 1229px; TOP: 290px; WIDTH: 65px;
HEIGHT: 24px; COLOR: #000000; BACKGROUND-COLOR: #90EE90; " FdxUserLevel="0" FdxEditGroup="15" FdxGroupCode=
"401-TE52" FdxPntId="401_TE52_M" FdxManualEnabled="1" FdxShowInfo="1" FdxType="FdxAnalogPoint"
FdxHideObject="0" FdxHideValue="0" FdxFrontColor="#000000" FdxBgColor="#90EE90" FdxShowSetValue="0">
```

KUVA 2. Kuvakaappaus grafiikkakuvan HTML-määrittelyistä

Pistetietokanta on kokojärjestelmän perusta, jonka päälle IEC-ohjelma ja grafiikkakuvat tulevat. Pistetietokantaan pystyy tekemään pieniä mikro-ohjelmia, jotka eivät vaadi IEC-ohjelmointia. Tällöin puhutaan pisteohjelmoinnista. Pisteohjelmoinnin avulla pystytään tekemään muun muassa PID-säätimiä sekä raja- tai ristiriitahälytyksiä. Nämä pystytään tekemään myös IEC-ohjelmoinnin kautta. Prosessin ohjelmoitsijan tulee tehdä päätös käyttääkö hän pisteohjelmointia hyväksi ja missä laajuudessa.

3 TEKOÄLY JA KONEOPPIMINEN

3.1 Johdotus tekoälyyn

Tekoälyn tutkiminen aloitettiin toisen maailman sodan jälkeen. Tekoäly (artificial intelligence) sanana vakiintui käyttöön vuonna 1956. Tekoälyn tutkiminen pitää sisällään useita eri tutkimusaloja. Tekoälytutkimukseen vaikuttaneisiin tutkimusaloihin lukeutuvat filosofia, matematiikka, taloustieteet, neurotiede, psykologia, tietotekniikka, kielitieteet, säätötekniikka sekä kybertekniikka. (Russel & Norvig 2016, 1–2). Tekoälyllä ei ole yksiselitteistä määritelmää. Yhtenä määritelmä voidaan pitää, että tekoälyllä opetetaan koneita tekemään ihmisten tehtäviä. (Trask 2018.) Ihmisten tekimien tehtävien suorittaminen edellyttää ihmismäistä käytöstä. Jotta tekoäly pystyisi ihmismäiseen käytökseen, sen pitäisi ajatella ja toimia kuten ihminen, sekä olla kykenevä rationaaliseen ajatteluun ja toimintaan. (Russel & Norvig 2016, 1–2.)

Vuonna 1950 Alan Turing esitti testausmenetelmän, jolla voitaisi vastata kysymykseen *voivatko koneet ajatella*? Testin tarkoitus on mitata tekoälyn ihmismäisyyttä. Testissä ihminen haastattelee kahta henkilöä, joista toinen on ihminen ja toinen tietokone. Tekoäly läpäisee testin, jos haastattelija ei pysty kertomaan kumpi haastateltavista on tietokone. Testiä kutsutaan kehittäjänsä mukaan Turingin testiksi ja siitä on muodostunut tekoälytutkimuksen klassikko. (Russel & Norvig 2016, 1–2.)

Tekoälyn tutkimuksessa ja käyttöönotossa on ollut paljon odotusarvoja. Historian aikana odotuksia ei ole läheskään aina pystytty täyttämään, jolloin tutkimus tai käytännön sovelluksen kehitys on lopetettu. Tietotekniikan kehittyminen mahdollistaa suurien datajoukkojen käyttämisen tekoälytutkimuksessa, jolloin datan kautta tapahtuva tutkimus on saanut uutta nostetta. Tällä hetkellä tekoälyä käytetään kaikilla talouden aloilla. (Russel & Norvig 2016, 16–30.)

3.2 Tekoälystä koneoppimiseen

Tekoäly voidaan jakaa kahteen eri ryhmään: kapeaa ja vahvaan tekoälyyn. Nykyiset sovellukset ovat kapean tekoälyn sovelluksia. (Trask 2018.) Kapean tekoälyn sovelluksia ovat muun muassa hakukoneet, roskapostisuodattimet sekä kasvojen- ja puheentunnistusohjelmat. (Tuominen & Neittaanmäki 2018, 8.)

Kapea tekoälysovellus pystyy ratkaisemaan ongelman mihin se on kehitetty, kun taas vahvalla tekoälyllä pystyttäisiin selviämään kaikista yleistoiminnoista mistä ihminenkin. Täten vahva tekoäly muistuttaisi paljon ihmisen älyä. Systeemillä olisi kognitiivisia taitoja ja yleinen kokemuksellinen ymmärrys sen toimintaympäristöstä. (Trask 2018.)

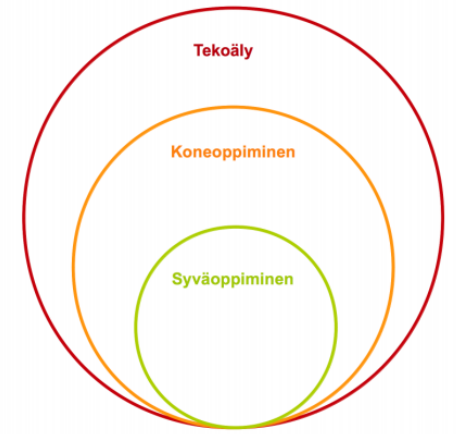
Toinen määritelmä vahvalle tekoälylle voisi olla koneen oppiminen ihmistä älykkäämmäksi ja toimiminen täysin irrallaan ihmisälystä. Jotta tekoäly pystyisi ihmismäiseen oppimiseen, sen tulisi pystyä oppimaan asioita kuvista, puheesta, teksteistä tai muista tietolähteistä, joita sille ei kuitenkaan ole alun perin opetettu. Tämän hetken tekoälyalgoritmit eivät pysty oppimaan sellaista tietoa, jota sen opettamiseen käytetyssä datassa ei ole. (Tuominen & Neittaanmäki 2018, 2–3.)

Markkinoinnissa nykypäivän tekoäly saa turhankin isot mittasuhteet. Tekoälyä ylistetään uutena kultakaivoksena. Todellisuudessa nykypäivän tekoälysovellukset ovat datan analysointia, josta pyritään löytämään datapisteiden välisiä yhteyksiä käyttämällä hyväksi matemaattisia menetelmiä ja ennustamaan tulevaa tai luokitella asia kuuluvaksi tiettyyn ennalta määriteltyyn ryhmään. (Esposito & Esposito 2020, luku 2.)

Nykypäivän käytännön tekoäly on koneoppimista (*machine learning* ML). Koneoppimisen prosessi voidaan jakaa kahteen osaan. Aluksi kone opetetaan tunnistamaan datapisteiden välisiä yhteyksiä. Toisessa vaiheessa koneelle syötetään entuudestaan tuntematon datasarja, joka generoi vastauksen opetetun mallin mukaisesti. Jos mallin käytöstä saatavat vastaukset eivät ole riittävän hyviä, täytyy opetusdataa muokata ja kouluttaa kone uudestaan. (Esposito & Esposito 2020, luku 2; Tuominen & Neittaanmäki 2018, 6.)

Koneoppiminen pohjautuu vahvasti tilastotieteisiin. Tilastotieteen tarkoitus on löytää teoreettinen malli, joka selittää käytetyn datan. Koneoppimisen tarkoitus on

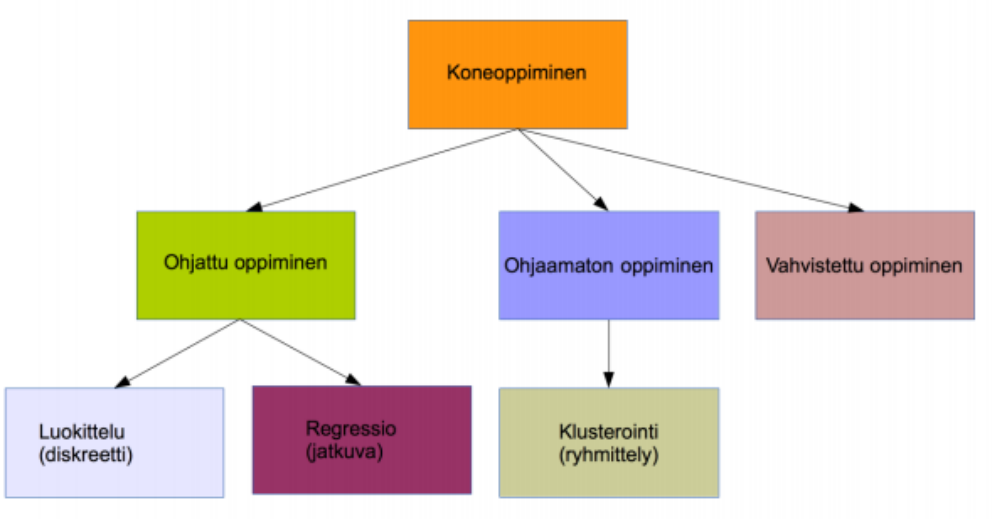
löytää käytännön malli, jolla pystytään tekemään riittävän hyviä ennustuksia ennestään tuntemattomasta datasta. Tämä on hyvin yksinkertaistettu eroavuus tilastotieteen ja koneoppimisen välillä. Tilastotiede on paljon muutakin kuin teoreettisen mallin etsimistä. Käytännön mallin rakentaminen tehdään opetusdatan ja testidatan avulla. Opetusdatalla malli opetetaan tunnistamaan muuttujien väliset yhteydet ja testidatalla mallin toiminta varmistetaan. (Esposito & Esposito 2020, luku 10; Witten ym. 2017, luku 1.5.)



KUVIO 5. Koneoppiminen on tekoälyn osa-alue ja syväoppiminen on yksi koneoppimismenetelmä (Tuominen & Neittaanmäki 2018, 8)

Koneoppiminen voidaan jakaa kolmeen eri kategoriaan oppimistyylin perusteella: ohjattu oppiminen, ohjaamaton oppiminen ja vahvistettu oppiminen (kuvio 6). Ongelma, johon koneoppimista halutaan käyttää, ja käytössä oleva data määrittelevät mitä opetustyyliä kulloinkin on hyvä käyttää.

Ohjatussa oppimisessa alkuperäinen opetusdata sisältää oikeat vastaukset ongelmaan. Ohjatun oppimisen voidaan katsoa olevan oppimista esimerkistä. Ohjaamattomassa oppimisessa oikeita vastauksia ei tiedetä, jolloin oppiminen on havaintojen kautta oppimista. (Esposito & Esposito 2020, luku 3.) Yllättävän suuri osa ongelmista on luonteeltaan luokitteluun tai ennustamiseen liittyviä (Marsland 2014, luku 1.4.1). Koneopettamista voidaan suorittaa neuroverkkojen avulla. Neuroverkot ovat usein monikerroksisia. Tällöin oppimista voidaan kutsua myös syväoppimiseksi. (Tuominen & Neittaanmäki 2018, 8.)



KUVIO 6. Koneoppimisen oppimistyylit ja alalajit (Tuominen & Neittaanmäki 2018, 12)

3.3 Ohjattu oppiminen

Ohjatussa oppimisessa (*supervised learning*) konetta opetetaan opetusdatan avulla tekemään valintoja samankaltaisesta datasta. Opetusdatassa muodostetaan syöte-tavoite-parit. Opetusdata voidaan esittää ($\mathbf{x}_i$, $\mathbf{y}_i$) muodossa, jossa $\mathbf{x}_i$ ovat syöte (input) ja $\mathbf{y}_i$ ovat tavoite (*targets*). Kirjaimet x ja y ovat kirjoitettu lihavoituna, koska ne voivat sisältää useita arvoja (muuttujia). Indeksi i kuvastaa vastaavasti sitä, että syöte-tavoite-pareja on 1 – N kappaletta. (Marsland 2014, luku 1.4; Tuominen & Neittaanmäki 2018, 13.)

Ohjatussa oppimisessa opetusdata voidaan esittää kaavan 1 muodossa (Murphy 2012, 2; Jung 2021, 32).

$$X = \{(x_i, y_i)\}_{i=1}^{n}, \tag{1}$$

jossa X on opetusdata ja n on syöte-tavoite-parien lukumäärä. Yksinkertaisimmillaan syötedata x_i voi olla esimerkiksi henkilön pituus ja paino, mutta syötedata x_i voi olla myös monimutkaisesti jäsennelty objekti, esim. kuvatiedosto tai sähköpostiviesti. (Murphy 2012, 2.)

Tavoitedata voi olla lähes mitä tahansa, mutta yleisimmät käytetyt tekniikat olettavat y_i:n olevan kategorioitavissa, jolloin se voi saada nimellisen arvon rajatusta

joukosta, $y_i \in \{1, \dots, C\}$ (esim. ihminen, eläin tai talo) tai y_i voi saada reaaliluku-arvon. (Murphy 2012, 2.)

Lähes kaikkia koneoppimisongelmia voidaan lähestyä kirjoittamalla se funktion muotoon

$$y = f(x), \tag{2}$$

jossa *f* on tuntematon funktio. Opetusdatan avulla pyritään löytymään funktio, joka toteuttaa syöte-tavoite-parit riittävän hyvin. Täyttä varmuutta siitä, miten syöte-tavoite-parit toteutuvat, ei välttämättä voida saavuttaa, joten kirjoitetaan kaava 3.9 ennustuksia tehtäessä seuraavaan muotoon

$$\hat{y} = \hat{f}(x), \tag{3}$$

jossa hattu-merkillä tarkoitetaan arviota. (Murphy 2012, 3–4.)

Ohjattu oppiminen voidaan jakaa kahteen alaryhmään: luokittelu ja regressio. Jako tehdään tavoitedatan perusteella. Luokittelussa tavoitedata on diskreettiä, eli tavoitteet voidaan jakaa erillisiin ryhmiin. Regressiossa tavoitedata on jatkuvaa. (Tuominen & Neittaanmäki 2018, 13; Murphy 2012, 2.)

3.3.1 Luokittelu

Luokittelussa pyritään kartoittamaan sisään tulevista syötteistä *x* ulostulo *y*, kun $y \in \{1, \dots, C\}$,jossa *C* on luokkien lukumäärä. Jos luokkien lukumäärä on kaksi, sitä kutsutaan binääriseksi luokitteluksi; $y \in \{0,1\}$. Tällöin oikea vastaus voi yksinkertaisimmillaan olla ”kyllä” tai ”ei". (Elements of AI, n.d.; Murphy 2012, 3)

Kun luokkien lukumäärä on yli kaksi, kyseessä on moniluokkaluokittelu. Useissa tapauksissa luokat eivät ole toisiaan pois sulkevia, esimerkiksi ihminen voi kuulua luokkiin pitkä ja vahva. Tällaisissa tapauksissa ongelmaa kannattaa lähestyä usean toisistaan riippuvan binäärisen luokittelun kautta. Kun puhutaan luokittelusta, sillä tarkoitetaan yleisesti luokittelua, jossa on yksi ulostulo y. (Murphy 2012, 3; Marsland 2014, luku 1.4.2.)

Binääriluokittelu on moniluokkaluokittelun erikoismuoto. Perusperiaatteiltaan se on samanlainen, mutta käytännön algoritmien kannalta luokkien lukumäärällä on todella suuri merkitys. Käytännössä tämä johtaa siihen, että molemmille luokittelutavoille on omat algoritmiperheet. Yleisiä luokitteluun käytetty algoritmiperheitä ovat lähimmän naapurin luokitin ja tukivektorialgoritmit.
(Esposito & Esposito 2020, luku 8.)

Tyypillisiä luokittelu algoritmeja ovat mm.

- Päätöspuu (*Decision Tree*) soveltuu käytettäväksi binääriluokittimen kanssa. Päätöspuu alkaa juuresta ja jokaisessa haarassa on sääntö, jonka perusteella jatketaan suuntaan tai toiseen. Lopulta päästään lehteen asti, joka on luokittimen tulos. Haarojen ja sääntöjen määrittely tapahtuu opetuksen aikana. (Esposito & Esposito 2020, luku 3.)

- Satunnaismetsä (*Random Forest*) on päätöspuualgoritmista muokattu versio. Kun perinteisessä päätöspuussa on yksi puu, satunnaismetsäalgoritmeissa on useita yksinkertaisia puita, jotka opetetaan erikseen. Lopputulos saadaan yksittäisten puiden keskiarvona. (Esposito & Esposito 2020, luku 3.)

- Tukivektorikone (*Support Vector Machine*) syöte esitetään n-ulotteisessa tasossa. Algoritmi etsii riittävän isoa väliä näiden pisteiden välillä. Kaksiulotteissa tasossa algoritmi etsii käyrää, jolla taso voidaan jakaa kahtia. Kolmeulotteissa avaruudessa algoritmi hakee tasoa, joka jakaa avaruuden kahtia. (Esposito & Esposito 2020, luku 3.)

- Naïve Bayes laskee syötteestä ehdollisia todennäköisyyksiä, jolla syöte sisältyy ennalta määriteltyihin luokkiin. Algoritmi pohjautuu Bayesin teoreemaan, joka on todennäköisyysmatematiikan klassikkoja. (Esposito & Esposito 2020, luku 3.)

- Logistinen regressio (*Logistic Regression*) on nimestään huolimatta luokittelualgoritmi. Algoritmi laskee todennäköisyyden, jolla syöte on tosi. Laskennassa käytetään sigmoid-funktiota, jonka tulos on nollan ja yhden vä-

lillä. Sigmoid-funktion matemaattisesti luonteesta johtuen, logistinen regressio soveltuu parhaiten käytettäväksi binääriluokittelussa. (Esposito & Esposito 2020, luku 3.) Logistista regressiota voidaan kuitenkin käyttää myös useampiluokkaisen ongelman ratkaisuun. Tällöin käytettävä malli monimutkaistuu ja puhtaaseen luokitteluun suunnitellut algoritmit, esimerkiksi SVM-algoritmit, ovat tehokkaampia. (Hosmer, Lemeshow & Sturdivant 2013, luku 2).

Tyypillisiä sovelluksia luokittelussa ovat muun muassa (Esposito & Esposito 2020, luku 3; Bonaccorso 2018)

- Roskapostisuodatin
- Tietojen ja mielipiteiden analysointi
- Aikaisen vaiheen lääketieteelliset diagnoosit kuvista
- Kuvien ja kuvioiden tunnistus (esimerkiksi kasvontunnistus)
- Luonnollisen kielen käsittely (*Natural language process NLP*)
- Uutisten lajittelu
- Petoksien ja virheiden tunnistus

3.3.2 Luokittimen mittarointi

Tarkkuus (accuracy)

Luokittelumallin toimintaa voidaan mitata useilla eri tavoilla. Tyypillisin mittari on tarkkuus (accuracy). Tarkkuuden laskenta tapahtuu yksinkertaisesti jakamalla oikein luokitellut syötteet syötteiden kokonaismäärällä (kaava 4). (Bonaccorso 2018, luku10; Aghdam & Heravi 2017, luku 3.5)

$$accuracy = \frac{1}{N}\sum_{i=1}^{N} 1[y_i == \hat{y}_i], \quad (4)$$

missä *N* on syötteiden lukumäärä. Kaavassa 4 jokainen oikea syöte on saman arvoinen. Hieman epätasapainossa olevien datajoukkojen kanssa voi olla perusteltua käyttää painotettua kerrointa. Dominoivia luokkia sisältävä datajoukko voi saada hyvin tarkkuuden tunnistamalla pelkästään dominoivat luokat. Pienellä

osuudella olevia luokkia ei välttämättä tunnisteta ollenkaan. Jos pienellä osuudella olevat luokat ovat oleellinen osa ongelmaa, on syytä käyttää painotettua tarkkuudenlaskentaa, (Aghdam & Heravi 2017, luku 3.5.)

$$accuracy = \frac{1}{N}\sum_{i=1}^{N} w_i * 1[y_i == \hat{y}_i], \quad (5)$$

missä w_i on luokan *i* painokerroin ja lasketaan seuraavalla kaavalla

$$w_i = \frac{1}{C*y_c^{(i)}}, \quad (6)$$

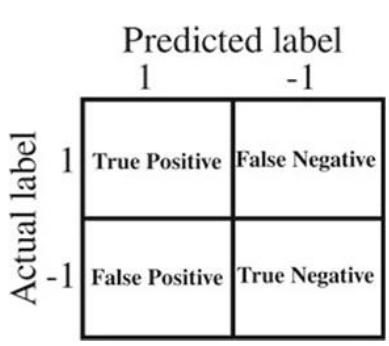
jossa C on kaikkien luokkien lukumäärä ja $y_c^{(i)}$ on luokkaan *i* kuuluvien näytteiden lukumäärä.

Sekaannusmatriisi (Confusion matrix)

Tarkkuus on hyvä mittari luokittimen toiminnasta, mutta sekaannusmatriisi kertoo siitä, kuinka hyvin luokitin toimii käytännössä (Bonaccorso 2018, luku 10). Luokittimen, jolla on lukumäärällisesti C luokkaa, sekaannusmatriisi M on $C * C$ kokoinen taulukko. Taulukon rivien ja sarakkeiden otsikot ovat luokkien nimet, arvot tai indeksit. Oikein luokitellut syötteet sijoitetaan taulukon soluun M_{ii}, missä *i* on todellisen luokan indeksi. Väärin luokitellut syötteet sijoitetaan soluihin M_{ij}, jossa *j* on luokittimesta saatu tulos (indeksi). (Aghdam & Heravi 2017, luku 3.5.)

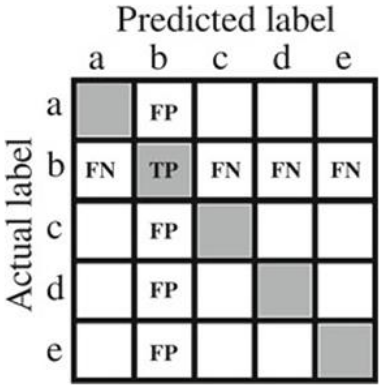
Binääriluokittimen sekaannusmatriisi on kuvion 7 mukainen. Kuviossa 7 TP (True positive) tarkoittaa oikein luokiteltua positiivista havaintoa. FP (False positive) positiiviseksi luokiteltu havainto, joka on kuitenkin väärin. FP on niin kutsuttu tyypin 1 virhe (väärä hälytys), joka voidaan havaita tarkemman tarkastelun kautta. TN (True negative) oikein luokiteltu negatiivinen havainto. FN (False negative) väärin luokiteltu negatiivinen havainto. FN on tyypin 2 virhe, jota ei ole helppo havaita, koska sitä ei todennäköisin syin oteta tarkempaan tarkasteluun. Tarkkuuden kaava 5 voidaan kirjoittaa myös kaavan 7 muotoon. (Bonaccorso 2018, luku 10; Aghdam & Heravi 2017, luku 3.5.)

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$



KUVIO 7. Binääriluokittimen sekaannusmatriisi on 2 * 2 taulukko (Aghdam & Heravi 2017, luku 3.5)

Useamman kuin kahden luokan luokittimen sekaannusmatriisi muodostaa taulukon, jonka diagonaalille tulevat TP arvot ja muihin soluihin FN arvoja (kuvio 8). Sekaannusmatriisi soveltuu käytettäväksi pienelle määrällä luokkia. Suurien sekaannusmatriisien analysointi on työlästä ja epäkäytännöllistä. Tästä syystä sekaannusmatriisista lasketaan usein muutama määrällinen arvo, jotka ovat luotettavia ja informatiivisia verrattuna tarkkuuteen. (Aghdam & Heravi 2017, luku 3.5.)



KUVIO 8. Moniluokkaisen luokittimen sekaannusmatriisi (Aghdam & Heravi 2017, luku 3.5)

Tarkkuus (precision) ja saanti (recall)

Tarkkuus ja saanti ovat määrällisiä arvoja, joilla voidaan arvioida mallin toimintaa. Tarkkuus on positiivisten havaintojen suhde kaikkiin positiivisiin havaintoihin (kaava 8). Saanti on positiivisten havaintojen suhde todellisiin positiivisiin havaintoihin (TP+FN) (kaava 9). (Bonaccorso 2018, luku 10; Aghdam & Heravi 2017, luku 3.5.)

$$precision = \frac{TP}{TP+FP} \quad (8)$$

$$recall = \frac{TP}{TP+FN} \quad (9)$$

Malli toimii täydellisesti, jos FP ja FN ovat nollia. Tällöin tarkkuus ja saanti saavat arvon yksi. Malli toimii epätarkasti, jos arvot ovat lähellä nollaa. Luokkien lukumäärän ollessa *C*, tarkkuus ja saanti lasketaan jokaisen luokan *i* yksilöllisten arvojen summana huomioiden painokertoimen w_i (kaavat 10–13). (Aghdam & Heravi 2017, luku 3.5.)

$$precision_i = \frac{M_{ii}}{M_{ii}+\sum_{j\neq i} M_{ji}} = \frac{M_{ii}}{\sum_j M_{ji}} \quad (10)$$

$$recall_i = \frac{M_{ii}}{M_{ii}+\sum_{j\neq i} M_{ij}} = \frac{M_{ii}}{\sum_j M_{ji}} \quad (11)$$

$$precision = \sum_{i=1}^{C} w_i * precision_i \quad (12)$$

$$recall = \sum_{i=1}^{C} w_i * recall_i \quad (13)$$

F-arvo

Kun halutaan arvioida mitkä luokittimista ovat selviytyneet ongelman ratkaisusta parhaiten, on kätevää yhdistää tarkkuus ja saanti yhdeksi arvoksi (Aghdam & Heravi 2017, luku 3.5.). Tätä arvoa kutsutaan F-arvoksi, josta käytetään usein F_1-variaatiota. F_1-arvossa tarkkuus ja saanti ovat saman arvoisia. F_1-arvon avulla pystytään havaitsemaan, jos tarkkuus tai saanti ovat epätasapainossa toisiinsa nähden (Bonaccorso 2018, luku 8).

F_1-arvo lasketaan tarkkuuden ja saannin harmonisena keskiarvona ja se voi saada arvon nollan ja yhden väliltä (kaava 14). Tuloksen ollessa yksi, malli toimii täydellisesti. (Aghdam & Heravi 2017, luku 3.5.)

$$F_1 = \frac{2}{\frac{1}{precision}+\frac{1}{recall}} = \frac{2TP}{2TP+FP+FN} \quad (14)$$

F_1-arvoa käytetään varsinkin alkuvaiheen luokittimien väliseen arviointiin. Riittävän hyvän F_1-arvon jälkeen, voidaan tarkastella mallia tarkemmin tarkkuuden, saannin ja sekaannusmatriisin kautta. (Aghdam & Heravi 2017, luku 3.5.)

3.4 Regressio

Regressiossa pyritään löytämään yhteys tavoitearvo y_i ja syötearvojen x_i välillä. Tavoitearvo on jatkuva reaaliluku; $y_i \in \mathbb{R}$. Tämän lisäksi ainakin yksi syötearvosta on myös realiluku. (Murphy 2012, 8.) Regressio-ongelman tavoite on löytää matemaattinen funktio tai malli, joka selittää tavoite-syötearvot riittävän hyvin. Löydettyä funktiota tai mallia käytetään ennustamaan tulevia tavoitearvoja. (Esposito & Esposito 2020, luku 3.)

Regression sovelluskohteet voidaan jakaa neljään osa-alueeseen: lineaarinen, usean muuttujan lineaarinen, polynomi ja epälineaarinen regressio (Esposito & Esposito 2020, luku 3).

Tyypillisiä regression sovelluskohteita ovat muun muassa (Esposito & Esposito 2020, luku 3; Bonaccorso 2018)

- Hinnan ennustus (talot, osakkeet, energia)
- Tuotannon ennustus (ruoka, tarvikkeet, energia, käytettävissä oleva vesi)
- Talousennustukset
- Aikasarjaennustus

Lineaariregressio (Linear Regression)

Lineaarisessa regressiossa tavoite on riippuvainen vain yhdestä muuttujasta. Riippuvuus voidaan kirjoittaa suoran muotoon (kaava 15)

$$\hat{y}_k \approx a\, x_k + b \tag{15}$$

Algoritmin tulee siis löytää arvot parametreille $a, b \in \mathbb{R}$ sekä minimoida virhefunktion tulos *E*. Perinteiset lineaariset regressioalgoritmit käyttävät keskimääräisen neliösumman virhettä (MSE mean squered error) (kaava 16). (Runkler 2012, 63-67; Jung 2021, 32-33.) MSE:n tilalla virhefunktiona voidaan käyttää myös SSE:a (sum of squared error). MSE:n ja SSE:n ero on, että SSE on n-kertainen MSE. (Hosmer, Lemeshow & Sturdivant 2013, luku 1.3; Esposito & Esposito 2020, luku 11.)

$$E = \frac{1}{n}\sum_{k=1}^{n}\big(y_k - f(x_k)\big)^2 = \frac{1}{n}\sum_{k=1}^{n}(y_k - ax_k - b)^2 \quad (16)$$

Virhefunktion minimiarvo saadaan selville osittaisderivoimalla kaava 16 ensiksi b:n suhteen (kaava 17). Fermet'n teoreeman mukaan kuperan funktion, kuten kaava 16, minimi (tai maksimi) arvo on pisteessä, jossa derivaatta on nolla (Esposito & Esposito 2020, luku 11). Sijoittamalla *b* takaisin virhefunktion kaavaan 16 saadaan laskutoimitus, jossa on vain yksi tuntematon tekijä *a*. Osittaisderivoimalla *a*:n suhteen saadaan ratkaistua *a* (kaava 19 & 20) ja *b* selviää sijoittamalla ratkaistu *a* takaisin kaavaan 17. (Runkler 2012, 63–67.)

$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{k=1}^{n}(y_k - ax_k - b) = 0 \;\Rightarrow\; b = \bar{y}_k - a\bar{x} \;\; , \quad (17)$$

missä $\bar{y}_k$ on tavoitearvojen keskiarvo ja $\bar{x}$ on syötearvojen keskiarvo.

$$E = \frac{1}{n}\sum_{k=1}^{n}(y_k - \bar{y}_k - a(x_k - \bar{x}))^2 \quad (18)$$

$$\frac{\partial E}{\partial a} = -\frac{2}{n}\sum_{k=1}^{n}(y_k - \bar{y}_k)\big(x_k - \bar{x} - a((x_k - \bar{x})\big) = 0 \quad (19)$$

$$a = \frac{\sum_{k=1}^{n}(y_k - \bar{y}_k)(x_k - \bar{x})}{\sum_{k=1}^{n}(x_k - \bar{x})^2} \quad (20)$$

Käytännön sovelluksissa on kuitenkin harvoin vain yksi syötearvo.

Usean muuttujan lineaarinen regressio (Multilinear Regression)

Kun tutkittavassa systeemissä on useita muuttujia ja ne käyttäytyvät lineaarisesti, puhutaan usean muuttujan lineaarisesta regressiosta. Tällöin jokainen muuttuja $x_k^{(i)}$ vaikuttaa systeemin lopputulokseen lineaarisesti. Jokaisella muuttujalla on oma painokerroin a_l. (Runkler 2012, 63–67; Jung 2021, 32–33.)

$$\hat{y}_k \approx \sum_{i=1}^{m} a_i\, x_k^{(i)} + b \quad (21)$$

Usean muuttujan lineaarinen regressio ratkaistaan samoilla periaatteilla kuin yhden muuttujan lineaarinen regressio. Algoritmin tulee löytää arvot parametreille $a_1, \ldots, a_m, b \in \mathbb{R}$ sekä minimoida virhefunktion tulos *E* (kaava 22). (Runkler 2012, 63-67; Jung 2021, 32-33.)

$$E = \frac{1}{n}\sum_{k=1}^{n}\big(y_k - f(x_k)\big)^2 = \frac{1}{n}\sum_{k=1}^{n}\Big(y_k - \sum_{i=1}^{m} a_i x_k^{(i)} - b\Big)^2 \qquad (22)$$

Vakiotermi *b* voidaan ratkaista osittaisderivoimalla virhefunktion kaava 22 ja asettamalla tulos nollaksi. (Runkler 2012, 63-67.)

$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{k=1}^{n}\Big(y_k - \sum_{i=1}^{m} a_i x_k^{(i)} - b\Big) = 0 \;\Rightarrow\; b = \bar{y}_k - \sum_{i=1}^{m} a_i \bar{x}^{(i)}, \qquad (23)$$

missä $\bar{y}_k$ on tavoitearvojen keskiarvo ja $\bar{x}^{(i)}$ on syötearvojen i:n muuttujien keskiarvo. Sijoittamalla *b* virhefunktion kaavaan 18 voidaan virhefunktio kirjoittaa seuraavaan muotoon. (Runkler 2012, 63–67.)

$$E = \frac{1}{n}\sum_{k=1}^{n}\Big(y_k - \bar{y}_k - \sum_{i=1}^{m} a_i\Big(x_k^{(i)} - \bar{x}^{(i)}\Big)\Big)^2 \qquad (24)$$

Kuten yhden muuttujan tilanteessa, tämän jälkeen kaava 24 derivoidaan jokaisen painokertoimen a_i:n suhteen ja lause asetetaan nollaksi ääreisarvon löytämiseksi. (Runkler 2012, 63-67.)

$$\frac{\partial E}{\partial a_r} = -\frac{2}{n}\sum_{k=1}^{n}(y_k - \bar{y}_k)\Big(x_k^{(i)} - \bar{x}^{(i)} - \sum_{i=1}^{m} a_i\Big(x_k^{(i)} - \bar{x}^{(i)}\Big)\Big) = 0\,, r = 1, \dots, m \qquad (25)$$

Kaava 25 voidaan esittää lineaarisena yhtälöryhmänä.

$$\sum_{i=1}^{m} a_i \sum_{k=1}^{n}\Big(x_k^{(i)} - \bar{x}^{(i)}\Big)(y_k - \bar{y}_k) = \sum_{k=1}^{n}\Big(x_k^{(i)} - \bar{x}^{(i)}\Big)(y_k - \bar{y}_k) \qquad (26)$$

Kaavan 26 mukainen yhtälöryhmä voidaan ratkaista useilla eri algoritmeilla, jotka pohjautuvat Gaussin eliminointimenetelmään tai Cramerin sääntöön. Koska kaikki lineaarisen regression parametrit voidaan laskea keskiarvoista ja *X*:n kovarianssimatriisin avulla. Muokkaamalla kaava 21 matriisilaskennan muotoon ja samalla huomioiden kaava 24, pystytään ratkaisemaan painokertoiminen matriisi *A* (kaavat 27–30). (Runkler 2012, 63-67.)

$$Y = \begin{pmatrix} y_1^{(i)} - \bar{y}^{(i)} \\ \vdots \\ y_k^{(i)} - \bar{y}^{(i)} \end{pmatrix} X = \begin{pmatrix} x_1^{(j1)} - \bar{x}^{(j1)} & \cdots & x_1^{(jm)} - \bar{x}^{(jm)} \\ \vdots & \ddots & \vdots \\ x_k^{(j1)} - \bar{x}^{(j1)} & \cdots & x_k^{(jm)} - \bar{x}^{(jm)} \end{pmatrix} A = \begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} \qquad (27)$$

$$Y = X \cdot A \qquad (28)$$

$$X^T \cdot Y = X^T \cdot X \cdot A \quad (29)$$

$$(X^T \cdot X)^{-1} \cdot X^T \cdot Y = A \quad (30)$$

Usean muuttujan lineaarinen regressio on yleinen ongelma, mutta suuressa osassa regressio ongelmia tarvitaan epälineaarista mallia. Tästä huolimatta lineaarisen regression käyttäminen ensimmäisenä regressiona on hyvä vaihtoehto, jos on pienikin epäilys, että ongelma saattaisi ratketa lineaarisesti. (Esposito & Esposito 2020, luku 11.) Lineaarisesta regressiosta on helppo siirtyä epälineaarisiin menetelmiin.

Polynomiregressio (Polynomial Regression)

Tavoite- ja syötearvoja pyritään mallintamaan n-asteen polynomilla. Polynomiregressio on käyttökelpoinen, kun datan riippuvuuksien oletetaan voivan mallintaa kaarevana (Murphy 2012, 3). Stone-Weierstrassin teoreeman mukaan minkä tahansa epälineaarisen riippuvuuden voi esittää riittävällä tarkkuudella käyttämällä polynomifunktiota $\sum_{r=0}^{n} a_r x^{r-1}$ riittävän suurella n-arvolla (Rudin 1976, 159).

Polynomiregressio voi toteuttaa lineaarisena regressiona, laskemalla *x*:n potenssit omiin muuttujiin ennen algoritmiin syöttämistä. Tällöin ongelman ratkaisemiseen voidaan käyttää usean muuttujan lineaariregressio algoritmeja. (Runkler 2012, 67.)

3.5 Virhefunktion minimimointi algoritmeja

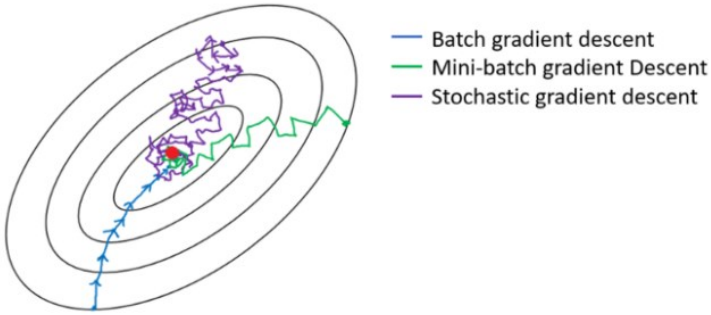
Laskeva gradientti (*Gradient Descent*) on yleisin virhefunktion optimointialgoritmi koneoppimisessa ja syväoppimisessa (Dabbura 2017). Usean muuttujan funktioissa gradientti on eräänlainen kompassi, joka näyttää käyrän suunnan eli vektorin. Vektori saadaan laskettua osittaisderivoimalla virhefunktio jokaisen muuttujan suhteen. Derivaatta kertoo, kuinka nopeasti muuttujan arvo muuttuu. Algoritmin tarkoitus on löytää virhefunktion minimiarvo. Käytännössä laskenta lopetetaan, kun virhefunktion tulos on hyväksyttävien raja-arvojen sisällä. Laskentaa suoritetaan iteratiivisesti muuttamalla muuttujien arvoja oppimiskertoimen α avulla. Jokaisella algoritmin suorituskerralla lasketaan virhefunktion osittaisderivaatat. (Esposito & Esposito 2020, luku 11; Dabbura 2017.)

$$a_n^{(new)} = a_n^{(old)} - \alpha \frac{\partial MSE}{\partial a_n^{(old)}} \quad (31)$$

$$b^{(new)} = b^{(old)} - \alpha \frac{\partial MSE}{\partial b^{(old)}} \quad (32)$$

Oppimiskerroin α on tyypillisesti hyvin pieni luku (0,001–0,3), mutta kuitenkin aina nollan ja yhden välillä. Liian pienellä oppimiskertoimella laskenta saattaa kestää kauan. Algoritmin suoritus saattaa olla rajattu esimerkiksi 2–10 opetusdatan läpikäyntiin (*epoch*), jolloin liian pieni oppimiskerroin ei koskaan ehdi löytää minimiä. Liian suurella oppimiskertoimella saatetaan hypätä minimikohdan yli, jolloin virhefunktion tulos pomppii minimin ympärillä ilman, että minimiä saavutetaan. Sopivalla oppimiskertoimella virhe pienenee aluksi nopeasti, mutta alkaa jo muutaman suorituskerran jälkeen tasaantua hyväksytylle tasolle. (Esposito & Esposito 2020, luku 11; Dabbura 2017.)

Koko opetusdatalla suoritettavaa laskentaa kutsutaan *batch gradient descent* algoritmiksi. Osittaisderivaattojen laskenta koko opetusdatalle on laskennallisesti todella raskasta. Tällöin laskentaa voidaan keventää suorittamalla laskenta satunnaisille opetusdatan syötteille tai satunnaisille muuttujille. Satunnaissyötteet tai muuttujat valitaan jokaisella algoritmin suorituskerralla uudestaan. Satunnaismuuttujilla tehtävää laskentaan kutsutaan *stochastic gradient descent* (SGD). Satunnaistettu alijoukko tunnetaan myös mini-batch gradient nimellä. (Esposito & Esposito 2020, luku 11; Dabbura 2017.) Satunnaistetulle alijoukolle tehtävä laskenta lisää algoritmin yleistyskykyä, koska ne lisäävät kohinaa dataan. Mini-batchin koko on yleensä jokin kahden potensseista (32, 64, 128, 256, 512 jne.) Tämä johtuu siitä, että jotkut prosessorit käsittelevät tehokkaammin kahden potenssin kokoisia datajoukkoja. SGD:n ja mini-batchin huonopuoli on, että ne eivät lähenny virheen minimikohtaa tasaisesti vaan ne vaeltavat kohti minimikohtaa (kuvio 9). (Dabbura 2017.)



KUVIO 9. Laskevan gradientti algoritmien lähentyminen kohti virheen minimikohtaa (Dabbura 2017)

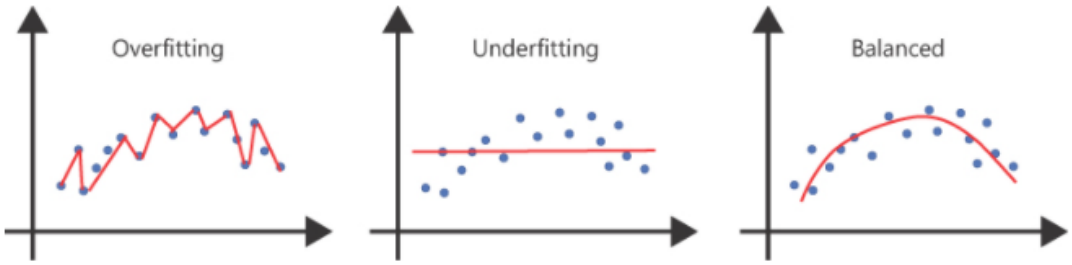
Muita käytettyjä algoritmeja ovat muun muassa (Esposito & Esposito 2020, luku 11.)

- Stochastic Dual coordinate Ascent SDCA
- Gradient Boosting Machine GBM

3.6 Yli- ja alisovittaminen

Koneoppimisen tarkoitus on approksimoida tuntematonta funktiota, joka selittää tavoitearvot syötearvoja käyttämällä. Opetusdata ei voi kuitenkaan sisältää kaikkia mahdollisia kombinaatioita. Opetusdata olisi hyvä olla riittävän yleisluonteinen otos käyttösovelluksen kaikista mahdollista syöte-tavoite-pareista. (Bonaccorso 2018.)

Tyypillinen ongelma koneoppimismallille on, että se on yli- tai alisovitettu. Ylisovitettu malli vastaa lähes täysin opetusdatan aineistoa, mutta opetusdatan ulkopuolisella datalla malli käyttäytyy ei halutulla tavalla. Alisovitettu malli antaa vastaavasti liian ylimalkaisia ja suuntaa antavia tuloksia (kuvio 10). (Esposito & Esposito 2020, luku 10.)



KUVIO 10. Yli- ja alisovitettu sekä tasapainoinen malli (Esposito & Esposito 2020, luku 10)

Mallin estimaattorin $\hat{\theta} = \hat{f}(X) = \hat{Y}$ suoriutumista voidaan mitata laskemalla MSE (*mean squared error*) (kaava 22). Malilla tulisi olla mahdollisimman pieni MSE sekä harha (*bias*) ja varianssi tasapainossa. Mallin harha on keskimääräinen virhe ennustettujen arvojen ja todellisten arvojen $\theta = f(X) = Y$ välillä (kaava 3). Jos harha on nolla kaikilla muuttujan arvoilla, mallin sanotaan olevan harhaton (*unbiased*). Varianssi kuvaa ennustettujen arvojen etäisyyttä datajoukon keskimääräisistä arvoista. Alisovitetuilla malleilla harha on yleensä hyvin suuri. Vastaavasti ylisovitetuilla malleilla varianssi on yleensä hyvin suuri. (Bonaccorso 2018 & Esposito & Esposito 2020, luku 9–10; Jung 2021, 160–161.)

$$MSE[\hat{\theta}] = E_\theta\left[(\hat{\theta} - \theta)^2\right] \quad (33)$$

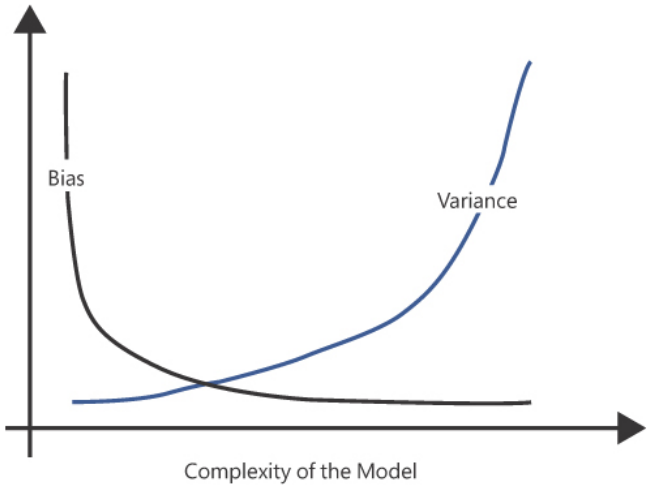
$$Bias[\hat{\theta}] = E_\theta[\hat{\theta}] - \theta \quad (34)$$

$$Variance[\hat{\theta}] = E_\theta\left[(\hat{\theta} - E_\theta[\hat{\theta}])^2\right] \quad (35)$$

MSE voidaan kirjoittaa varianssin ja harhan neliöin summana (kaava 36). (Murphy 2012, 197)

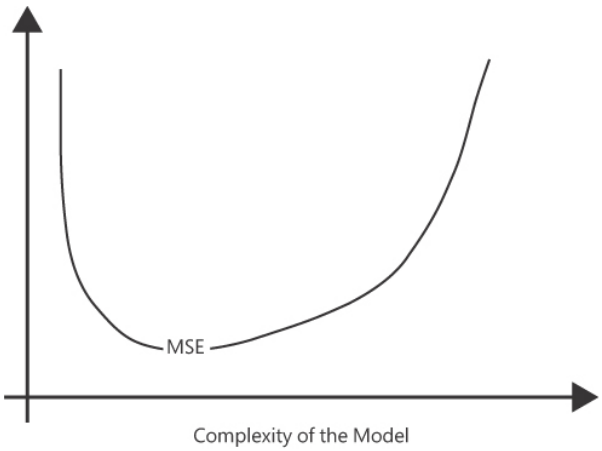
$$MSE(\hat{\theta}) = Variance[\hat{\theta}] + Bias[\hat{\theta}]^2 \quad (36)$$

Mallin kompleksisuus vaikuttaa harhan ja varianssin arvoihin. Kompleksisuutta kasvattamalla malli vastaa paremmin opetusdataa ja tätä kautta sen harha on pienempi. Liiallinen kompleksisuuden kasvattaminen näkyy nopeasti varianssin arvossa (kuvio 11). (Esposito & Esposito 2020, luku 10.)



KUVIO 11. Harha ja varianssi mallin kompleksisuuden funktiona (Esposito & Esposito 2020, luku 10)

MSE:tä käytetään harhan ja varianssin tasopainon löytämiseen. Kuvio 12 osoittaa, että MSE suhteen mallin kompleksisuudelta löytyy tasapainoinen keskialue, jossa mallia voidaan hienosäätää ilman että, että MSE:n arvo merkittävästi muuttuu. Opetusdatan MSE on yleensä aina hyvä, koska muutoin mallin kanssa ei jatkettaisi eteenpäin. Joten testidatan MSE on hyvä mittari mallin toimivuudelle. Ylisovitetun mallin MSE testidatalle on merkittävästi suurempi kuin opetusdatan MSE. (Esposito & Esposito 2020, luku 10 & Bonaccorso 2018.)



KUVIO 12. MSE mallin kompleksisuuden funktiona (Esposito & Esposito 2020, luku 10)

3.7 Ohjaamaton oppiminen

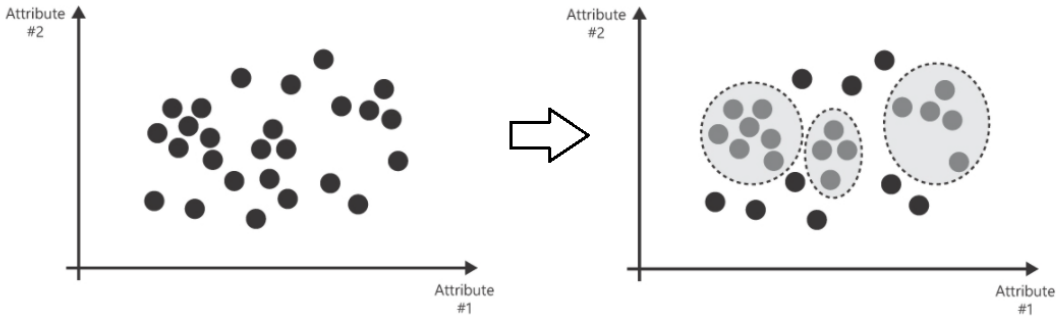
Ohjaamaton oppiminen (*unsupervised learning*) muistuttaa ihmisen tapaa oppia asioita. Oppimisen lopputulosta tai tavoitetta ei tiedetä prosessin alussa. Opetusdatasta pyritään algoritmien avulla tunnistamaan eri syötteiden välillä olevia suhteita, riippuvuuksia ja samankaltaisuutta. Syötteet pyritään jakamaan ryhmiin niin, että samaan ryhmään jaetuilla syötteillä on enemmän samankaltaisia ominaisuuksia kuin muiden ryhmien syötteillä. Tilastollinen lähestymistapa ohjaamattomaan oppimiseen tunnetaan tiheyden arviointina (*density estimation*). (Tuominen & Neittaanmäki 2018, 13; Marsland 2014, luku 1.)

Ohjaamattoman oppimisen opetusdata *X* voidaan esittää kaavan 37 muodossa.

$$X = \{(x_i)\}_{i=1}^{N} \tag{37}$$

Opetusdata ei sisällä tavoitetta, jolloin mallin toimivuuden arviointi on jossain määrin epämääräistä. Ohjaamattomassa oppimisessa ei ole käytössä samanlaisia virhemittareita kuin ohjatussa oppimisessa. (Murphy 2014, 2.)

Syötteiden ryhmittelystä käytetään yleisesti termiä klusterointi. Klusteroinnin algoritmit mittaavat syötteiden loogista etäisyyttä klusterin sisällä. Loogisen etäisyyden mittaustapa on riippuvainen käytetystä algoritmista. Algoritmit saattavat toimia täysin eriperustein. Osa algoritmeista saattaa luokitella syötteiden kuuluvan useaan klusteriin. Yleinen poikkeustapaus on, että syötettä ei sisällytetä mihinkään klusteriin (kuvio 13). (Esposito & Esposito 2020, luku 2.)



KUVIO 13. Syötteiden klusterointi (Esposito & Esposito 2020, luku 2).

Klustereista voidaan laskea etäisyyksiin pohjautuvia arvoja. Etäisyyttä voidaan laskea klusterin muihin pisteisiin, keskipisteeseen tai muiden klustereiden pisteisiin. Mittareilla pyritään intuitiivisesti selittämään, onko klusterin pisteet samankaltaisia tai eroavatko klusterit toisistaan.

Klusterointi analyysia tehdään yleensä suurelle datajoukolle, joka on suurelta osin tuntematonta. Datan määrä onkin usein niin suuri, että sitä ei ihminen pysty hahmottamaan. Tällöin klusterointia käytetään datan tiivistämiseen, jonka jälkeen datalle voidaan tehdä jatkoanalysointia. Tästä syystä datan onkin syytä olla luonteeltaan samankaltaista ja jaettavissa rajattuun lukumäärän klustereita. (Esposito & Esposito 2020, luku 9.)

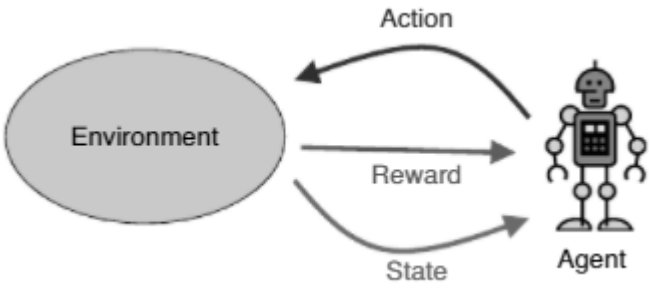
3.8 Vahvistusoppiminen

Vahvistusoppiminen (*reinforcement learning*) on jossain määrin ohjatun ja ohjaamattoman oppimisen välimaastossa. Vahvistusoppimisessa algoritmit saavat palkinnon laadituista toiminnoista, mutta palkinto/palaute ei koskaan sisällä tietoa siitä, miten toimintoa voisi parantaa. Algoritmin tehtävä on tutkia ja kokeilla muita vaihtoehtoja paremman palkinnon toivossa. (Marsland 2014, luku 1; Sutton & Barto 2016, 1–4.)

Ohjatulla ja ohjaamattomalla oppimisella koneoppimismalliin käytettävissä oleva data on jossain määrin vakio. Paljon aikaisemmin muodostettu malli saattaa antaa täysin vääriä ennustuksia, jos ilmiö on nopeasti muuttuva. Malli voidaan opettaa uudestaan uudella datalla, jolloin se on taas hetken käyttökelpoinen.

Vastaavasti vahvistusoppimisen agentti (malli) alkaa antamaan hieman sivuun meneviä toimintoja, kun uutta dataa tulee järjestelmään. Agentin saama palkinto pienentyy tai kääntyy negatiiviseksi, jonka seurauksena agentti joutuu muokkaamaan toimintoja paremman palkinnon saamiseksi. Vahvistusoppimisen sisältää siis yhden ylimääräisen ulottuvuuden, joka on yleensä aika. Tämän tyylinen oppiminen on paljon lähempänä tekoälyä, jona ihmiset sen mieltävät. (Lapan 2020, luku 1 & Sugiyama 2015, 3–5.)

Kuviossa14 esitetään vahvistusoppimisen toimintaperiaate. Vahvistusoppimisen tekoälyagentti havainnoi toimintaympäristöään. Havaintojen perusteella agentti tekee toimintoja, josta se saa palkinnon. Agentti pyrkii maksimoimaan samaansa palkinnon.



KUVIO 14. Vahvistusoppimisen toimintaperiaate (Sugiyama 2015, 4)

Vahvistusoppimisessa ei ole käytössä opetusdataa. Data agentin kehittämiseen saadaan käytön myötä. Vahvistusoppimisessa käytetään hyväksi monia ohjatun oppimisen menetelmiä, joskin hieman eri näkökulmalta. Agentin kehittymisen kannalta sen tulisi saada negatiivisia ja positiivisia palkintoja. Pelkästään negatiivisia palkintoja saava agentti ei voi kehittyä. Tällöin data ei yleensä ole itsenäistä ja tasaisesti jakautunutta (non-i.i.d. data (*independent and identically distriputed*)). Kaikkien ohjattuun oppimiseen perustuvien koneoppimisen menetelmien vaatimus on, että data on itsenäistä ja tasaisesti jakaantunut. (Lapan 2020, luku 1 Sutton & Barto 2016, 1–4.)

Toinen haaste vahvistusoppimisessa on tasapainottaa pyrkimys parempaan palkintoon. Lähtökohtaisesti agentti pyrkii hakemaan koko ajan parempaa palkintoa. Tällöin on riskinä, että agentti ajautuu harhapoluille ja sen saama palkinto huononee. Lisäksi riskinä on, että agentti unohtaa aikaisemmin opitut asiat. Fundamentaalinen dilemma vahvistusoppimisessa on kuinka hyvään palkintoon tulisi pyrkiä. Tähän ei ole yhtä oikeaa vastausta. (Lapan 2020, luku 1; Sutton & Barto 2016, 1–4.)

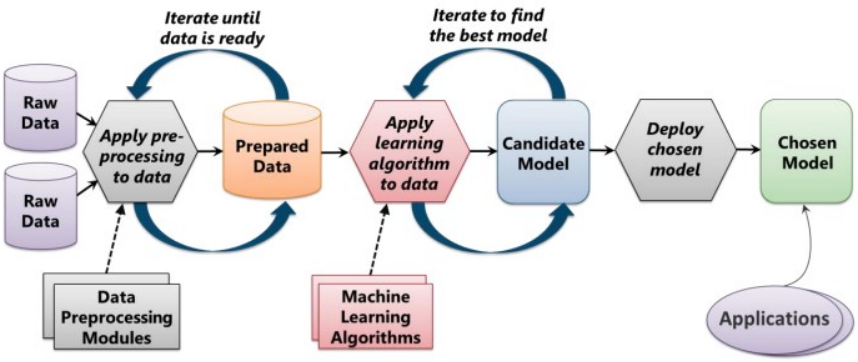
Vahvistusoppimisen kolmas haaste on viive. Ennustuksen ja siitä saatavan palkinnon välillä saattaa olla merkittävä viive. Oppimisessa tulisi havaita syy-seuraus-suhteet, jotka johtivat saatuun palkintoon. Tämä voi olla haastavaa ajan kuluessa ja päätöksiä tai ennustuksia tehtäessä. (Lapan 2020, luku 1; Elements of AI, n.d.) Eri viiveellä saatavat palkinnot diskontataan, jotta niiden arvot olisivat vertailukelpoiset keskenään (Sutton & Barto 2016, 52–53).

4 KONEOPPIMISPROJEKTIN PROSESSI

Koneoppimisongelman ratkaisussa on kolme aineisosaa: data, malli ja virhefunktio. Datalla tarkoitetaan datajoukkoa, joka koostuu syötteistä ja tavoitteista ohjatussa oppimisessa. Ohjaamattomassa oppimissa tavoitteet eivät ole ennalta tiedossa. Data on koneoppimisen tärkein ainesosa. Ennustuksia tehdessä suurin osa koneoppimismalleista nojaa vahvasti tilastollisiin keskiarvoihin. Tällöin datan määrällä on merkitystä. Käytössä olevan datan tulee kuitenkin olla relevanttia ongelman suhteen. Datassa oleva ylimääräiset ominaisuudet tai muuttujat kannattaa poistaa ennen mallin opetusta. Käytössä oleva malli on riippuvainen ongelmasta, johon koneoppimista halutaan soveltaa. Kolmas ainesosa, virhefunktio, kuvaa virhettä, jonka malli tekee ennustustoiminnan aikana. Virheen minimoinnilla voidaan parantaa mallin toimintaa. Virhe on mallin laatumittari. (Jung 2021, 8–16.)

Keskeistä on määritellä tavoite, johon koneoppimisella pyritään pääsemään mahdollisimman tarkasti. Tavoitteen jälkeen tulee selvittää, että minkälaista dataa on saatavilla tavoitteen saavuttamiseksi. Tavoitteen kautta saadaan lähtötiedot analysointiprosessin käyntiin laittamiselle.

Koneoppimisen prosessi muistuttaa pitkälti perinteistä data-analyysia (valmistelu, esikäsittely, analyysi, arviointi ja johtopäätökset) (Runkler 2012, 3). Koneoppimisprosessi on iteratiivinen (kuvio 15). Nopeita kokeiluja pystytään tekemään pienellä datamäärällä, mutta varsinaiseen tuotantoon menevät mallit vaativat useita iteraatioita prosessin läpi, ennen kuin ne täyttävät tuotannon vaatimukset.



KUVIO 15. Koneoppimisprojektin prosessi (Chappel 2015, 5)

Myös koneoppimisessa on hyvä pitää mielessä Occamin partaveitseksi (ongelman ratkaisu) kutsutun filosofian periaate. Sovitettuna koneoppimisongelmaan se tarkoittaa suunnilleen sitä, että ensimmäisen ratkaisumallin tulisi olla yksinkertainen ja vain jos malli ei sovi, siirrytään monimutkaisempiin malleihin. (Bonaccorso 2018; Witten ym. 2017, luku 5.)

4.1 Datan kerääminen ja esikäsittely

Kuten on jo todettu, koneoppimiseen tarvitaan dataa. Opetusdata voi olla valmis kolmannen osapuolen julkaisema datapaketti, mutta hyvin usein opetusdata joudutaan muodostamaan prosessin alussa. Tämä pätee varsinkin tilanteissa, jossa halutaan ratkaista tiettyä, hyvin spesifistä, ongelmaa koneoppimisen avulla. Opetusdata joudutaan usein yhdistelemään useista eri lähteistä, jonka lisäksi datalle tehdään erilaisia yhtenäistämistoimenpiteitä. Keskeinen haaste on myös tunnistaa se, mikä datassa on relevanttia. Ohjatussa oppimisessa opetusdataan tulee yhdistää myös haluttu lopputulos (tavoite). Tämä vaatii usein kohdealan asiantuntijoiden käyttöä sekä merkittävää ajankäyttöä. (Marsland 2014, luku 1.5.)

Myös datan määrällä on lopulta merkitystä. Koneoppimisen algoritmit tarvitsevat suuria datajoukkoja, jotka eivät saisi mielellään sisältää liikaa virheitä ja taustakohinaa. Data on ylivoimaisesti tärkein osa koneoppimista, sillä ilman dataa koneoppiminen on vain teoreettista kaavojen pyörittelyä (Esposito & Esposito 2020, luku 4).

Suurien datajoukkojen myötä myös datan laskentaan liittyvät kustannukset nousevat. Laadukkaan datajoukon koko tulisikin olla riittävän iso ja edustava. Liian suuret datajoukot nostavat laskennallisia kustannuksia ilman, että datan määrästä saadaan varsinaista hyötyä. (Marsland 2014, luku 1.5.)

4.1.1 Opetusdatan määrä

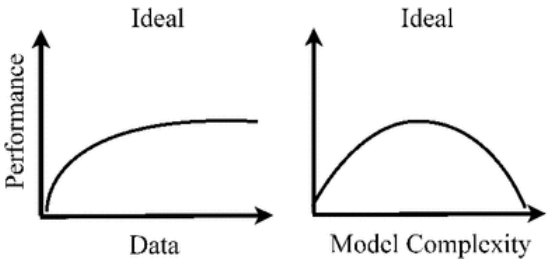
Opetusdata määrittelee tavoitteen mihin koneoppimisella pyritään. Perustavan laatuisena haasteena on, että datan pitäisi selittää ja hahmottaa ilmiötä riittävän tarkasti ja olla koneellisesti luettavissa samanaikaisesti. Opetusdatan laatu riip-

puu ensisijaisesti siitä, kuinka hyvin se kuvaa ihmisen sille määrittelemää tarkoitusta ja kuinka kohtuudella se edustaa ilmiön todellista jakaumaa. Opetukseen käytettävän datan tulee vastata tuotannollista dataa. (Sarkis 2020, luku 1.)

Opetusdatan määrälle ei ole yksiselitteistä laskentakaavaa. Tarvittavaan määrään vaikuttaa ensisijaisesti ilmiön kompleksisuus. Ilmiön mallintamiseen käytettävä algoritmi saattaa kasvattaa tarvittavan opetusdatan määrää. Epälineaariset algoritmit, kuten satunnaismetsät tai neuroverkot, tarvitsevat lähtökohtaisesti paljon enemmän opetusdataan kuin lineaariset algoritmit. (Cogito 2019; Brownlee 2017.)

Tarvittavan opetusdatan määrään voidaan arvioida ongelmakohtaisesti heuristisesti. Luokittelumallin tapauksessa heuristinen arvio voi pohjautua luokkien, muuttujien tai mallin parametrien lukumäärään. Riippumattomia syötteitä pitää olla x kertaa enemmän, kuin luokkien, muuttujien ja parametrien määrä. Kerroin voi olla 10,100, 500 tai suurempi. Kertoimen suuruus riippuu täysin ilmiön kompleksisuudesta. Yleisesti voidaan todeta, että lähes aina enemmän opetusdataa on aina parempi. (Cogito 2019; Brownlee 2017.)

Mallin suorituskykyä voidaan ideaalitilanteessa visualisoida opetusdatan koon ja mallin kompleksisuuden funktiona. Opetusdatan lisääminen parantaa mallin suorituskykyä yleistettävyyden kautta. Opetusdatan lisämäärän merkitys vähenee suurilla tietomäärillä. Mallin kompleksisuuden kasvattaminen parantaa suorituskykyä vain tiettyyn pisteeseen asti, jonka jälkeen suorituskykyä alkaa vähenemään ylisovituksen takia (kuvio 16). Ylisovitus johtuu yleensä mallin kompleksisuudesta tai opetusdatan määrästä. (Zhu ym. 2015.) Kompleksisuudelle ja datan määrälle voidaan määritellä raja-arvot oppimisteorian kautta (McAllester 1999).

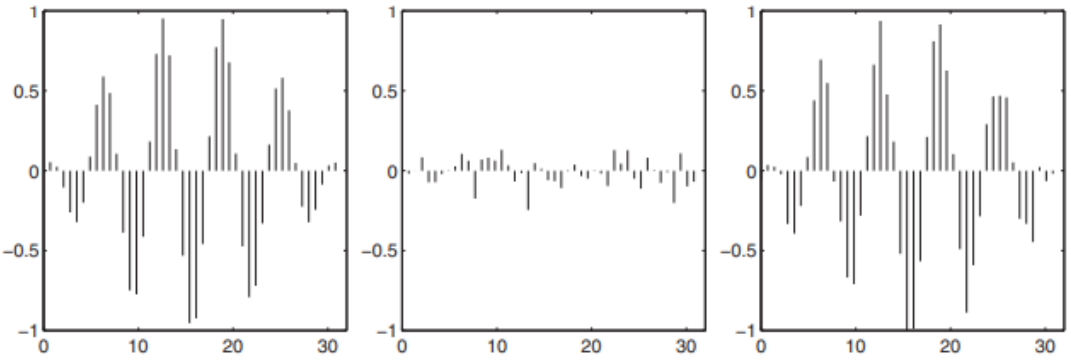


KUVIO 16. Mallin suorituskyky opetusdatan ja mallin kompleksisuuden funktiona ideaalitilanteessa (Zhu ym. 2015)

Kuvia käsittelevät neuroverkot tarvitsevat todella suuren opetusdatan. Opetusdata saattaa olla satoja tai tuhansia kuvia tunnistettavaa luokkaa kohti, jotta malli saavuttaisi hyväksyttävän tason. Kuvantunnistusta neuroverkoilla voidaan käyttää lähtötilanteena esiopetetussa mallissa. Mallin alkutilanteen painoarvot on opetettu esim. ImageNet datalla. Tällöin painoarvoa ei tarvitse hakea alusta asti, vaan riittää että ne säädetään sopivaksi kulloinkin kyseessä olevalla opetusdatalle. Opetusdatan määrä voi olla paljon pienempi kuin nollatilanteesta lähtiessä huomioiden kuitenkin ylisovituksen riskin, joka kasvaa pienillä datamäärillä. Mallin laskenta-aika on paljon pienempi esiopetetuilla painoarvoilla. (Basha ym. 2020; Ng 2015; Hu 2017.)

4.1.2 Datan virheiden käsittely

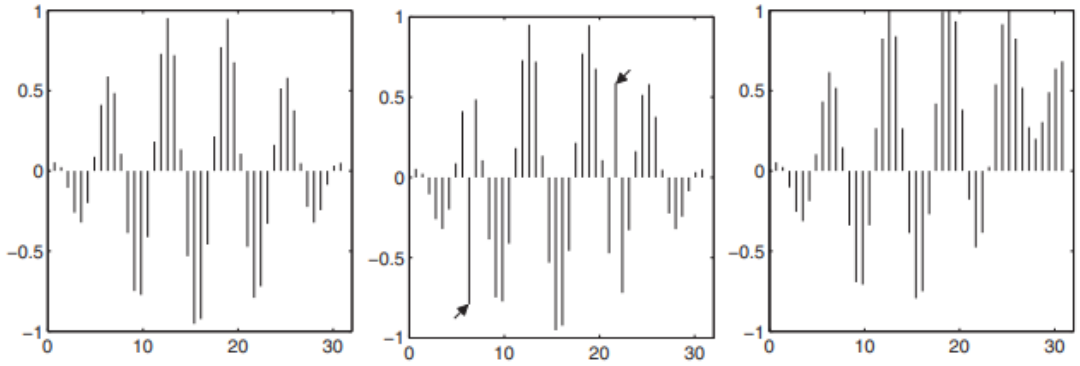
Data on harvoin virheetöntä. Virheellinen data voi johtaa vääriin johtopäätöksiin. Virheet voidaan jakaa kahteen kategoriaan: stokastiset ja deterministiset virheet. Stokastiset virheet ovat yleensä mittausvirheitä tai muunnosvirheitä. Virheen vaikutus dataan ja analysoinnin lopputulokseen on yleensä pieni. Stokastinen virhe on ns. taustakohinaa, jota data sisältää tasaisesti (kuvio 17). (Runkler 2012, 21–22.)



KUVIO 17. Alkuperäinen data, taustakohina ja näiden yhdistelmä, joka muistuttaa alkuperäistä dataa (Runkler 2012, 22)

Datassa esiintyvät epärealistiset arvot ovat yleensä deterministisiä virheitä. Näitä kutsutaan myös poikkeavuuksiksi (*outlier*). Poikkeavuus voi syntyä esimerkiksi merkittävästä mittausvirheestä, data-arvon tallettamisesta väärään sarakkeeseen tai datapaketti saattaa olla hävinnyt tai korruptoitunut. Desimaalimerkin

väärä tulkinta aiheuttaa deterministisen virheen. Datassa esiintyvä poikkeavuus on usein niin merkittävä, että se pitää jollain tavalla korjata ennen analyysiä. Data voi olla myös vinoutunutta esimerkiksi väärin suoritetun kalibroinnin tai muun systemaattisen mittausvirheen takia (kuvio 18). Tällöin deterministinen virhe voidaan korjata vain, jos virheen systematiikka tunnetaan tarkasti. (Runkler 2012, 21–22.)



KUVIO 18. Alkuperäinen data, poikkeavuuden sisältävä data (poikkeavuus merkitty nuolella) ja vinoutunut data (Runkler 2012, 22)

Kuviossa 18 esitettyä poikkeavuutta ei ole helppoa havaita, koska sen arvo on datan normaalin arvojen sisällä. Datan normaali arvojen sisällä olevaa poikkeavuutta kutsutaan myös termillä *inlier*. Aikasarja datassa *inlier* voidaan havaita ja käsitellä käyttämällä menetelmiä, jotka huomioivat poikkeavan muutoksen datassa. Helpommin havaittavissa ovat poikkeavuudet, jotka poikkeavat merkittävästi datan normaaleista arvoista. Tällöin poikkeavuus voidaan havaita esimerkiksi 2-sigma säännöllä. (Runkler 2012, 23.)

$$x_k \in X \; on \; outlier \;\Leftrightarrow\; \exists i \in \{1, \dots, p\}. \left| \frac{x_k^{(i)} - \bar{x}^{(i)}}{s_x^{(i)}} \right| > 2, \tag{37}$$

missä $\bar{x}^{(i)}$ tarkoittaa muuttujan *i* keskiarvoa (kaava 38) ja $s_x^{(i)}$ on *i* muuttujan keskihajonta (kaava 3.3).

$$\bar{x}^{(i)} = \frac{1}{n} \sum_{k=1}^{n} x_k^{(i)} \tag{38}$$

$$s_x^{(i)} = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} \left(x_k^{(i)} - \bar{x}^{(i)} \right)^2} = \sqrt{\frac{1}{n-1} \left(\sum_{k=1}^{n} \left(x_k^{(i)} \right)^2 - n(\bar{x}^{(i)})^2 \right)} \tag{39}$$

Datassa saattaa olla 2-sigma säännön mukaan poikkeavuuksiksi tulkittavia arvoa, mutta ne saattavat olla myös täysin oikeita arvoja. Tässä tapauksessa näitä poikkeavuuksia ei tule poistaa datasta, koska se vääristävät dataa. Datan tunteminen on avain asemassa poikkeavuuksien käsittelyssä. Puhtaasti mittausvirheestä syntyneet arvot voidaan havaita asettamalla muuttujille rajat (minimi ja maksimi). Muuttujan arvon ollessa rajojen ulkopuolella arvo on virheellinen (kaava 40) (Runkler 2012, 23).

$$x_k \in X \; on \; virheellinen \; \Leftrightarrow \; \exists i \in \{1, \dots, p\}. \left(x_k^{(i)} < x_{min}^{(i)}\right) \vee \left(x_k^{(i)} > \; x_{max}^{(i)}\right) \qquad (40)$$

Virheiden, poikkeavuuksien tai puuttuvan datan käsittelyllä on käytettävissä menetelmiä, josta parhaiten soveltuva riippuu lähdedatasta (Runkler 2012, 24; Esposito & Esposito 2020, luku 4).

1. Dataan lisätään sarake, joka kertoo virheettömyydestä/virheellisyydestä. Alkuperäinen data pysyy muuttumattomana.
2. Virheellinen arvo korvataan perusarvolla.
3. Virheellisen tai puuttuvan arvon korjaaminen tai arviointi
    a. Korvataan arvo datan oikeiden arvojen keskiarvolla, mediaanilla, minimillä tai maksimilla.
    b. Korvataan arvo lähimmän naapurin arvolla ilman, että huomioidaan puuttuvaa muuttujaa *i*.

$$x_k^{(i)} = x_j^{(i)}, \text{ kun } \left\|x_j - x_k\right\|_{\neg i} = \min_{l \in \{1,\dots,n\}} \|x_l - x_k\|_{\neg i} \qquad (41)$$

    c. Lineaarinen interpolointi aikasarjassa, jossa näytteet yhtä kaukana toisistaan.

$$x_k^{(i)} = \frac{x_{k-1}^{(i)} - x_{k+1}^{(i)}}{2} \qquad (42)$$

    d. Lineaarinen interpolointi aikasarjassa, jossa näytteet eivät ole yhtä kaukana toisistaan.

$$x_k^{(i)} = \frac{x_{k-1}^{(i)} \times (t_{k+1} - t_k) - x_{k+1}^{(i)} \times (t_k - t_{k-1})}{(t_{k+1} - t_{k-1})} \qquad (43)$$

    e. Epälineaarinen interpolointi
    f. Suodatus

   g. Regressio
4. Virheellinen arvon poistaminen: x_k vektori poistetaan kokonaisuudessaan

Virheellisen arvon poistamista datajoukosta tulisi välttää. Erityisesti sellaisissa tilanteissa, joissa ei ole käytettävissä hyvin suurta datajoukkoa. Kohdassa 3b-g esitetyt matemaattiset keinot korjata virheellinen arvo johtavat helposti merkittävään lisälaskentaan datajoukolla. Joten hyvin usein käytännön kannalta paras vaihtoehto on korvata virheellinen arvo perusarvolla tai 3a:n mukaisella arvolla. (Bonaccorso 2018.) Puuttuvat arvot aiheuttavat yleensä ongelmia algoritmien kanssa. Ne tulee korjata sovellukseen sopivalla tavalla.

Osa koneoppimisen sovelluksista on poikkeavuuksien havainnointi (*anomaly detection*) datasta. Varsinkin näiden sovellusten opetusdatasta ei tule poistaa poikkeavuuksia.

4.2 Muuttujien valinta

Kerätty data saattaa sisältää paljon muuttujia, jotka eivät ole oleellisia ongelman ratkaisun kannalta. Tämänkaltaiset muuttujat kannattaa karsia pois opetusaineistoista. (Marsland 2014, luku 1.5; Runkler 2012, 76.) Tällä tavalla kevennetään mallin opetuksen laskennallisia kustannuksia, sillä laskenta-aika kasvaa eksponentiaalisesti muuttujien määrän kasvaessa. Lisäksi ylisovittamisen riski kasvaa muuttujien määrän kasvaessa. (Asaithambi 2018.)

Muuttujien karsintaa voidaan tehdä manuaalisesti tai siihen voidaan käyttää algoritmeja. Manuaalisessa karsimisessa riskinä on, että piileviä yhteyksiä ei havaita. Datan ja ilmiön tunteminen on tärkeässä asemassa. Algoritmeihin perustavat tavat, joita on mm. varianssin kynnysarvo ja korrelaatioanalyysi, vähentävät ainakin inhimillisen virheen mahdollisuutta. (Esposito & Esposito 2020, luku 4.)

Ongelmasta riippuen alkuperäisistä muuttujista voidaan muodostaa laskennallisia arvoja (Esposito & Esposito 2020, luku 4). Esimerkiksi lääketieteellisen mallin kannalta BMI voi olla hyödyllisempi kuin paino ja pituus erikseen. Tämä vähentää yhden muuttujan opetusdatasta olettaen, että alkuperäiset pituus ja paino poistetaan BMI:n laskennan jälkeen.

Muuttujien arvoja voidaan myös ryhmitellä, eli kategorisoida. Esimerkiksi karttakoordinaatit voidaan ryhmitellä paikkakuntien mukaisiin ryhmiin. Koska algoritmit ymmärtävät vain numeraalisia arvoja, täytyy kategoriat muuttaa numeraaliseksi. (Esposito & Esposito 2020, luku 4.) Diskreettejä numeroita ei voida käyttää algoritmeissa, joten kategoriset muuttujat muutetaan niin kutsutuiksi dummy-muuttujiksi. Dummy-muuttujien optimaalinen lukumäärä on kategorioiden lukumäärä vähennettynä yhdellä. Dummy-muuttuja voi saada arvon nolla tai yksi. Taulukossa 2 on esitetty neljän paikkakunnan dummy-muuttujat.

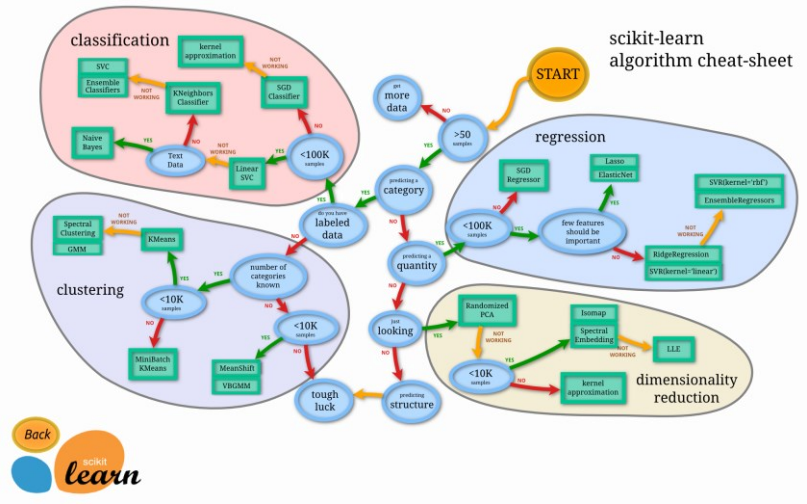
TAULUKKO 2x. Esimerkki kategorisen muuttujan esittäminen dummy-muuttujina

	Dummy-muuttujat		
Kategoria	Helsinki	Vantaa	Espoo
Helsinki	1	0	0
Vantaa	0	1	0
Espoo	0	0	1
Kauniainen	0	0	0

Syvät neuroverkot pystyvät käsittelemään suuria muuttuja määriä ilman samanlaisia ongelmia kuin perinteisissä koneoppimisalgoritmeissa. Neuroverkko algoritmit tekevät muuttujien valinnat ja painotukset osana algoritmin toimintaa. (Sarkis 2020, luku 1.)

4.3 Algoritmin valinta

Algoritmin valintaan vaikuttavat lähdedata, sovellus ongelma ja tavoite. Koneoppimisprosessin aikaisemmissa vaiheissa kohdealan tuntemus oli avainasemassa. Tästä vaiheesta eteenpäin tärkeään asemaan nousee tekoälyn ja koneoppimisen tuntemus. Algoritmien toimintaperiaatteiden tuntemus auttaa oikean valinnan tekemisessä. Algoritmeista on laadittu *cheat-sheet*-kaavioita, jotka helpottavat ja ohjaavat oikeiden algoritmien suuntaan. Kuviossa 44 on esitetty scikit-learn kirjaston *cheat-sheet*.



KUVIO 44. Scikit-learn ohjelmakirjaston algoritmien *cheat-sheet* (Scikit-learn n.d.)

Algoritmeilla on yleensä niin kutsuttuja hyperparametreja, jotka vaikuttavat lopputulokseen. Hyperparametrien arvot joudutaan usein hakemaan manuaalisena työnä tai rakentamalla ohjelma, joka opettaa mallia eri hyperparametrien arvoilla. Useat ohjelmakirjastot sisältävät valmiita apuohjelmia hyperparametrien optimointiin. Näitä menetelmiä kutsutaan *grid-search* ohjelmiksi.

4.4 Mallin opetus

Ennen mallin opetusta käytössä oleva data jaetaan opetusdataan ja validointidataan. Validointidatan avulla vahvistetaan mallin toimivuus. Jako datajoukkoihin tehdään yleensä 80/20 tai 70/30 suhteella. Datajoukkojen muodostamisessa on huomioitava, että molempien joukkojen tulisi edustaa jakaantumaltaan alkuperäistä datajoukkoa. Lisäksi jaon tulisi tapahtua satunnaisesti, jotta vältytään valitsemasta peräkkäisiä dataelementtejä. Peräkkäiset dataelementit saattavat olla jollain tasolla riippuvaisia toisistaan. (Esposito & Esposito 2020, luku 4; Bonaccorso 2018.)

Mallin opetukseen liittyy virhefunktion minimointi. Opetuksessa pyritään minimoimaan opetusdatan virhefunktion tulos. Minimointi tapahtuu muuttamalla algoritmin sisäisiä painoarvo ja se tapahtuu iteratiivisesti. Opetusdata käydään läpi useita kertoja ja painoarvoihin tehdään pieniä päivityksiä jokaisella kierroksella. Lopulta virhefunktion tulos saavuttaa hyväksyttävän tason tai hyperparametriin määritelty opetusdatan läpikäynti arvo (*epoch*) tulee täyteen.

4.5 Mallin arviointi

Opetusprosessista ulos saatavaa mallia testataan datalla, jota ei ole käytetty opetukseen. Mallin arvioinnin tarkoitus on varmistaa, että malli pystyy yleistämään saamansa opetuksen entuudestaan tuntemattomalle datalle. Testidatalle on ratkaisevan tärkeää, että datan tilastollinen merkitys ja relevantit muuttujat ovat päteviä ja linjassa opetusdatan kanssa. Opetus- ja testidatan tulee olla linjassa odotetun tuotantodatan kanssa. (Esposito & Esposito 2020, luku 4.)

Mallin suoriutumista voidaan mitata esimerkiksi tarkkuuden tai keskimääräisen virheen avulla. Mittari ja hyväksyttävä raja-arvo riippuu sovelluksesta. Raja-arvon asettamista voidaan pohtia seuraavan kysymyksen kautta: *mitkä ovat seuraukset, jos mallin ennustus on väärä ja voidaanko se hyväksyä n:ssä osassa tapauksia?* Koneoppimisprosessin iteratiivisen luonteen takia tässä vaiheessa palataan yleensä säätämään algoritmia tai keräämään lisää dataa, kuten kuvassa 15 on esitetty. Hyväksytty malli julkaistaan ja sen käyttäminen tapahtuu pääsääntöisesti ulkopuolisella ohjelmalla, joka kutsuu mallia ohjelmallisen rajapinnan kautta (API).

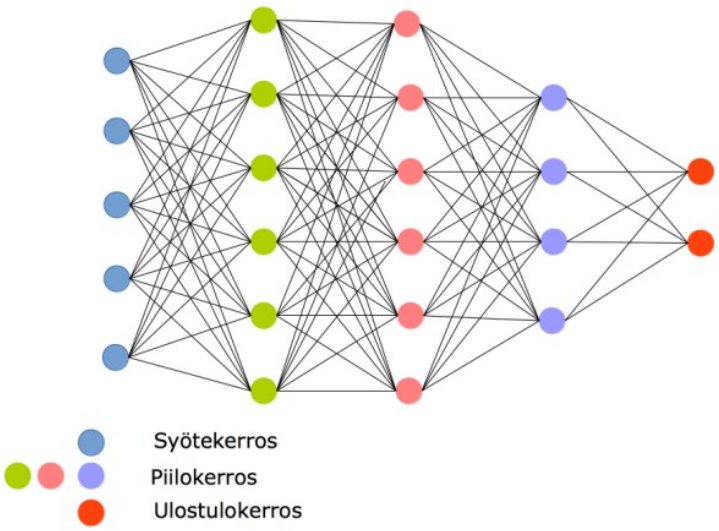
5 NEUROVERKOT

5.1 Neuroverkon arkkitehtuuri

Useat koneoppimisen ongelmista voidaan yksinkertaistaa luokitteluongelmiksi. Luokittelua on esimerkiksi objektientunnistus kuvasta tai tekstin tulkitseminen. Tämän tyylisiä ongelmia voidaan ratkaista eteenpäin kytkeytyvien neuroverkkojen (*deep feedforward neural network*) avulla tai usean kerroksen perseptroniverkolla (*multilayer perceptron* MLP). (Murphy 2012, 999–1000.)

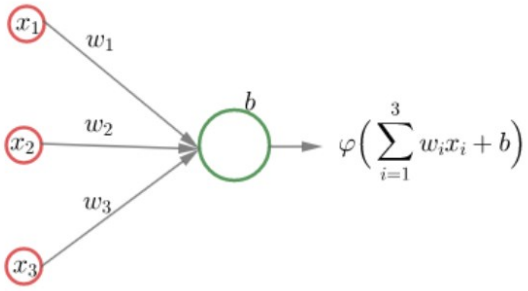
Neuroverkko rakentuu syöte-, piilo- ja ulostulokerroksista (kuvio 4.2). Piilokerroksia voi olla useita, jopa tuhansia. Tällöin neuroverkon sanotaan olevan syvä neuroverkko (*deep neural network*). Kerrokset koostuvat neuroneista, joilla on parametreja. Parametrejä muutetaan verkon opetuksessa. (Tuominen 2019, 23–28.) Neuroverkon parametrien lukumäärä voi olla jopa miljoonia (Murphy 2012, 999–1000).

Myös neuroverkkoa voidaan ajatella kaavan 2 tapaan tuntemattomana funktiona $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$; syöte on n-ulotteinen vektori $x = (x_1, \dots, x_n)$ ja tuloksena saadaan m-ulotteinen vektori $y = f(x) = (y_1, \dots, y_m) \in \mathbb{R}^m$. Neuroverkon käyttötarkoitus selittää, miten tulosta tulkitaan. (Tuominen 2019, 23–28.)



KUVIO 45. Eteenpäin syöttävän neuroverkon arkkitehtuuri: syöte-, piilo- ja ulostulokerros (Tuominen 2019, 25)

Data syötetään neuroverkkoon syötekerroksen kautta. Syötekerroksen neuroneiden lukumäärä on riippuvainen siitä, montako ominaisuutta syötedata sisältää. Syötekerros välittää datan piilokerroksille, jossa datasta lasketaan neuronikohtainen painotettu keskiarvo ja lisätään neuronin vakiotermi. Ennen neuronin tuloksen välittämistä seuraavalla neuronille summa viedään aktivointifunktiolle φ (kuvio 46). Aktivointifunktio φ muuttaa lineaarisen syötteen epälineaariseksi. (Tuominen 2019, 23–25.)



KUVIO 46. Sisääntulojen painotettu summa ja vakiotermi viedään aktivointifunktion φ läpi, josta muodostuu neuronin ulostulo (Tuominen 2019, 35)

Jokaisella neuronilla on kahdenlaisia parametreja, neuroneiden välillä olevat painoarvot (*weight*) ja neuronin vakiotermi (*bias*). Neuronin tulos eli seuraavalle kerrokselle vietävä arvo saadaan kaavalla 4.1. (Tuominen 2019, 27–29.)

- l = kerrosindeksi (l=0 viittaa syötekerrokseen ja l=L ulostulokerrokseen)
- N_l = kerroksen *l* neuroneiden lukumäärä
- $a_i^0 = x_i$ syötedata (N_0 kappaletta)
- ω_{ij}^l = kerroksen *l – 1* neuronin *i* ja kerroksen *l* neuronin *j* välillä vaikuttava painoarvo
- b_j^l = kerroksen *l* neuronin *j* vakiotermi

$$a_j^l = \varphi(z_j^l) = \varphi\left(\sum_{i=1}^{N_{l-1}} \omega_{ij}^l a_i^{l-1} + b_j^l\right), \quad (47)$$

missä φ on aktivointifunktio, joka saattaa vaihdella kerroksien välillä. Kaava 47 voidaan kirjoittaa vektori- ja matriisimuodossa (kaava 48). Kaavassa 48 vektorit on määritelty vaakavektoreiden ja W_l painokerroin matriisissa neuroneiden indeksit ovat järjestyksessä *i* ja *j*. (Tuominen 2019, 27–29.)

$b^l = \left(b_1^l, \dots, b_{Nl}^l\right)$, kerroksen *l* vakiotermien vektori

$z^l = \left(z_1^l, \dots, z_{Nl}^l\right)$, kerroksen *l* neuroneiden painotettu summa vektori

$a^l = \left(a_1^l, \dots, a_{Nl}^1\right)$, kerroksen *l* neuroneiden tulos vektori

$W^l = \begin{pmatrix} \omega_{11}^l & \omega_{12}^l & \dots & \omega_{1N_l}^l \\ \omega_{21}^l & \omega_{22}^l & \dots & \omega_{2N_l}^l \\ & & \dots & \\ \omega_{N_{l-1}1}^l & \omega_{N_{l-1}2}^l & \dots & \omega_{N_{l-1}N_l}^l \end{pmatrix}$, kerroksen *l* painokertoimet N_{l-1} x N_l – matriisi

$$a^l = \varphi(z^l) = \left(\varphi\left(z_1^l\right), \dots, \varphi\left(z_{N_l}^l\right)\right), \tag{48}$$

jossa painotettu summa z_l on laskettavissa kaavalla 49.

$$z^l = a^{l-1}W^l + b^l \tag{49}$$

5.2 Perseptroni

Yksittäinen perseptroni on minimaalinen neuroverkko. Perseptroni koostuu syötekerroksesta ja yhdestä neuronista. Syöte on n-ulotteinen vektori ja tulos on $y \in \{0,1\}$. Perseptronin aktivointifunktio on yksikköaskelfunktio (Heavisiden funktio). Aktivointifunktion tuloksena saadaan nolla, kun *x* on negatiivinen ja yksi, kun *x* on positiivinen (kaava 50). (Tuominen 2019, 30–34.)

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{50}$$

Perseptronilla voidaan erotella syötteet tuloksiin kyllä (1) ja ei (0), jos ne ovat lineaarisesti eroteltavissa. Perseptronin ominaisuus on, että pienetkin muutokset painokertoimissa tai syötteessä voivat aiheuttaa ison muutoksen tuloksessa (0/1). Verkon opetuksen kannalta tämä on huono asia. (Tuominen 2019, 30–34.)

Perseptroni itsestään ei ole kovin käyttökelpoinen, mutta pienillä aktivointifunktion muutoksilla ja rakentamalla verkko koostumaan suuresta määrästä rinnakkain ja sarjassa olevista perseptroneista saadaan aikaiseksi tehokasta neuroverkko. (Marsland 2014, luku 4)

5.3 Aktivointifunktiot

Yleensä opetukseen käytetään yksikköaskelfunktion sijasta jotain paremmin opetukseen soveltuvaa aktivointifunktiota. (Tuominen 2019, 30–34.)

Aktivointifunktiolla olisi toivottavaa olla seuraavat ominaisuudet:

- epälineaarisuus: mahdollistaa neuroverkolla minkä tahansa epälineaarisen funktion oppimisen, kun neuroverkko sisältää riittävästi neuroneita. (Aghdam & Heravi 2017, luku 2.6.2.)
- derivoituvuus: vastavirta-algoritmeissa (backpropagation) sekä muissa virhefunktion minimointi tavoissa käytetään aktivointifunktion derivaattaa. Aktivointifunktion ollessa derivoitava voidaan käyttää yleisiä gradienttiin perustuvia keinoja (*gradient descent*). (Tuominen 2019, 35; Aghdam & Heravi 2017, luku 2.6.2.)
- identtisen funktion approksimointi: Neuroverkko, jonka aktivointifunktio on nollan lähialueilla lähellä identtistä funktiota, oppii tehokkaasti. Neuroneiden painoarvot voidaan alustaa satunnaisluvuilla. Muilla aktivointifunktiolla painoarvot tulee alustaan huolellisesti tehtävänä manuaalisena työnä. (Tuominen 2019, 35.)

Aktivointifunktio voi olla rajallinen tai rajaton. Tämä vaikuttaa oppimisen vakauteen ja nopeuteen. Rajatut aktivointifunktiot ovat yleensä vakaita oppimaan ja rajoittamattomat oppivat tehokkaasti. (Tuominen 2019, 35.)

Tyypillisiä aktivointifunktiota ovat muun muassa sigmoid-funktio, hyperbolinen tangentti ja ReLu-funktio. Toimivin aktivointifunktio riippuu siitä, mitä neuroverkollaan tekemässä. (Tuominen 2019, 35–39.)

5.3.1 Sigmoid-funktio (logistinen funktio)

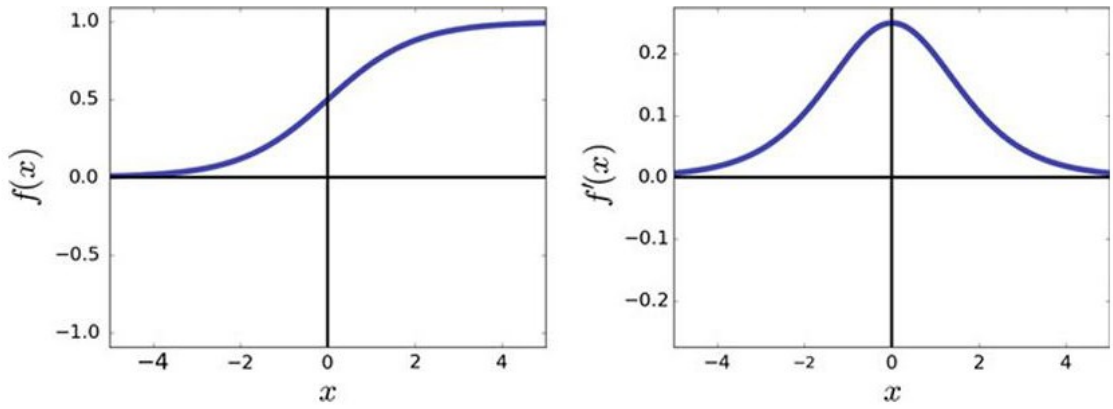
Sigmoid-funktio on muokattu yksikköaskelfunktio $\sigma : \mathbb{R} \to]0,1[$,

$$\sigma(x) = \frac{1}{1+e^{-x}} \ . \tag{51}$$

Ominaisuuksiltaan sigmoid-funktio on

- rajoitettu, jatkuva ja aidosti kasvava
- $lim_{x\to-\infty}\sigma(x) = 0, lim_{x\to\infty}\sigma(x) = 1$
- funktio on kaikilla kertaluvuilla jatkuvasti derivoitavissa ja

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)\big(1-\sigma(x)\big). \tag{52}$$



KUVIO 19. Sigmoid-funktio ja derivaatta nollan lähistöllä (Aghdam & Heravi 2017, luku 2.6.2.)

Sigmoid-funktion huono ominaisuus on, että tulos kasvaa hitaasti *x*:n kasvaessa ja vastaavasti pienenee hitaasti *x*:n pienentyessä (kuvio 19). Funktion derivaatta (kaava 52) on lähellä nollaa, kun *x* on suuri tai pieni. Neuroverkko oppii hitaasti, jos virhefunktioiden osittaisderivaatat ovat pieniä. Sigmoid-funktio ei ole symmetrinen nollan suhteen. Tästä seuraa se, että sigmoid-funktiota käytetään lähinnä ulostulokerroksen aktivointifunktiona, kun verkon tulokset ovat välillä [0,1]. (Tuominen 2019, 35–39.)

5.3.2 Hyperbolinen tangentti (tanh)

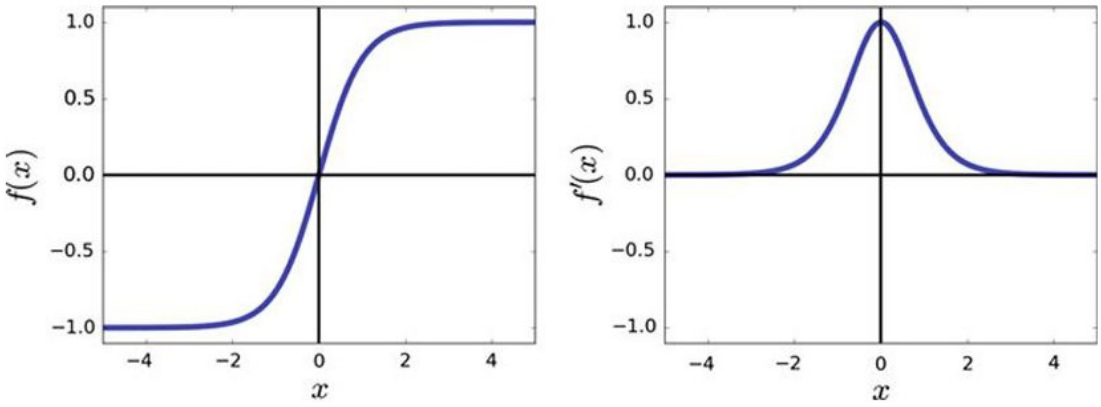
Hyperbolinen tangentti $tanh\colon \mathbb{R} \to]-1,1[$,

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}. \tag{53}$$

Funktiolla on useita samoja ominaisuuksia kuin sigmoid-funktiolla, mutta tanh-funktio on symmetrinen nollan suhteen (kuvio 20). Tämän lisäksi se kasvaa nopeammin nollan lähellä, jolloin sen derivaatta on myös suurempi. (Tuominen 2019, 35–39.)

Hyperbolisen tangentin ominaisuuksia on

- rajoitettu, jatkuva ja aidosti kasvava
- $lim_{x\to-\infty} tanh(x) = -1, lim_{x\to\infty} tanh(x) = 1$
- $tanh'(x) = 1 - tanh^2(x)$



KUVIO 20. Hyperbolinen tangentti ja derivaatta nollan lähistöllä (Aghdam & Heravi 2017, luku 2.6.2.)

Hyperbolisen tangentin ollessa aktivointifunktiona saattaa neuroverkon opettamisessa esiintyä hitautta. Tämä johtuu gradientin (derivaatan) pienuudesta isoilla ja pienillä arvoilla. (Tuominen 2019, 35–39.)

5.3.3 Softsign

Softsign-funktio on hyperbolisen tangentin lähisukulainen, mutta sillä on toivottavampia ominaisuusia. Softsign-funktio määritellään; $softsign: \mathbb{R} \to]-1,1[$,

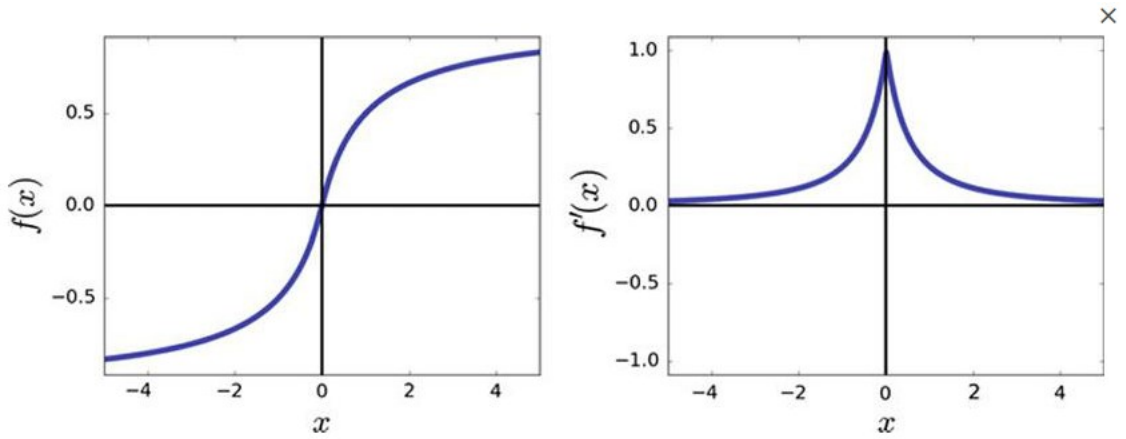
$$\text{softsign}(x) = \frac{x}{1+|x|}. \tag{54}$$

Softsign-funktio saa arvon nolla, kun x:n arvo on nolla. Funktion derivaatta on yksi x:n ollessa nolla, joten Softsign-funktiolla on identtisen funktion ominaisuuksia nollan lähistöllä. Funktion derivaatan arvo pienenee nollan läheisyydessä voi-

makkaasti, mutta kauempana nollasta pieneminen on hitaampaan kuin hyperbolisella tangentilla. Tämä on toivottava ominaisuus aktivointifunktiolta. Näiden lisäksi softsign-funktion laskenta vaatii prosessoreilta vähemmän laskentakapasiteettia. Edellä mainittujen ominaisuuksien johdosta softsign-funktiota voidaan käyttää vaihtoehtona hyperboliselle tangentille. (Aghdam & Heravi 2017, luku 2.6.2.)

Softsign-funktion ominaisuuksia on

- rajoitettu, jatkuva ja aidosti kasvava
- $lim_{x\to-\infty} softsign(x) = -1, lim_{x\to\infty} softsign(x) = 1$
- $softsign'(x) = \frac{1}{(1+|x|)^2}$

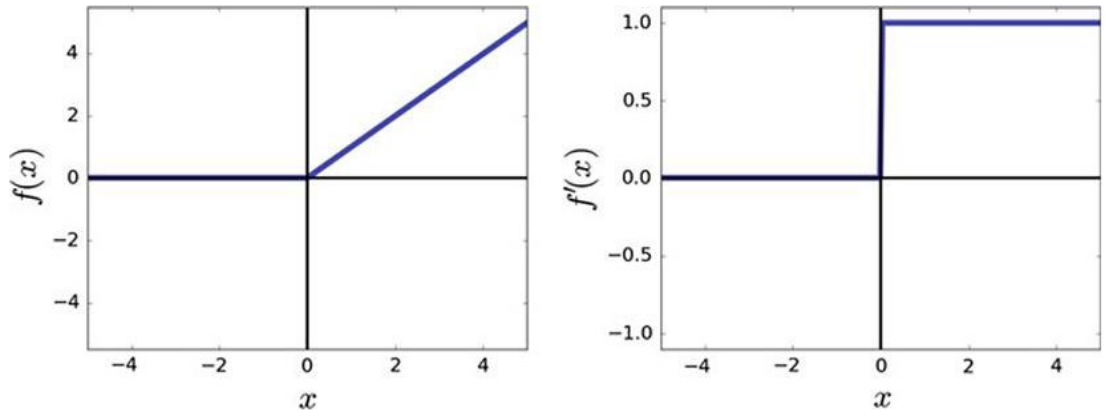


KUVIO 21. Softsign-funktio ja derivaatta nollan läheisyydessä (Aghdam & Heravi 2017, luku 2.6.2.)

5.3.4 ReLu

ReLu (Rectified Linear Unit) on yleinen neuroverkkojen piilokerroksissa käytetty aktivointifunktio. ReLu-funktio määritellään $f: \mathbb{R} \to [0, \infty[$,

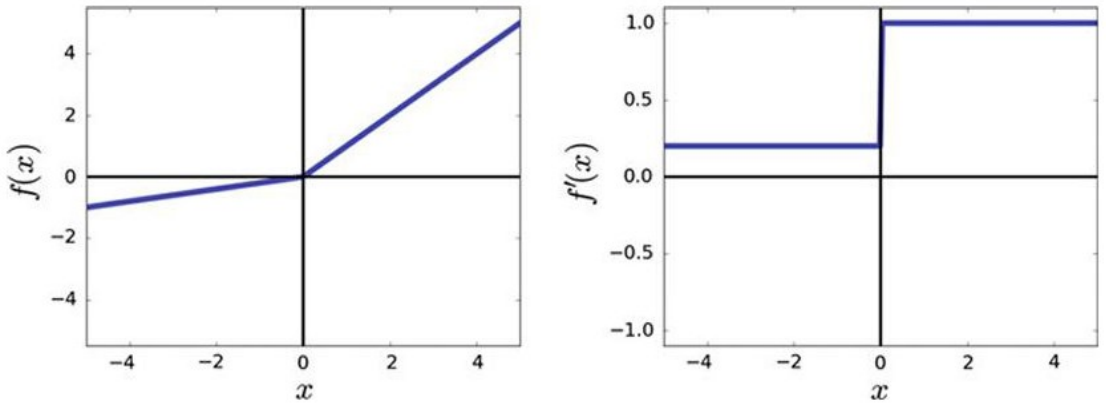
$$f(x) = max\{0, x\}. \tag{55}$$



KUVIO 22. ReLu-funktio ja derivaatta nollan lähistöllä (Aghdam & Heravi 2017, luku 2.6.2.)

ReLu-funktio ei ole derivoitavissa nollassa. Tämän lisäksi funktion derivaatta on negatiivisilla arvoilla nolla. Tästä seuraa se, että opetuksen aikana joidenkin neuronien painokertoimet saattavat päivittyä nollaksi. Tällöin neuronin ulostulona on nolla. Tällöin neuronin sanotaan olevan kuollut. Neuronien kuoleentumista pyritään välttämään. ReLu-funktiosta on kehitetty eri variaatioita, joilla tämä ongelma pyritään välttämään. Yksi Relun variaatio on ”Leaky Relu”, $f: \mathbb{R} \rightarrow \mathbb{R}$, (Tuominen 2019, 35–39.)

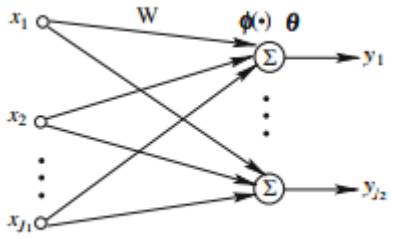
$$f(x) = max\{ax, x\}, 0 < a < 1. \tag{56}$$



KUVIO 23. Leaky Relu-funktio ja derivaatta nollan lähistöllä (Aghdam & Heravi 2017, luku 2.6.2.)

5.4 Single layer perceptron SLP

Yhden kerroksen sisältävää neuroverkkoa, jonka aktivointifunktio on yksikköaskelfunktio, kutsutaan termillä *single layer perceptron* (SLP). SLP:lla voidaan luokitella syötedata useampaan luokkaan (kuvio 24). (Du & Swamy 2014, 68–69)



KUVIO 24. Single layer perceptron SLP:n periaatekaavio (Du & Swamy 2014, 69)

5.5 MLP (multilayer perceptron)

MLP on yksi- tai useampikerroksinen eteenpäin kytkeytyvä neuroverkko, jota voidaan käyttää luokittelu- ja regressio-ongelmien ratkaisemiseen. Tilanteesta riippuen sovitus voidaan tehdä kolmella tai neljällä kerroksella. Matemaattisesti on todistettu, että kolmekerroksinen MLP, jossa aktivointifunktiona toimii sigmoidfunktio, pystyy sovittamaan minkä tahansa jatkuvan funktion. Neljän kerroksen MLP:ssä tarvitaan vähemmän neuroneiden välisiä yhteyksiä, mutta tällöin sovituksessa saattaa ilmetä ylimääräinen paikallinen minimi. MLP on todella tehokas useiden muuttujien funktioiden approksimoinnissa. Perinteiset lineaariset regressio menetelmät kärsivät liian suurien muuttuja määrien käytöstä. Tätä ongelmaa MLP neuroverkossa ei ole. (Du & Swamy 2014, 83–84.)

MPL:n opetusalgoritmit perustuvat virhefunktion *E* minimointiin käyttäen euklidista itseisarvoa (kaava 57). Minimointi tapahtuu löytämällä optimaaliset painoarvot *w* jokaiselle neuronille. (Du & Swamy 2014, 85–89; Osowski, Siwek & Markiewicz 2004.)

$$E(w) = \frac{1}{2}\sum_{k=1}^{n}\|f(x_k, w) - y_k\|^2 \qquad (57)$$

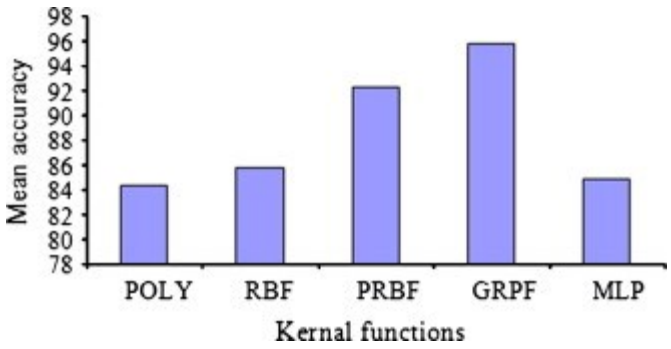
Tehokkain tapa virheen minimointiin on vastavirta gradient-algoritmit, joiden laskennassa käytetään osittaisderivaattoja ja derivoinnin ketjusääntöä. Aktivointifunktiona käytetään yleensä hyperbolista tangenttia. (Du & Swamy 2014, 85–89). Levenberg Marguard-algoritmi on tehokas keskikokoisella verkolla ja suurilla verkolla conjugate gradient (Zanaty 2012). Yleensä kaikissa gradient-algoritmeissa painoarvojen sovittaminen suoritetaan vaihe kerrallaan kaavan 58 mukaisesti.

$$w(k+1) = w(k) + \eta p(k), \tag{58}$$

missä *p(k)* on minimoinnin suunta algoritmin k:lla suorituskerralla ja η on oppimiskerroin. Minimointi suunta arvon *p(k)* laskentatapa vaihtelee algoritmien välillä. Oppimiskerroin η asetetaan manuaalisesti tai käyttämällä adaptiivista tapaa, joka ottaa huomioon virhefunktion todellisen pienentymisen. (Osowski, Siwek & Markiewicz 2004.) Perustilanteessa vastavirta gradient-algoritmia käyttävien neuroverkkojen oppimiskyky on hidasta. Lopullisten tulosten saaminen vaatii suurta määrää painoarvojen päivityskierroksia. Oppimiskertoimen määrittäminen adaptiivisella tavalla nopeuttaa neuroverkkoa. Kuviossa 25 esitetään visuaalisesti virhefunktion tuloksen pienentyminen. (Du & Swamy 2014, 102–108.)

KUVIO 25. Virhefunktion tuloksen pienentyminen, mustat pisteet ovat päivityskertoja. a) pieni oppimiskerroin b) suuri oppimiskerroin c) adaptiivinen oppimiskerroin. (Du & Swamy 2014, 89)

MLP:n toimiessa luokittelussa ja suurella muuttuja määrällä se vaatii useita piilokerroksia, jolloin laskenta-ajat kasvavat (Zanaty 2012; Osowski, Siwek & Markiewicz 2004). Tämän lisäksi Zanatyn (2012) suorittamissa testeissä havaittiin myös, että MLP:n luokittelutarkkuus on heikompi verrattuna SVM:n eri algoritmeihin (kuvio 26). (Zanaty 2012.)

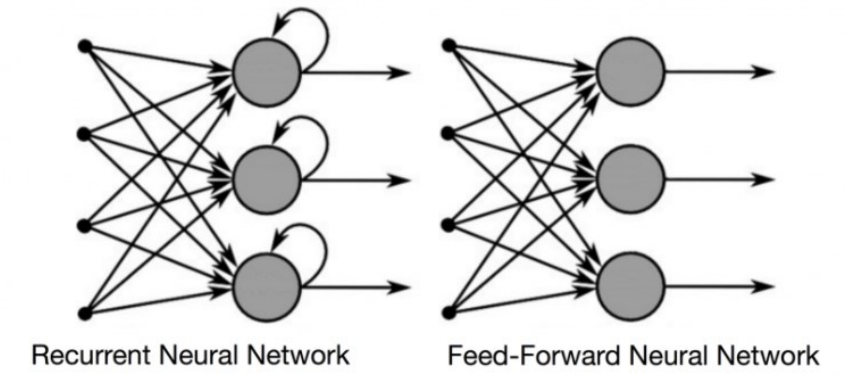


KUVIO 26. SVM algoritmien ja MLP keskimääräinen luokittelutarkkuus (Zanaty 2012)

5.6 RNN (Recurrent neural network)

Ihmisen aivoissa on voimakas takaisinkytkentärakenne. Neuroneiden ulostulot palaavat enemmin tai myöhemmin samaan neuroniin sisääntuloina. RNN-verkoissa hyödynnetään samaa periaatetta. RNN-verkossa on ainakin yksi takaisinkytkentä. Takaisinkytkennän ansiosta RNN-verkon koko voi olla merkittävästi pienempi kuin perinteisillä eteenpäin kytkeytyvillä verkoilla (MLP). RNN-verkon approksimointikyky dynaamisissa järjestelmissä on samalla tasolla kuin eteenpäin kytkeytyvissä verkossa. MLP-verkot ovat puhtaasti staattisia ja eivätkä pysty prosessisoimaan ajan vaikutusta. Takaisinkytkentä mahdollistaa RNN-verkolle tulevien tapahtumien ennustamisen. Takaisinkytkentä toimii niin sanottuna muistina. (Du & Swamy 2014, 337–339.) Yleensä RNN-verkolla on käytössä lyhytaikainen muisti, mutta se voidaan muuttaa pitkäaikaiseksi käyttämällä LSTM (Long Short-Term Memory) (Alfarraj & Alregib 2018; Donges 2021).

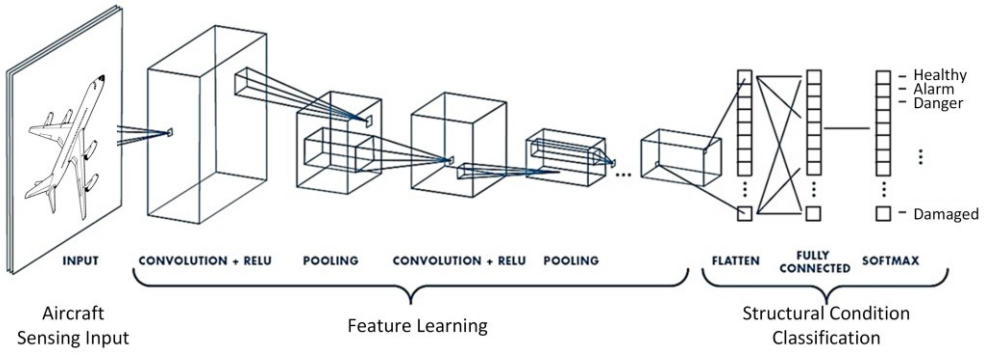
RNN-verkon arkkitehtuuri sallii yhteydet jokaisen neuronin välillä (kuvio 27). Yleisesti tämän tyyppiset verkot kärsivät vakausongelmista opetuksen aikana. Opetusalgoritmit ovat monimutkaisia ja paljon aikaa vieviä. Stabiilimpi ratkaisu muistuttaa MLP-verkkoa, mutta siinä on paikallinen takaisinkytkentä. Paikallinen takaisinkytkentä tarkoittaa sitä, että neuroni saa sisääntulona myös oman ulostulonsa (kuvio 27). (Du & Swamy 2014, 337–339.)



KUVIO 27 RNN ja perinteisen eteenpäin kytkeytyvän neuroverkon periaatekaavio (Donges 2021)

5.7 CNN (Convolutional neural network)

Konvoluutioneuroverkot (*Convolutional neural network* CNN) ovat suunniteltu erityisesti visuaalisten kuvioiden tunnistusta varten. CNN:ssä on minimoitu kuvioiden tunnistukseen tarvittava laskenta. CNN:n topologia on yleensä eteenpäin kytkeytyvä neuroverkko (kuvion 27 oikeapuoli). (Sewak, Karim & Pujari 2018, luku1.) CNN-verkoilla on paljon yhteisiä ominaisuuksia kuin MLP-verkolla. Molemmissa neuronit suorittavat laskutoimituksia ja välittävät tulokset eteenpäin epälineaarisen aktivointifunktion kautta. Yleensä (yksinkertaisessa) CNN-verkossa on kolme kerrosta. Kerrokset on nimetty konvoluutio- (convolution layer), yhdistämis- (pooling layer) ja täysin yhdistetty kerros (fully connected layer). (Sewak, Karim & Pujari 2018, luku2.) Tyypillisessä CNN-mallissa on useita konvoluutio- ja yhdistämiskerroksia peräkkäin ennen viimeistä täysin yhdistettyä kerrosta (kuvio 28) (Dertat 2017).



KUVIO 28. Konvoluutio neuroverkon tyypillinen arkkitehtuuri (Tabian, Fu & Sharif Khodaei 2019)

Teoriassa myös MLP-verkoilla pystytään luokittelemaan kuvia, mutta CNN-verkot on suunniteltu juuri kuvien luokittelua varten. CNN-verkkoihin on sisäänrakennettu ominaisuus, että ne ymmärtävät kuvan pikseleiden välisiä yhteyksiä. Lähellä toisiaan olevat pikselit ovat enemmän riippuvaisia toisistaan kuin kauempana olevat pikselit. CNN-verkko muodostaan kuvista kolmiulotteisen syötematriisin. Matriisin dimensiot ovat leveys, korkeus ja syvyys. Matriisin syvyys on kolme. (Sewak, Karim & Pujari 2018, luku 2.)

Syvyys kolme tulee kuvissa yleisesti käytettävästä värikoodausjärjestelmästä RGB (*red, greed, blue*), jossa jokaisen pikselin väri esitetään kolmen alkuvärin summana. RGB-värijärjestelmässä jokainen värikanava voi saada arvon 0–255. Värien numeraaliset arvot skaalataan lähes poikkeuksetta välillä 0–1. Arvojen skaalaaminen välille 0–1 nopeuttaa laskentaa ja pienentää riskiä, että virhefunktion minimointi jää jumiin paikalliseen minimiin.

Syötekerros sisältää kuvan tiedot matriisimuodossa, jonka jälkeen syöte viedään konvoluutiokerrokseen. Ero täysin yhdistetyn ja konvoluutiokerroksen välillä on se, että konvoluutiokerroksessa neuroneihin kytkeytyy vain lähialueiden syötteet (pikselit). Lisäksi neuroneilla on yhteiset parametrit. Aktivointifunktiona ReLu on hyvin käyttökelpoinen. (Sewak, Karim & Pujari 2018, luku2; Aghdam & Heravi luku 3, 2017.)

5.7.1 Konvoluutiokerros (convolution layer)

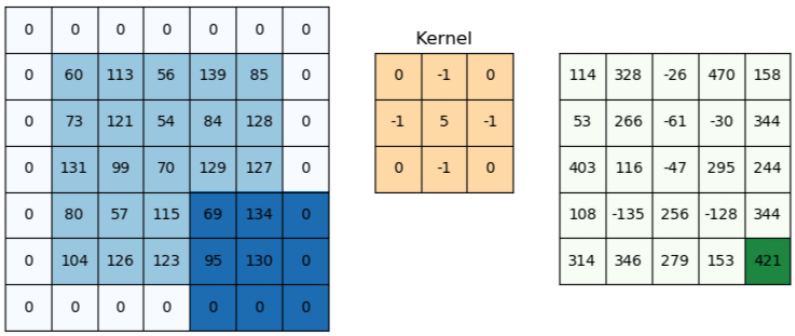
Konvoluutiokerroksen tehtävä on louhia kuvasta oleellisia piirteitä ja tiivistää datamassaa. Konvoluution avulla kuvasta muodostetaan piirrekartta (*feature map*). Piirrekartta muodostetaan syöttämällä kuva CNN-verkoissa käytettävän suodattimen (*filter, kernel*) läpi (kuvio 29). Tyypillinen suodattimen koko on $3 * 3$, $5 * 5$ tai yleisesti $n_k * n_k$ pikseliä. Piirrekarttaan muodostuva arvo on summa suodattimen ja sisääntulo datan tuloista. (Sewak, Karim & Pujari 2018, luku2; Michelucci 2019, luku 3.) Yhden suodattimen läpi syöttämällä data saadaan vain yksi näkemys datan piirteistä. Konvoluutioneuroverkoissa käytetään useita rinnakkaisia suodattimia datan piirteiden oppimiseen. Tyypillisesti suodattimien lukumäärä n_c on jokin kahden potensseista 32 ja 1024 välillä. Jokaisen suodattimen tuloksena

saadaan oma piirrekartta. Suurempi suodattimien määrä tekee mallista tehokkaamman, mutta samalla ylisovituksen riski kasvaa. Yleensä ensimmäisissä konvoluutiokerroksissa suodattimien määrä on pienempi ja määrää kasvatetaan myöhemmissä kerroksissa. (Dertat 2017.)

Konvoluutioneuroverkon ensimmäisellä kerroksella toimivat suodattimet louhivat kuvasta alemman tason ominaisuuksia. Näitä voivat olla muun muassa viivat ja reunat. Konvoluutioprosessia jatketaan, kunnes tuloksena saadaan ylemmän tason ominaisuuksia, kuten kasvot, rakennukset tai eläimet. Tästä syystä konvoluutioneuroverkot ovat usein monikerroksia. (Brownlee 2019a.) Kun useampi konvoluutiokerros kytketään peräkkäin, tulee edellisen tason ulostulosta seuraavan kerroksen syöte. Monikerroksisuus lisää mallin kompleksisuutta ja tarkkuutta, mutta samalla lisääntyy ylisovituksen riski sekä tarvittava laskenta-aika. Malliin tarvittavien kerrosten lukumäärä riippuu ongelman kompleksisuudesta ja oikea kerrosmäärä saadaan usein vain kokeiluiden kautta (*trial-and-error*). (Tabian, Fu & Sharif Khodaei 2019.)

Askellus (stride *s*) voi olla maksimissaan suodattimen koon verran, esimerkiksi $3 * 3$ suodattimen askel voi olla yksi, kaksi tai kolme pikseliä. Suodattimella ja askeleen koolla pystytään vaikuttamaan siihen, paljonko dataa tiivistetään seuraavaa kerrosta varten. (Sewak, Karim & Pujari 2018, luku2; Michelucci 2019, luku 3.)

Värikuvien kanssa työskennellessä on huomioitava, että konvoluutio ja yhdistäminen täytyy tehdä 3d-matriisissa. Tällöin suodatin on muotoa $n_k * n_k * n_{ch}$, missä n_{hc} on värikanavien lukumäärä, RGB-kuvissa $n_{ch} = 3$. Tämä lisää kompleksisuutta malliin. (Michelucci 2019, luku 3–4; Brownlee 2019a.) Kuvien muuttaminen mustavalkoisiksi yksinkertaistaa mallin 2d-matriisiksi ja pienentää datan määrän kolmasosaan.



KUVIO 29. Konvoluution periaate, askel $s = 1, suodatin\ A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$

(Raydan 2021)

Usein ei ole optimaalista, jos konvoluution ulostulomatriisin koko on eri kuin alkuperäisen datan. Tällöin käytetään yhden askellusta $s = 1$ sekä datalle lisätään kehykset (*padding*). Kehyksiin tulevien datasolun arvo on nolla 0 tai lähimmän oikean datasolun arvo. (Michelucci 2019, luku 3.)

5.7.2 Yhdistämiskerros (pooling layer)

Konvoluutiokerroksessa jokaisesta suodattimesta muodostuu oma piirrekartta, joten jokainen suodatin lisää dimension konvoluutiokerroksen ulostuloon. Jokainen suodatin lisää parametrien määrää mallissa. Yhdistämiskerroksen tehtävä on kontrolloida ylisovituksen riskiä pienentämällä esityksen kokoa (spatial size). Samalla vähennetään mallin tarvitsemaa laskentaa. (Sewak, Karim & Pujari 2018, luku 2.)

Yhdistämiskerroksen sisääntulona on yleensä konvoluutiokerroksen ulostulo. Yleisin yhdistämisfunktio on maksimi (*max pooling*). Toinen käytetty yhdistämisfunktio on keskiarvo (*avarage poo*ling). Yhdistäminen tehdään esimerkiksi 2 x 2 kokoiselle joukolle. Myös yhdistämiskerroksessa sisään tuleva data käydään askel askeleelta läpi. (Michelucci 2019, luku 3; Sewak, Karim & Pujari 2018, luku2.) Yhdistämisessä menetetään osa datasta, mutta sillä on myös useita hyötyjä CNN-verkolle. Yhdistäminen vähentää neuroverkon kompleksisuutta, parantaa suorituskykyä ja vähintään ylisovituksen riskiä. (IBM 2020.) Kuten kuviossa 28 on esitetty, yhdistämisen jälkeen siirrytään seuraavaan konvoluutiokerrokseen tai viimeisiin täysin yhdistettyyn kerrokseen.

5.7.3 Täysin yhdistetty kerros (fully connected layer / Dense layer)

Konvoluutio- ja yhdistämiskerroksissa data on järjestetty 2d- tai 3d-matriisissa, mutta täysin yhdistetyt kerrokset vaativat sisään tulevan datan olevan 1d-vektorin muodossa. Matriisimuotoinen data muunnetaan vektorimuotoon ennen täysin yhdistettyä kerrosta. (Dertat 2017.)

Täysin yhdistetty kerros toimii samalla tavalla kuin eteenpäin kytketty neuroverkko. CNN-malli sisältää yleensä muutaman piilokerroksen, joissa aktivointifunktiona on ReLu. Ulostulokerroksen aktivointifunktio riippuu sovelluksessa.
Eteenpäin kytketyillä neuroverkoilla voidaan teoriassa tehdä kuvien luokittelua. Tämä ei ole kuitenkaan käytännöllistä, koska tarvittavan laskennan määrä kasvaa eksponentiaalisesti. Tästä syystä kuvien kanssa käytetään CNN-mallia, jossa kuvista aluksia louhitaan oleellinen informaatio ennen varsinaista luokittelua. Näin piilokerrosten neuroneiden lukumäärä pysyy hallittavissa rajoissa. (Michelucci 2019, luku 3; Sewak, Karim & Pujari 2018, luku 2.).

5.7.4 Opeteltavien parametrien määrä CNN-verkossa

CNN-verkoissa kuvan koko ei vaikuta opeteltavien parametrien määrään. Parametrien määrään vaikuttaa konvoluutiokerroksessa olevien suodattimien n_c lukumäärä sekä suodattimen koko $n_k * n_k * n_{ch}$. Näiden lisäksi jokaiselle suodattimella on oma vakiotermi, joka tulee huomioida laskennassa. Yhdistämiskerroksessa ei ole opeteltavia parametrejä. Konvoluutiokerroksen opeteltavien parametrien määrä on $n_c * n_k * n_k * n_{ch} + n_c$. Esimerkiksi kuvien ollessa mustavalkoisia, suodattiminen lukumäärä ollessa 32 ja ne ovat $5 * 5$ kokoisia. Tällöin opeteltavien parametrien lukumäärä kyseisessä konvoluutiokerroksessa on $32 * 5 * 5 * 1 + 32 = 832$ kappaletta. (Michelucci 2019, luku 3.)

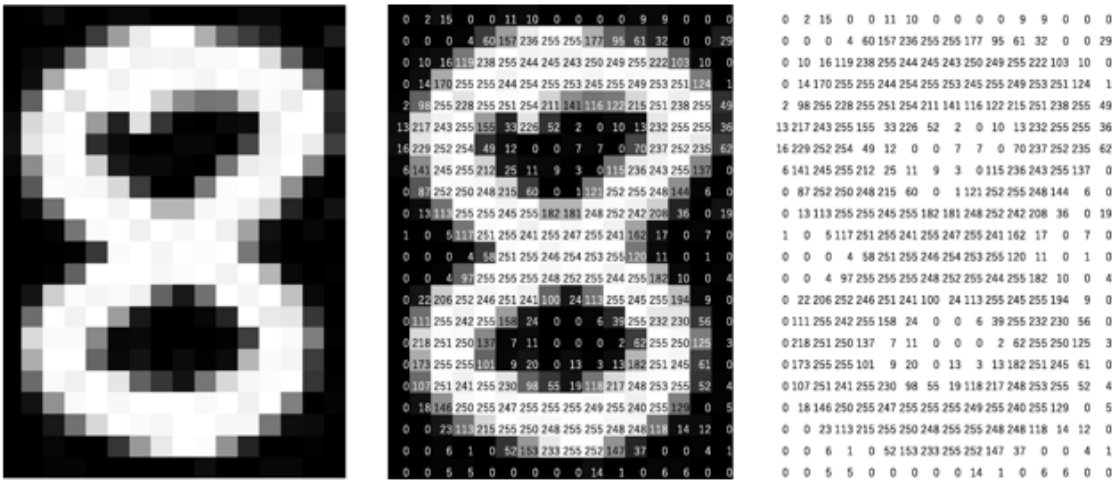
Täysin yhdistetyssä kerroksessa on ratkaistavana samat painoarvot kuin perinteisessä eteenpäin kytketyssä neuroverkossa. Opeteltavien parametrien määrään vaikuttaa neuronien sekä edeltävän kerroksen ulostulojen lukumäärä. (Michelucci 2019, luku 3.)

6 KUVIEN JA OBJEKTIEN TUNNISTAMINEN

6.1 Kuva datalähteenä

Ihmiset näkevät kuvan kaksiulotteisena ruudukkona, jossa jokaisella ruudukolla on väriarvo. Näitä ruudukoita kutsutaan pikseleiksi. Tietokoneet näkevät kuvat samankaltaisella tavalla. Tietokoneella kuva on tallennettu kaksiulotteiseen matriisiin numeroina (kuvio 30). Numerot edustavat pikselin väriarvoja. Pikselit on tallennettu matriisiin järjestyksessä rivi kerrallaan. (Kapur 2017, 7–8; Kadam 2020a; Sewak, Karim & Pujari 2018, luku2.)

Väriarvot koodataan käytettävän värimallin mukaisesti. Yleisin värimalli on RGB, jossa jokaisella pikselillä on kolme värikanavaa (punainen, vihreä ja sininen). Värikanavan arvo voi olla 0-255 välillä, jolloin se pystytään esittämään tietoteknisesti 8 bitin sarjana (8bitti = 1 tavu). Täten yhden pikselin värien esittäminen vaatii 24 bittiä tai 3 tavua. Mustavalkoisissa kuvissa käytössä on vain yksi värikanava, joka voi saada arvon väliltä 0–255 (8bit koodaus).



KUVIO 30. Käsin kirjoitettu numero kahdeksan (8), oikeassa reunassa miten tietokone näkee kuvan. (Kadam 2020a)

Kuvat sisältävät paljon piilotettua informaatiota, jota ihmiset käsittelevät tiedostamatta. Ihmiset pystyvät hahmottamaan kuvassa olevien objektien eroja tai vastaamaan kysymykseen; onko kuvassa auto? Tietokoneelle nämä eivät ole itses-

täänselvyyksiä ja ovat oikeastaan hyvin vaikeita tehtäviä. Kuvankäsittely- ja analysointimenetelmillä kuvasta etsitään algoritmeilla tunnistettavia piirteitä. Tunnistettavia piirteitä voivat olla muun muassa reunat, ääriviivat, objektien sijoittelu, väri tai pikseleiden väliset vuorovaikutukset. (Kapur 2017, 7–8)

Tunnettu kuvista koostuva datajoukko, MNIST, sisältää käsin kirjoitettuja numeroita. Kuvat ovat kooltaan 28x28 pikseliä, jolloin pikseleiden yhteismäärä on 784. Kuvadata on 2d (tai 3d) muotoisessa matriisissa. Jotta sitä voidaan käsitellä perinteisillä menetelmillä (neuroverkko tai klassiset koneoppimisalgoritmit), tulee se muuntaa aluksi vektorimuotoon. Kuvan muuntamin vektorimuotoon kadottaa datasta pikseleiden välisen vuorovaikutuksen (Raydan 2021; Kadam 2020b).

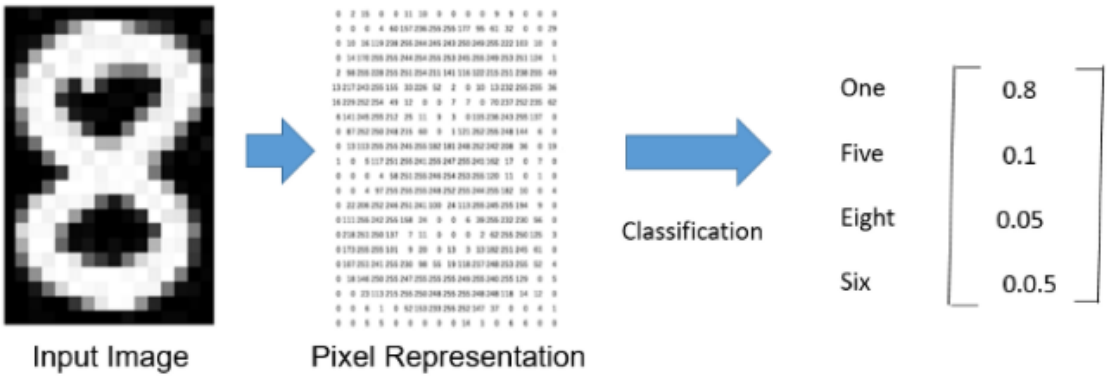
Perinteiset koneoppimisalgoritmit pystyvät käsittelemään kuvadataa rajoitetusti, mutta käytännössä neuroverkot ovat hallitseva menetelmää. Usean kerroksen neuroverkot, joissa kaikki sisääntulomuuttujat yhdistetään jokaiseen neuroniin, muodostuvat liian kompleksisiksi ja aiheuttavat mallin ylioppimista. Ylioppimisen torjumiseksi tulee opetusdataa lisätä lähes eksponentiaalisesti. MNIST datajoukon kuvia pystytään käsittelemään tavallisella neuroverkolla, mutta isompien kuvien kanssa neuroverkko alkaa tukehtua. (Raydan 2021; Ying 2019.)

Kuvista etsitään aluksi piirteitä ja tiivistetään oleellinen informaatio pienemmäksi. Piirteiden louhinta (*feature extraction*) tehdään lähes poikkeuksetta konvoluution avulla ja tiivistäminen tapahtuu yhdistämällä arvoja maksimi- tai keskiarvoperiaatteella (*maxPooling, avgPooling*).

Tapauskohtaisesti kuvan kokoa voidaan pienentää jo ennen, kun se syötetään neuroverkkoon. Voidaan ajatella, että tietokoneen on tunnistettava kuvasta oleellinen informaatio, jos ihminen kykenee siihen. Tämän lisäksi kuva voidaan muuntaa mustavalkoiseksi, jos kuvassa olevat värit eivät ole oleellisia.

6.2 Kuvien luokittelu

Opetusdatan koostuessa kuvista on lähes aina kyseessä luokitteluongelma. Luokittelulla pyritään vastamaan kysymykseen: mitä kuvassa on? Sallitut vastaukset on määritelty mallin luonnin yhteydessä. MNIST datajoukon tapauksessa sallittuja vastauksia on 10 kappaletta. Tämä tarkoittaa sitä, että mallin luokkien lukumäärä on 10. Mallin ulostulona saadaan jokaisen luokan todennäköisyys (kuvio 31).



KUVIO 31. Kuvanluokittelun periaate (Kadam 2020a)

Jotta kuva voidaan luokitella oikein, tulee mallin pystyä ymmärtämään, mitä ainutlaatuisia ominaisuuksia (*features*) luokassa on verrattuina muihin luokkiin. Luokittelu tehdään tunnistamalla kuvasta ominaisuuksia. Mallin opetusvaiheessa kuvasta löytyvät ominaisuudet yhdistetään luokkiin. Tunnistamalla riittävästi ominaisuuksia kuvasta malli pystyy melko suurella varmuudella ilmoittamaan kuvan kuuluvan tiettyyn luokkaan. Kuvion 31 tapauksessa ominaisuudet voivat olla esimerkiksi kaareva muoto tai musta kohta kaarevien muotojen sisäpuolella. (Kadam 2020a.)

Isojen kuvien luokittelu vaatii monikerroksisien neuroverkkojen käyttämissä. Ohjelmointikirjastot sisältävät neuroverkojen peruskerrokset valmiiksi paketoituina funktioina, jolloin niitä voidaan yhdistää melko vapaasti tehokkaan verkon aikaansaamiseksi. Toisaalta ohjelmointikirjastot sisältävät usein myös valmiiksi paketoituja kokonaisia neuroverkkoarkkitehtuureja. Valmiit neuroverkot sisältävät usein mahdollisuuden käyttää esiopetettuja painoja.

Kuvan luokittelussa koko kuvalle annetaan yksi luokka ja moniluokkaluokittelussa (*multi-label*) kuvalle annetaan useita toisistaan osittain tai täysin riippumatonta luokkaa. Kuvat sisältävät usein paljon ylimääräistä informaatiota luokitteluongelman näkökulmasta, jolloin koko kuvan luokittelu kerralla on haastavaa. Luokitteluongelman näkökulmasta kuvassa saattaa olla relevantteja pikseleitä vain murto-osa. Tällöin ei ole järkevää syöttää koko kuvaa mallin opetukseen vaan käyttää objektintunnistusta.

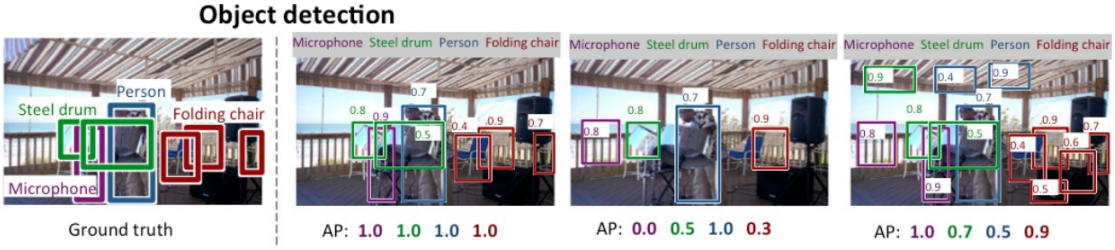
6.3 Objektintunnistus

Objektin tunnistaminen kuvasta on pohjimmiltaan luokitteluongelma, mutta luokkaa ei kohdisteta koko kuvaan. Luokiteltava objekti tulee ensiksi paikantaa kuvasta, jonka jälkeen sille voidaan antaa luokka. Objektintunnistusalgoritmit koostuvat kahdesta osasta; paikannuksesta ja luokittelusta. Opetusdata sisältää kuvien lisäksi tiedon kuvasta löytyvistä objekteista. Objekteista tulee tietää sijainti (x-y koordinaatit) sekä objektin luokan. (Michelucci 2019, luku 6; Hulstaert 2018; Russakovsky ym. 2014.)

Algoritmit ovat toimintaperiaatteen perustella jaettavissa kahteen pääryhmään. Ryhmät ovat yksi- ja monivaiheiset menetelmät. Yksivaiheiset menetelmät priorisoivat tunnistuksen nopeutta osittain tarkkuuden kustannuksella. Tunnettuja yksivaiheisia algoritmeja on YOLO (You Only Look Once) ja SSD (Single Shot Detection). Monivaiheiset menetelmät painottavat tarkkuutta ja ovat toiminnaltaan yksinkertaisempia. (Jiao ym. 2019.) Vuonna 2014 julkaistussa tutkimuksessa esiteltiin R-CNN menetelmä objektien tunnistamiseen (Girshick ym. 2014). Tänään R-CNN ja siitä jalostetut algoritmit ovat vallitseva menetelmä objektintunnistuksen toteuttamisessa. Ensimmäiset R-CNN pohjaiset algoritmit olivat verrattain hitaita ja vaativat paljon laskentaa. Uusimmat versiot algoritmista ovat laskennaltaan huomattavasti kevyempiä ja ovat nopeuden suhteen jo verrattavissa yksivaiheisiin algoritmeihin. (Jiao ym. 2019.)

Monivaiheiset algoritmit toimivat siten, että kuvasta muodostetaan pienempiä kuvia (*region proposals*), jotka syötetään luokittelualgoritmiin. Pienempien kuvien muodostaminen tehdään hyödyntämällä neuroverkon kuvasta louhimia ominai-

suuksia. (Hulstaert 2018.) Algoritmin ulostulona saadaan objektin luokka ja koordinaatit kuvassa. Usein tulos visualisoidaan piirtämällä kuvaan suorakulmainen laatikko löydetyn objektin ympärille (kuva 3).



KUVA 3. Objektin tunnistus: vasemmassa reunassa opetusdata, oikealla olevissa kuvissa mallien tuloksia (Russakovsky ym. 2014.)

Opetusdatassa olevia objekteja kutsutaan pohjatotuudeksi (*ground truth*). Objektintunnistusmallin tarkkuutta mitataan vertaamalla löydettyjä objekteja pohjatotuuteen. Havainto todetaan oikeaksi (TP), jos se täyttää seuraavat ehdot

- havainnon luottamus (*confidence score*) on yli kynnysarvon (≥ 0.5).
- leikkausalue IoU (*Intersect Over Union*) on yli kynnysarvon (tyypillisesti $0.25 - 0.5$).
- havainnon pohjatotuuden luokka on sama kuin luokittimen tulos.

Havainnon luottamus kuvaa todennäköisyyttä, jolla löydetty havainto sisältää mallin mukaisen luokan. Luokittelu tehdään pääsääntöisesti luokittimella. Leikkausalue lasketaan kaavalla x, jossa B_p on havainnon rajausalue ja pohjatotuuden alue on B_{gt}. (Michelucci 2019, luku 6; Zeng 2018; Yohanandan 2020)

$$IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{59}$$

Havainto luokitellaan vääräksi positiiviseksi (FP), jos se ei täytä kahta viimeisintä ehtoa. Luottamuksen ollessa alle kynnysarvon havainto luokitellaan vääräksi negatiiviseksi (FN). Oikeita negatiivisia (TN) havaintoja ei tarvita mittareiden laskennassa. Tarkkuus (*precision*) ja saanti (*recall*) lasketaan kaavoilla 8 ja 9. Säätämällä luottamuksen kynnysarvoa saadaan tarkkuudelle ja saannille eri tulokset, jotka voidaan visualisoida PR-käyrällä (*precision-recall curve).* PR-käyrää voidaan käyttää mallin suorituskyvyn arviointiin, mutta mallien väliseen vertailuun

puhtaat numeraaliset arvot ovat parempia, kuten keskimääräinen tarkkuus (AP). (Zeng 2018; Yohanandan 2020.)

PR-käyrään perustava keskimääräinen tarkkuus on kaikkien saanti tasojen tarkkuuden keskiarvo. Ennen AP:n laskentaa lasketaan interpoloitu tarkkuus p_{interp} (kaava xx) tietyllä saannin arvolla r, joka on suurin löytyvä tarkkuus millä tahansa saannin arvolla $\acute{r} \geq r$. Tällä vähennetään PR-käyrässä esiintyvän huojunnan vaikutusta lopputulokseen. (Zeng 2018; Yohanandan 2020; Manning, Raghaven & Schütze 2008, 158–160.)

$$p_{interp}(r) = \max_{\acute{r} \geq r} p(\acute{r}) \tag{60}$$

Kaavan xx tulosten laskemiselle on kaksi vakiintunutta menetelmää. Perinteisessä menetelmässä interpoloitu tarkkuus lasketaan yhdelletoista, tasaisin välein olevalle, saannin arvolle (0.0, 0.1, 0,2, … 1.0). Uuden menetelmän mukaan interpoloidut arvot lasketaan kaikilla uniikeilla saanti arvoilla. Uuden menetelmän on esitetty parantavan tarkkuutta ja mittaavaan mallien eroja paremmin, kun AP on alhainen. AP voidaan määritellä alueeksi, joka jää interpoloidun käyrän alle PR-käyrässä ja voidaan laskea kaavalla 61. (Zeng 2018; Yohanandan 2020; Manning, Raghaven & Schütze 2008, 158–160.)

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i)\, p_{interp}(r_{i+1}), \tag{61}$$

missä $r_{1,} r_2, \dots, r_{n,}$ on saanti arvo laskevassa järjestyksessä, jossa tarkkuus oli interpoloitu. Kaavalla 61 lasketaan vain yhden luokan mallin AP. Useita luokkia sisältävän objektin tunnistus mallin tärkein mittari on mAP (*Mean average precision*), joka lasketaan kaavalla 62, jossa K on luokkien lukumäärä. (Zeng 2018; Yohanandan 2020; Manning, Raghaven & Schütze 2008, 158-160.)

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{62}$$

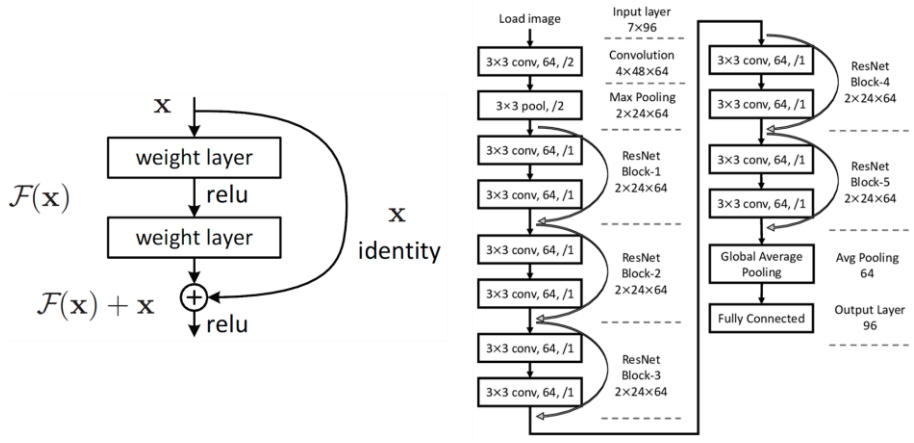
Usein mAP arvon lisäksi ilmoitetaan IoU:lle käytetty kynnysarvo, esimerkiksi $mAP@0.25$ tarkoittaa mAP arvoa IoU:n ollessa 0.25.

6.4 Kuvien luokittelu ja objektin tunnistus Azuressa

Microsoft Azure on pilvipalvelualusta, joka tarjoaa erilaisia palveluita pilvessä. Palveluihin sisältyy infrastruktuurin vuokraamista (IaaS), ohjelmakehitysalustoja (PaaS) sekä valmiita ohjelmia käytettäväksi (SaaS). Azuressa on useita tekoälyyn liittyviä palveluita. Yksi näistä on Azure Machine Learning (ML) Studio, joka toimii alustana koneoppimissovellusten kehittämiseen. Alustalla voidaan toteuttaa kaikki koneoppimissovelluksen tarvitsemat vaiheet opetusdatan noutamisesta mallin julkaisemiseen ja käyttöön. Sovellusten tekeminen onnistuu python tai R-kielillä sekä graafisen ohjelmointityökalun kautta.

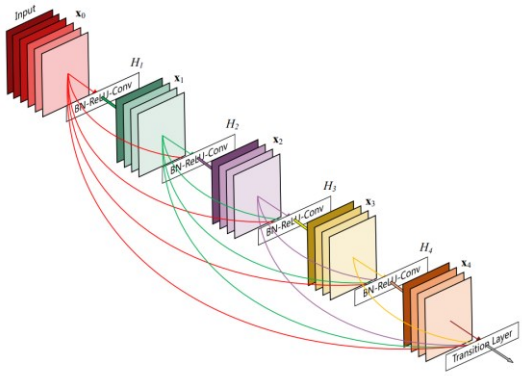
Useat tuotantojärjestelmiin liitettävät koneoppimismallit hakevat opetusdatansa tietokantokannoista ja käytössä olevia malleja opetetaan jatkuvasti tai määräajoin uudestaan. Näin mallit tulevat ajan myötä paremmiksi. Mallin opetus ja hallinnointi pilvessä mahdollistaan tehokkaan tavan suorittaa jatkuvaa uudelleenopetusta. Ulkopuoliset hyötyohjelmat voivat käyttää mallia valmiiden rajapintojen kautta.

Kuvien luokitteluun Azure ML Studio tarjoaa tehokkaat ResNet ja DenseNet neuroverkkoarkkitehtuurit käytettäväksi graafisen käyttöliittymän kautta. Kyseiset arkkitehtuurit edustavat *state-of-art* neuroverkkoja kuvien luokitteluun. ResNet-verkon keskeinen ideana on, että aktivointifunktiolle syötetään käsitellyn datan $F(x)$ lisäksi kerroksen syötedata x (kuvio 32).



KUVIO 32. Vasemmalla ResNet-mallin rakennusyksikkö, oikealla ResNet-12 arkkitehtuuri (He ym. 2015)

Katoava gradientti aiheuttaa syvissä neuroverkoissa tarkkuuden heikentymistä. DenseNet-arkkitehtuuri on kehitetty lieventämään katoavaa gradienttia ja täten parantamaan verkon tarkkuutta. DenseNet:ssä jokainen lohko saa sisääntulona edellisten kerrosten ulostulodatan sekä alkuperäisen sisääntulodatan (kuvio 33). (Huang ym. 2017)



KUVIO 33. DenseNet-arkkitehtuurin periaate (Huang ym. 2017)

Microset tarjoaa Azuressa kuvien luokittelu- tai objektintunnistusohjelman palveluna nimeltä Custom Vision. Ohjelma toimii SaaS-periaatteella. Käyttäjälle ohjelman käyttö on tehty helpoksi, mutta myös rajatuksi. Ohjelmassa pystyy rakentamaan ja julkaisemaan malleja, mutta mallien hyperparametrit ja algoritmit ovat täysin piilotettu käyttäjältä. Mallien rakentaminen Custom Visionilla edellyttää tasapainoista opetusdataa. Negatiivisten kuvien lisäämisellä opetusdataan voidaan parantaa mallia. Negatiivisella kuvalla tarkoitetaan kuvaa, jossa ei ole mallin sisältämiä luokkia. Negatiiviset kuvat ovat samankaltaisia kuin luokitellut kuvat, mutta niille kerrotaan, että tämä ei ole mikään mallin luokista. Esimerkkinä kissoja ja koiria tunnistavaan malliin voi olla hyvä lisätä kuvia myös muista karvaisista eläimistä, mutta niitä ei merkitä kuuluvaksi mihinkään luokkaan. Toinen vaihtoehto on luoda luokka muut, johon sisällytetään kaikki negatiiviset kuvat.

Custom Visionilla pystyy rakentamaan kuvienluokittelumallin verrattain vähäisellä kuvamäärällä. Ohjelma on optimoitu tunnistamaan kuvista merkittäviä eroavaisuuksia. Microsoftin mukaan 50 kuvaa luokkaa kohti riittää mallin prototyypin luomiseen. Optimoinnista johtuen ohjelma ei ole parhaimmillaan pienten erojen havaitsemiseen kuvista. (Microsoft n.d.)

7 CASE-TUTKIMUS

7.1 FXT ohjelmakirjasto, periaatteet ja parametrisointi

FX-Editor-ohjelmistoon on rakennettu mahdollisuus luoda parametroitavia mallipohjia. Näistä käytetään myös nimitystä template. Käyttö tapahtuu Template Managerin kautta, joka on graafinen käyttöliittymä mallipohjan valintaa ja parametrointia varten. Parametreilla valitaan toimintoja käyttöön tai pois, joten parametrointi on binääristä. Parametrien toiminnallisuus on linkitetty ohjelmakoodin ja grafiikkaan ryhmätunnuksen (groupID) kautta. Mallipohjan grafiikkakuviin on määritelty objektit kuulumaan useiden ryhmätunnusten alle. Yksittäinen objekti voi kuulua kerrallaan vain yhteen ryhmään. Asettamalla ryhmätunnuksen arvo nollaksi objektit poistuvat grafiikalta.

Ryhmätunnus (groupID) voidaan määritellä automaattisesti ryhmän luonnin yhdessä juoksevan numeron periaatteella tai numero voidaan antaa käsin. Ryhmätunnukselle voidaan antaa myös nimi, jonka tarkoitus on kuvata ryhmää selkokielisesti. Esimerkiksi ilmanvaihtokoneen tulopuhaltimeen liittyvät objektit voidaan ryhmitellä kuuluvaa ryhmään, jonka nimi on TULOPUHALLIN. Ryhmän nimen merkistössä sallitaan vain isot kirjaimet ja numerot sekä muutama erikoismerkki.

Ryhmätunnusnumero on grafiikkakuvakohtainen. Osa mallipohjista sisältää kaksi tai useamman grafiikkakuvan. Tällöin sama ryhmätunnuksen numero saattaa esiintyä useammassa kuvassa, mutta tarkoittaa eri asiaa. Linkitys samaa merkitseväksi tehdään ryhmätunnuksen nimen kautta. Ryhmätunnusten lukumäärä vaihtelee mallipohjista riippuen noin 10–40 välillä.

Generoitava ohjelmakoodi muodostetaan yksinkertaisen IF-ELSE valintojen kautta. IF-lauseen ehdot rakennetaan käyttämään samoja ryhmätunnuksia kuin grafiikkakuvissa ja ehtoja voi olla useita IF-lausetta kohti. Ehtoja voidaan linkittää lauseeseen Boolen algebran operaattoreilla JA (AND), TAI (OR) sekä EI (NOT). ELSEIF-tyylistä valintaa skriptissä ei ole mahdollista rakentaa. Tämä voidaan kiertää tekemällä monta toisistaan riippumatonta IF-lausetta, joissa kuitenkin vain

yksi voi toteutua kerralla. IF-lauseen ehtojen toteutuessa tulostetaan lauseen sisälle määritelty ohjelmakoodi. Jos lauseeseen on määritelty ELSE-ehto, tulostetaan sen sisältö, jos IF-lause ei toteudu.

Tällä hetkellä Template-ohjelmakirjaston jakelu tapahtuu jaetun verkkokansion kautta. Käyttäjän vastuulla on ohjelmakirjaston kopiointi ja tarvittaessa uuden version lataaminen omalle tietokoneelleen. Jakelupaketti sisältää useita mallipohjatiedostoja, mutta jakelu voisi tapahtua myös yksittäisten mallipohjatiedostojen kautta. Ohjelmakirjasto(t) ovat lisäominaisuus FX-editoriin ja niiden kehitys ei ole suoraan riippuvainen pääohjelman kehitystyöstä. Tämä mahdollistaan sen, että käyttäjä voi yhdistää useita ohjelmakirjastoja omalla tietokoneella tai tehdä uusia mallipohjia.

Mallipohjien käyttö tapahtuu graafisen käyttöliittymän kautta manuaalisesti. Käyttöliittymän kautta valitaan oikea mallipohja ja tehdään siihen tarvittavat parametroinnit. Käytännössä tällä tarkoitetaan ryhmien valintaa luettelosta. Lisäksi käyttöliittymän kautta voidaan vaikuttaa järjestelmän positiointiin, eli laitetunnuksiin. Positiointimuutokset ovat rajattu tämän opinnäytetyön ulkopuolelle.

Tässä case-tutkimuksessa esitetään menetelmä, jolla voidaan suorittaa mallipohjan valinta ja parametrointi koneoppimisen keinoin. Ohjelmakehityksen tarkkoja yksityiskohtia ei esitellä. Koneoppimisen osuus on kirjallisen raportin pääosassa.

7.2 Toteutuksen yleiskuvaus

Pääohjelmaan rakennetaan mahdollisuus avata pdf-tiedosto. Pdf-tiedostosta otetaan kuva leikkaustyökalu-tyylisellä ominaisuudella. Kuvan tulee olla säätökaavion PI-kaaviosta. Pääohjelma lähettää kuvan koneoppimisympäristöön, joka on pilvipalvelussa ja jää odottamaan vastausta tai mahdollista aikakatkaisua.

Säätökaavioiden PI-kaaviot muodostavat mallin opetukseen käytettävän opetusaineiston. Käyttämällä pelkästään PI-kaavioita yksinkertaistetaan toteutettavaa sovellusta. Samalla tehdään oletus, että pelkän PI-kaavioin perusteella voidaan toteuttaa järkevä ohjelma. Toimintaselostuksen tulkitseminen koneoppimisen

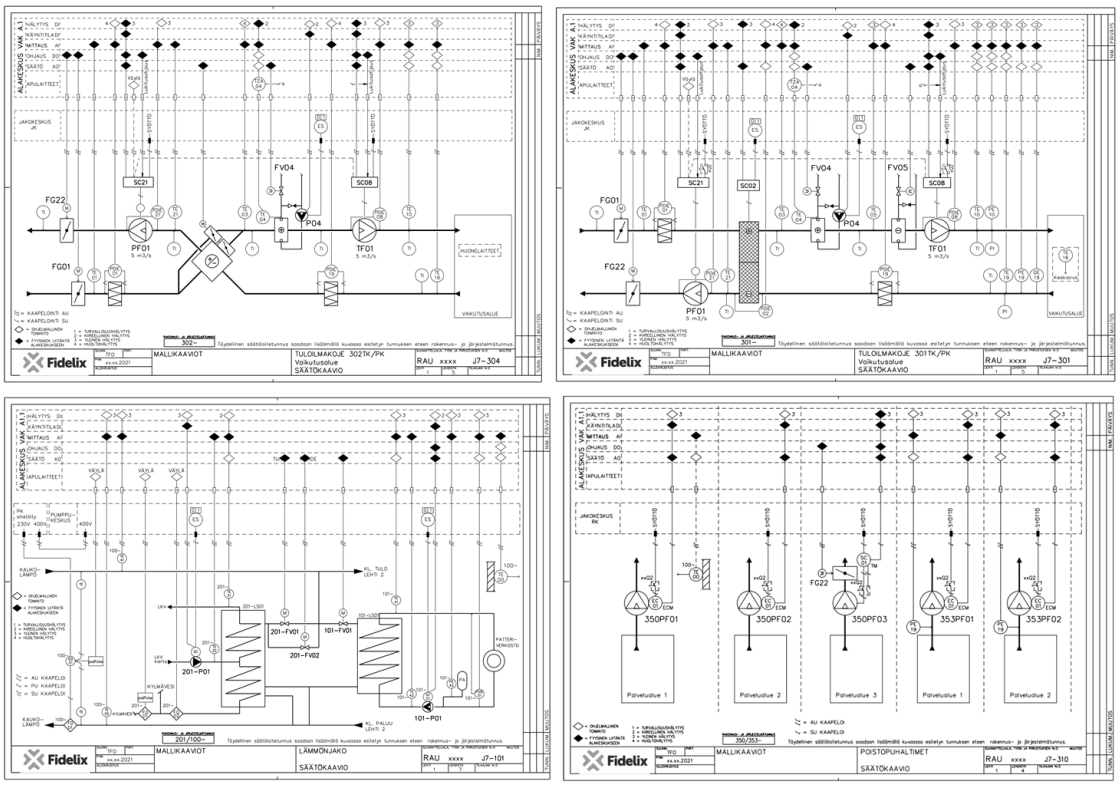
keinoin on haastavaa. Toimintaselostukset ovat usein tulkinnanvaraisia tai vähintään pitää osata yhdistää selostuksen osa koskemaan jotain tiettyä osaa PI-kaaviosta ja linkittää nämä ohjelmakoodiin.

Koneoppimissovellus jaetaan kaksivaiheiseksi. Ensimmäisessä vaiheessa PI-kaavio luokitellaan kuulumaan johonkin mallikirjaston mallipohjaan. Toisessa vaiheessa haetaan mallipohjan mukaisille parametreille (ryhmille) arvot. Jokaiselle mallipohjalle luodaan oma moniluokkainen koneoppimismalli. Koneoppimissovelluksesta saadaan ulostulona kutsukoodi, joka palautetaan pääohjelmaan. Kutsukoodin rakenne on $[templateName, (gID_1, gID_2, \dots, gID_n)]$, missä $templateName$ mallipohjan nimi (merkkijono), $gID_x \in \{0,1\}$ mallipohjan x:nnen ryhmän arvo ja *n* mallipohjan ryhmien lukumäärä kyseissä mallipohjassa. Koneoppimissovellus ei tiedä ryhmätunnusten todellisia nimiä, joten ryhmätunnusten järjestys on sovittava, esimerkiksi aakkosjärjestys. Rajapinta pääohjelman ja koneoppimisympäristön välillä täytyy olla joustava ja skaalautuva. Koneoppimisympäristön tuloksesta (merkkijonosta) generoidaan XML-tiedosto, johon määritellään ryhmätunnukset sekä niiden arvot. XML-tiedosto sisältää vain ne ryhmätunnukset, jotka koneoppimissovellus on laskenut. Muut ryhmätunnukset pääohjelma tulkitsee perusarvoilla.

Pääohjelma tekee mallipohjan ja ryhmien valinnat XML-tiedoston mukaisesti ja jää odottaa käyttäjältä saatavaa vahvistusta. Käyttäjä voi muokata valintoja tai vahvistaa tuloksen. Muokattu ja vahvistettu tulos palautetaan koneoppimisympäristöön mallien uudelleenopetusta varten.

7.3 Opetusdata

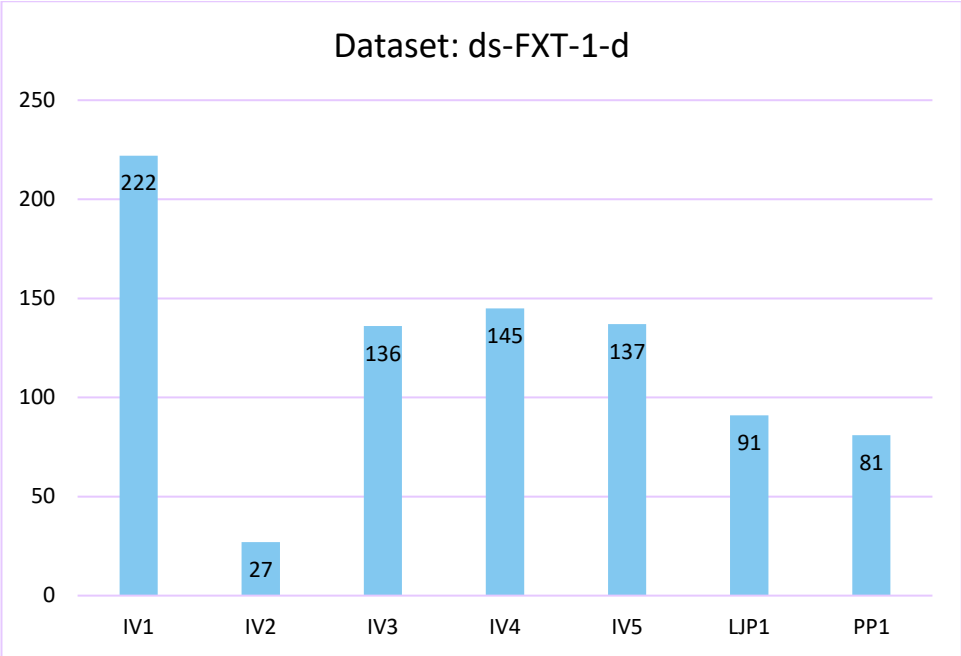
Case-tutkimuksen opetusdatana on käytetty säätökaavioita. Säätökaaviot ovat muunnettu kuvatiedostoiksi. Pdf-tiedostot sisältävät useita sivuja, josta useat ovat tämän tutkimuksen kannalta ylimääräisiä. Ylimääräiset kuvat siivottiin opetusdatakansiosta ennen eteenpäin siirtymistä. Tämän tutkimuksen kannalta olennaisin sivu säätökaavioista oli PI-kaavion ensimmäinen sivu. Kuviossa 34 on esitetty neljä opetukseen käytettyä PI-kaaviota. Ylimmät kuvat ovat ilmanvaihtokoneita ja alemmissa kuvissa on lämmönjakopaketti ja poistopuhaltimia.



KUVIO 34. Opetukseen käytettyjä PI-kaavioita. Kuvion kaaviot lainattu Fidelix Template-kirjaston dokumentaatiosta (Fidelix 2021).

Tutkimukseen mukaan otetut säätökaaviot on saatu yrityksen verkkolevyiltä löytyvistä projektikansioista. Projektikansiot ovat käyttäjäoikeuksilla suojattuja ja niiden kansiorakenne vaihtelee. Säätökaavion kokoaminen tehtiin manuaalisesti. Kuvatiedostoiksi muuntaminen tehtiin python-skriptin avulla. Skripti muuntaa kaikki määritellystä löytyvät pdf-tiedostot jpg-kuviksi siten, että jokaisesta sivusta tulee oma kuvatiedosto.

Tutkimukseen mukaan otettujen säätökaavioiden lukumäärä oli noin 2 000, joista muodostui noin 8000 kuvan joukko. Tästä kuvajoukosta karsittiin pois säätökaavioiden kansilehdet, laiteluettelot ja toimintaselostukset. Kuvatiedostot jaettiin mallikirjaston mukaisiin kansioihin. Opetusdata on rajattu viiteen ilmastointikoneeseen, lämmönjakopakettiin ja poistopuhaltimiin, joten kansioiden lukumäärä on seitsemän. Kukin kansio edustaa omaa mallia mallikirjastosta. Kuvat, joita ei pystytty luokittelemaan tämän jaon mukaisesti, siirrettiin omaan kansioon, jota ei käytetty koneoppimismallin opetukseen. Luokiteltujen kuvien jakauma kansioittain on esitetty kuviossa 35.



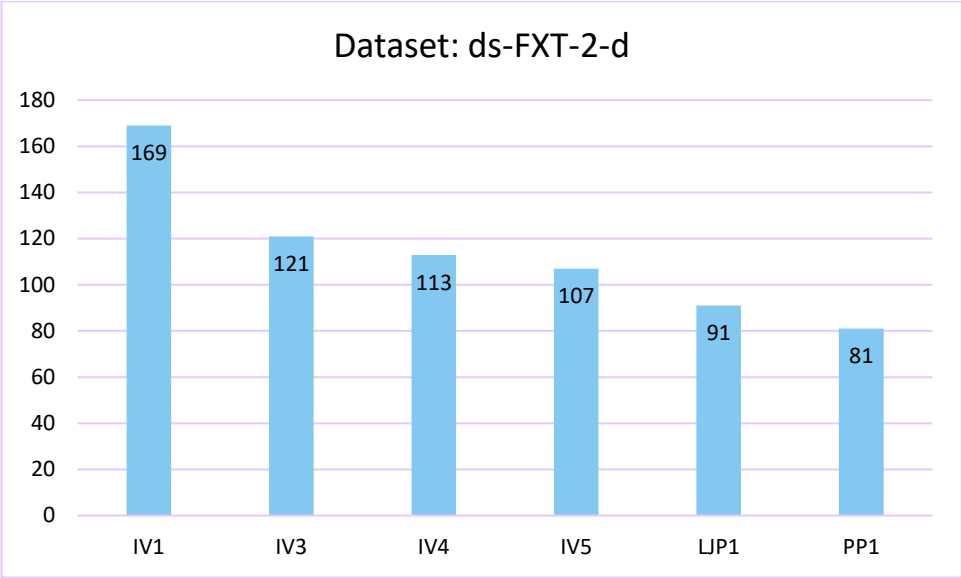
KUVIO 35. Ensimmäisen datajoukon tiedostojen jakauma

Taulukossa 3 on esitetty luokkien selitteet ja viittaukset mallipohjakirjastoon. Huomion arvoista on se, että IV5, LJP1 ja PP1 luokkia vastaa useampi mallipohja. Nämä otettiin mukaan case-tutkimukseen, koska haluttiin selvittää miten mallit suoriutuvat erilaisten kaavioiden kanssa. Tuotantojärjestelmää varten kyseiset luokat tulee hajottaa mallikirjaston mukaisiin luokkiin.

TAULUKKO 3. Luokkien selitteet ja mallipohjan nimi

Luokka	**Selite**	**Mallipohja**
IV1	Ilmastointikone, Kiekko LTO	FX_Template1_IVkoneKiekko
IV2	Ilmastointikone, Kiertoilmapelti	FX_Template1_IVkoneKiertoilma
IV3	Ilmastointikone, Kuutio LTO	FX_Template1_IVkoneKuutio
IV4	Ilmastointikone, Neste LTO	FX_Template1_IVkoneNeste
IV5	Ilmastointikone, "pakettikone"	useita
LJP1	Lämmönjakopaketti	useita
PP1	Poistopuhallin	useita

Tiedostojen jakauma osoittautui haastavaksi mallinopetuksessa. IV2-kansion tiedostojen määrä oli merkittävästi pienempi kuin muiden kansioiden ja tämä teki opetusdatasta epätasapainoisen. IV2-kansio päätettiin poistaa ja muista kansioista poistettiin identtiset tiedostot. Näiden toimenpiteiden jälkeinen tiedostojakauma on esitetty kuviosta 36. Opetusdatan luokkien lukumäärä on kuusi.



KUVIO 36. Karsitun ja tasapainotetun datajoukon tiedostojakauma

Kuvatiedostojen koko vaihteli alkuperäisen pdf-tiedoston asetusten mukaisesti. Pääosa kuvista on muodostettu pdf-sivusta, jotka ovat A4-kokoisia. Pääosa kuvista oli 2921 x 2066 pikselin kokoisia.

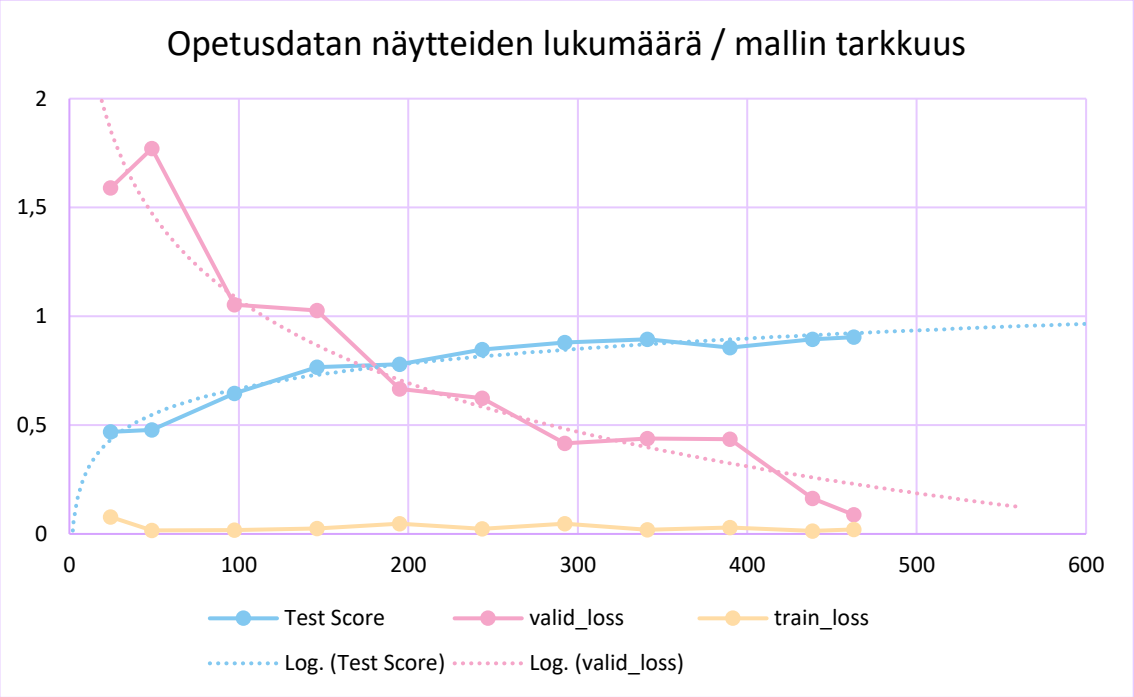
7.4 Mallipohjan valitseminen koneoppimisen avulla

Oikean mallipohjan valitseminen on absoluuttisen tärkeä vaihe. Väärin valittua mallipohjaa ei pystytä muokkaamaan riittävästi, jotta se taipuisi oikeaan järjestelmään. Oikein valitusta mallipohjasta pystytään muokkaamaan soveltuva sovitus, vaikka ryhmävalinnat olisi alkuvaiheessa väärin valittu. Proof-of-Conceptin toteamiseksi tarkkuudelle asetettiin vähimmäistavoite > 90 %. Tuotanto-olosuhteissa tarkkuuden tulee olla > 95 %. Mallipohjan valitsemista tutkittiin kahdella eri metodilla. Kuvanluokittelussa koneoppimismalli antaa tuloksena suoraan oikean mallipohjan nimen (luokan). Objektin tunnistuksessa kuvasta tunnistetaan avainobjekteja, joiden perusteella voidaan päätellä mallipohja. Päättelyä varten tulee luoda ohjelmalogiikka. Lähtöoletuksena kuvantunnistuksessa tarvitaan vähemmän ohjelmointityötä, mutta opetusdataa tarvitaan enemmän. Objektintunnistuksessa opetusdataan tarvitaan lukumääräisesti vähemmän, mutta datan merkitsemiseen tarvitaan enemmän työtä. Lisäksi mallipohjan päättelyalgoritmi tulee rakentaa.

7.4.1 Kuvien luokittelu

PI-kaavioiden luokittelu tehtiin Microsoft Azure Machine Learning Studio:ssa käyttämällä graafista työkalua nimeltä Designer. Designerissä kuvien luokittelua voidaan tehdä DenseNet tai ResNet algoritmeja käyttämällä. Algoritmin valinnan lisäksi keskeisiä asetuksia ovat kuvan esikäsittelyyn liittyvät hyperparametrit sekä mallin opetukseen liittyvät *epoch* ja *batch*-koko. Kaikki opetukseen käytetyt mallit on alustettu ImageNet mukaisilla painoarvoilla (Microsoft n.d.).

Opetusdatan kuvat sisältävät paljon informaatiota ja kuvien lukumäärä on rajattu. Aluksi tutkittiin opetusdatan riittävyyttä suhteessa mallin tarkkuuteen. Kuvion 37 x-akselilla on opetusdatan näytteiden lukumäärä ja y-akselilla virhefunktion tulos sekä tarkkuus. Opetukseen käytetyn mallin keskeiset asetukset on esitetty taulukossa 4.



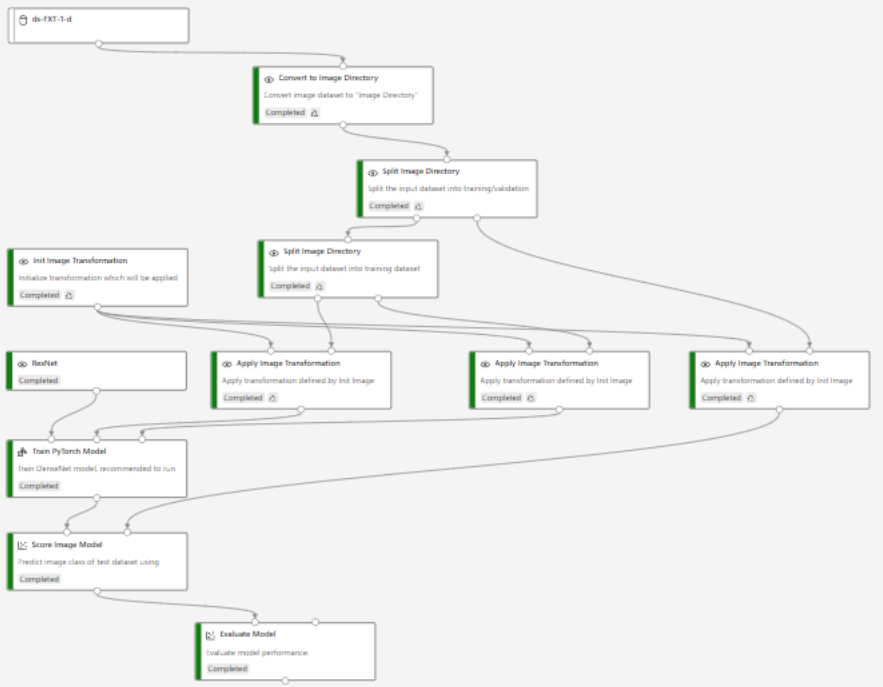
KUVIO 37. Opetusdata näytteet lukumäärä ja mallin tarkkuus

TAULUKKO 4. Koneoppimismallin keskeiset vakioarvoista muutetut parametrit

Init Image Transformation		Model settings	
Resize	600	Model	DenseNet161
Center crop	580	Pretrained	True
Grayscale	True	Epochs	15
Random rotation degrees	10	Batch size	6
Random horizontal flip	True		

Kuviosta 37 voidaan tulkita, että näytteiden lukumäärän ollessa yli 400 validoinnin virhefunktion tulos alkaa lähestyä opetuksen virhefunktio tulosta. Tämä voidaan tulkita siten, että opetusdataa on riittävästi ja mallin kompleksisuus riittää ongelman hahmottamiseen (Muralidhar 2021; Brownlee 2019b; Perlich 2010).

Mallien opetus suoritettiin Azure ML Designerissä kuvan 4 mukaisella kaaviolla. Kaavoissa ylhäällä ladataan ensimmäisenä datajoukko, joka on pakattu ZIP-tiedosto. ZIP-tiedosto sisältää opetukseen käytettävät kuvat lajiteltuina luokan mukaisiin kansioihin. Datajoukko tulee konvertoida algoritmien ymmärtämään taulukkomuotoon. Tämä tapahtuu *Convert to Image Directory* moduulissa. Tämän jälkeen data jaetaan opetus- ja testidatoihin. Opetusdata jaetaan vielä kahteen osaan: opetus- ja validointidata. Jaettuihin datajoukkoihin kohdistaan sama kuvien käsittely. Esikäsittelyn jälkeen opetus- ja validointidata syötetään *PyTorch*-moduuliin, jossa tapahtuu mallin opetus. Lopuksi testataan mallia testidatalla.



KUVA 4. Kuvaruutukaappaus Azure ML Designer kuvanluokittelu kaaviosta

Mallien opetusta varten kuvien kooksi asetettiin 800 pikseliä ja leikattiin 640 pikselin kokoisiksi (*center crop*). Kuvakoon kasvattaminen aiheuttaa muistiongelmia (OOM *out-of-memory*). Ongelma voidaan ratkaista lisäämällä muistia laskentaa suorittavaan tietokoneeseen tai pienentämällä *batch*-kokoa. Ongelma ratkaistiin pienentämällä *batch*-koko neljään. Datajoukko jaettiin seuraavasti: opetusdata 70 %, validointidata 20 % ja testidata 10 %. Muilta osin hyperparametriasetukset ovat samat kuin taulukossa 4 tai asetuksia ei muutettu. Taulukossa 5 on esitetty DenseNet-mallien tulokset. Kaikki mallit on laskettu samalla virtuaalikoneella (Standard_NC6: 6 cores, 56 GB RAM, 380 GB HD, GPU 1 x NVIDIA Tesla K80).

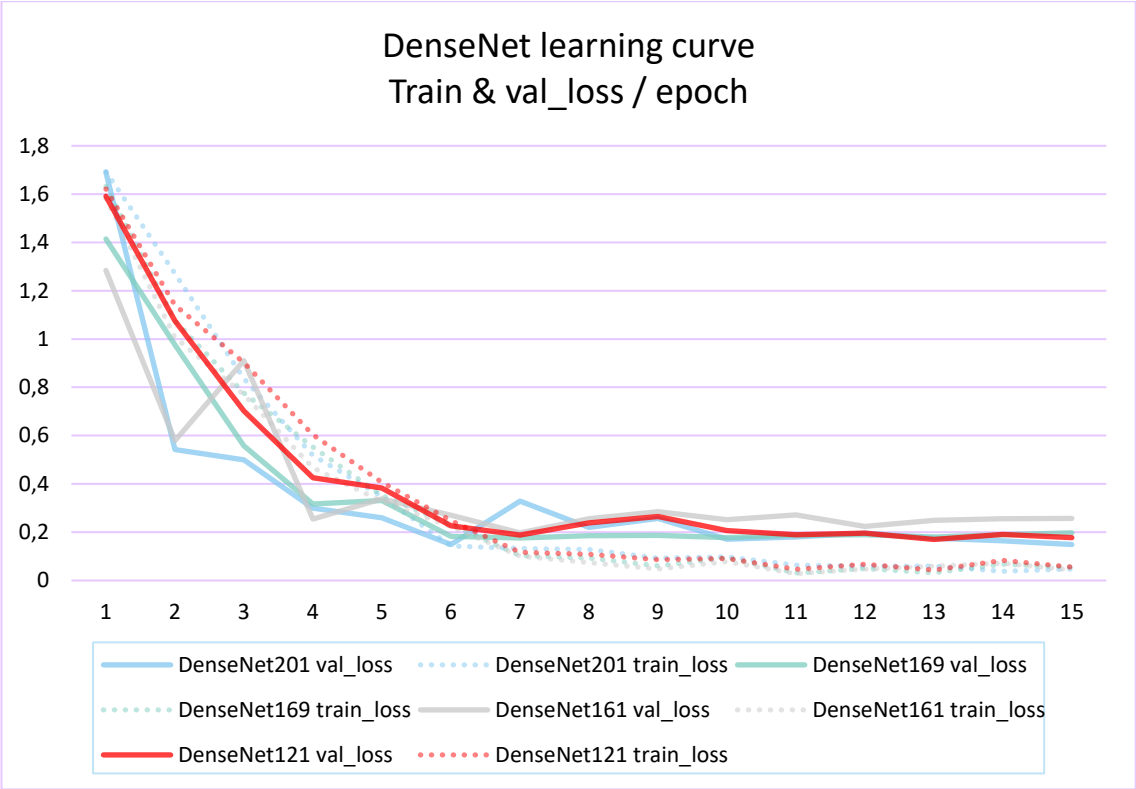
TAULUKKO 5. DenseNet-mallien tulokset

Model	**Training time**	**Macro precision**	**Macro recall**	**Micro precision**	**Micro Recall**	**Overall Accuracy**
DenseNet201	0:45:36	0,958	0,96	0,957	0,957	0,957
DenseNet169	0:36:56	0,954	0,939	0,943	0,943	0,943
DenseNet161	1:07:44	0,936	0,93	0,929	0,929	0,929
DenseNet121	0:30:35	0,916	0,913	0,914	0,914	0,914

Macro *Metriikka lasketaan jokaiselle luokalle erikseen ja näistä otetaan painottomaton keskiarvo. Kaikki luokat ovat saman arvoisia laskennassa.*

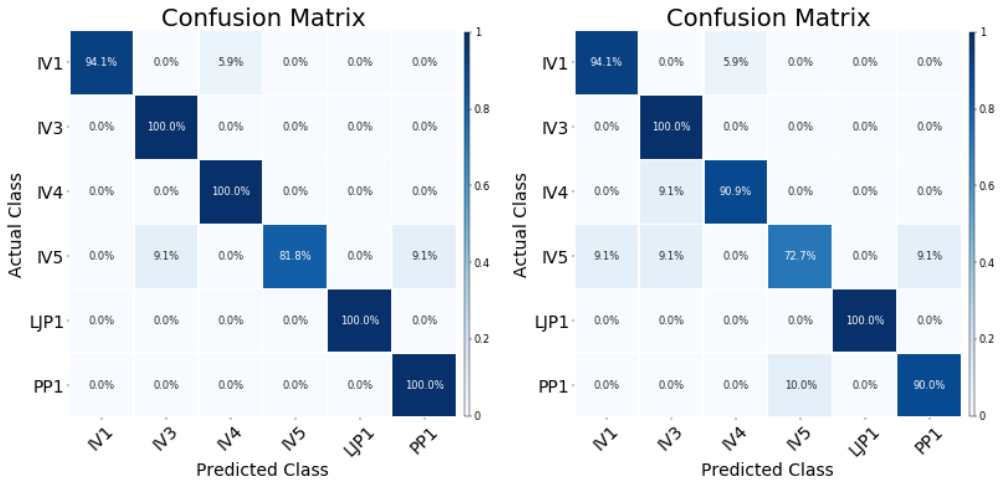
Micro *Metriikka lasketaan koko datajoukolle. Luokkien välisiä epätasapainoja ei huomioida laskennassa. (Microsoft n.d.)*

DenseNet-mallien virhefunktion tulos on visualisoitu kuviossa 38. Kuviosta voidaan tulkita, että kaikki mallit oppivat suunnitellusti eikä yli- tai alisovitusta esiinny. Opetus- ja validointikäyrien väliin jää niin kutsuttu yleistysvirhe (*generalization error*).



KUVIO 38. DenseNet-mallien oppimiskäyrä

DenseNet-mallien tarkkuus, saanti ja tarkkuus ovat lähellä toisiaan. Muutaman prosentin erot voivat johtua pelkästään algoritmien opetuksessa olevasta satunnaisuudesta. Ilman lisätutkimusta ei voida sanoa, että DenseNet201 olisi merkittävästi parempi malli kuin DenseNet121. Mallien tuloksissa oleva pieniä erot tulevat selkeimmin esiin sekaannusmatriisissa (kuvio 39). Luokka IV5 osoittautui kaikille DenseNet-malleille haastavaksi.



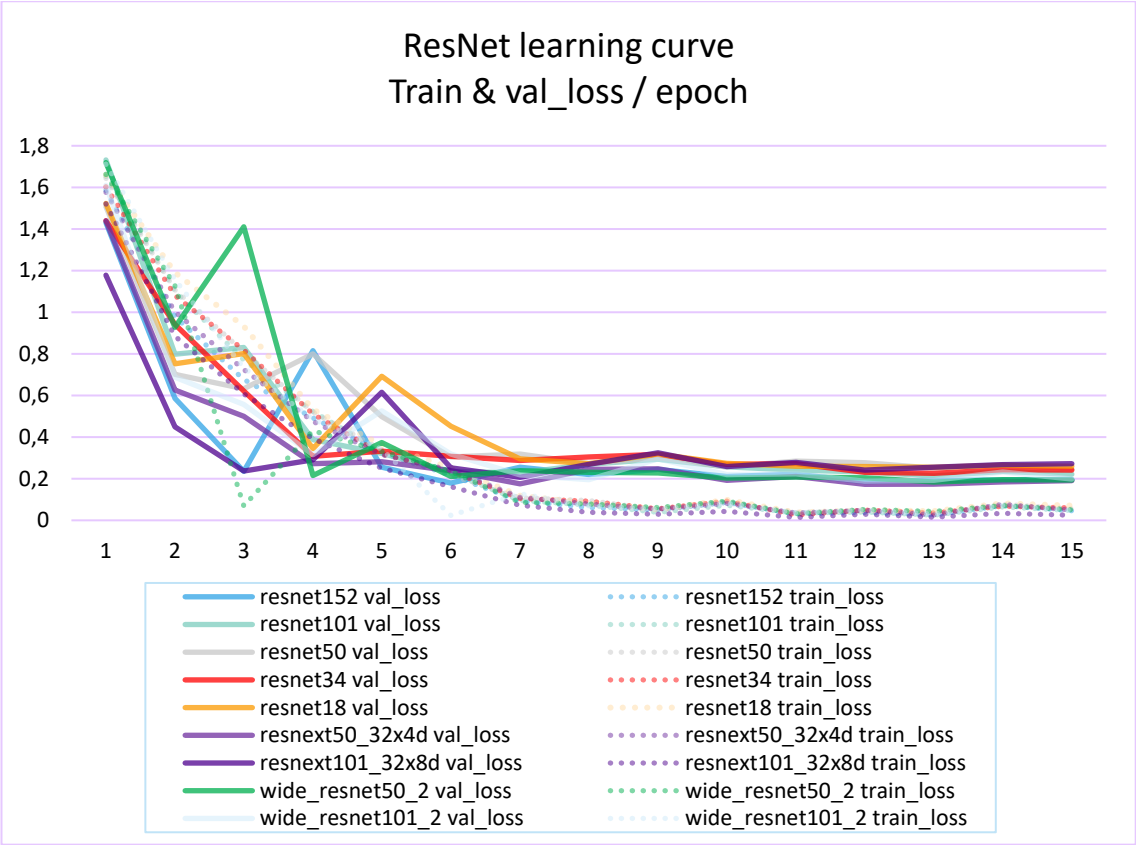
KUVIO 39. Mallien sekaannusmatriisit (oikealla DenseNet201, vasemmalla DenseNet121)

Azure ML Designer tarjoaa kuvantunnistukseen myös ResNet algoritmiin pohjautuvia malleja. Algoritmin muutoksen lisäksi muita hyperparametrimuutoksia ei tehty. ResNet-mallien tulokset on esitetty taulukossa 6.

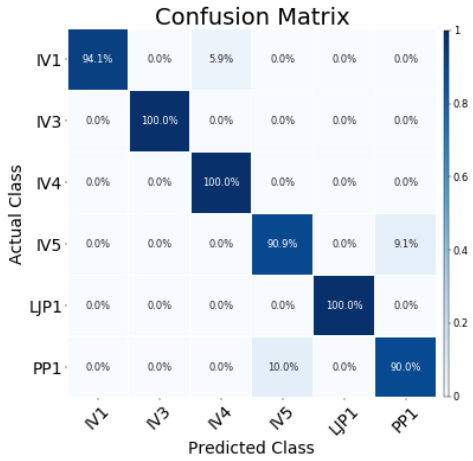
TAULUKKO 6. ResNet-mallien tulokset

Model	Training time	Macro precision	Macro recall	Micro precision	Micro recall	Overall Accuracy
resnet152	1:08:59	0,917	0,924	0,914	0,914	0,914
resnet101	0:48:27	0,959	0,965	0,957	0,957	0,957
resnet50	0:30:01	0,914	0,915	0,914	0,914	0,914
resnet34	0:22:10	0,921	0,915	0,914	0,914	0,914
resnet18	0:12:49	0,908	0,903	0,9	0,9	0,9
res-next50_32x4d	0:44:03	0,954	0,958	0,957	0,957	0,957
res-next101_32x8d	1:57:45	0,946	0,95	0,943	0,943	0,943
wide_res-net50_2	0:56:30	0,958	0,957	0,957	0,957	0,957
wide_res-net101_2	1:42:10	0,922	0,918	0,914	0,914	0,914

ResNet-mallien tulokset ovat saman suuntaisia kuin DenseNet. Oppimiskäyrästä (kuvio 40) nähdään, että kahdeksannen epoch:in kohdalla mallit ovat saavuttaneet virhefunktion minimiarvon. Luokka IV5 osoittautui haastavaksi myös ResNet-malleille. Muutamalla ResNet-mallilla päästiin luokan IV5 osalta noin 90 % tarkkuuteen. Tasapainoisin sekaannusmatriisi saatiin resnext50_32x4d mallia käyttämällä (kuvio 41).

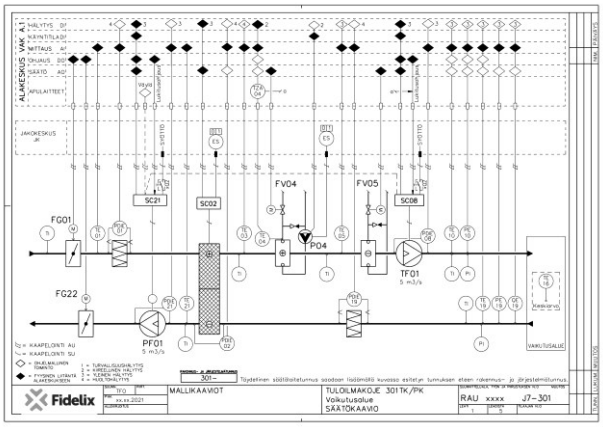


KUVIO 40. ResNet-mallien oppimiskäyrä



KUVIO 41. resnext50_32x4d mallin sekaannusmatriisi

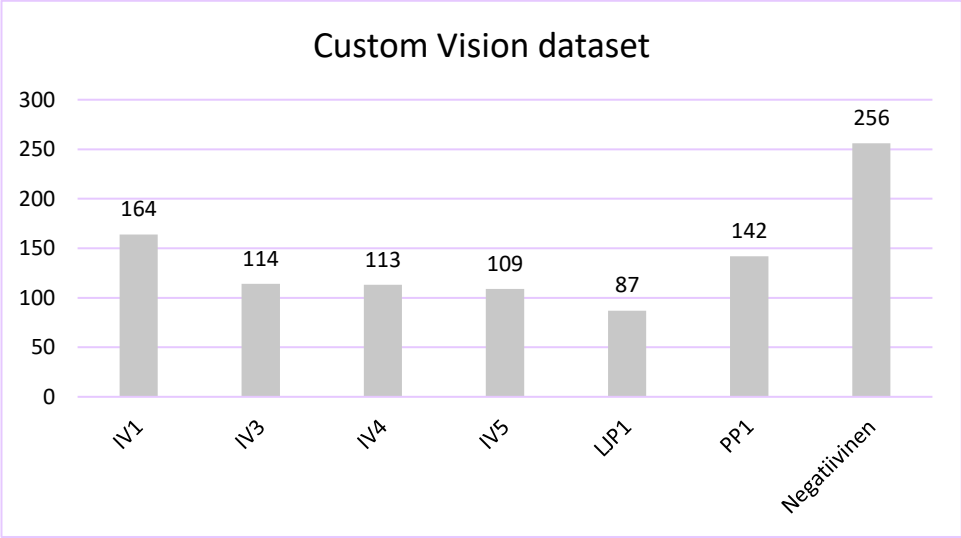
Azure ML Designerissa luotu malli voidaan julkaista, jolloin mallia voidaan käyttää myös Azuren ulkopuolelta. Julkaisua varten mallista luodaan niin sanottu endpoint. Endpointin kutsumista varten Azure ML tarjoaa valmiit kutsukoodit C++, python- ja R-kielillä. Kutsun paluuviestinä saada mallin ennustuksen tulos. Tulos sisältää jokaisen luokan todennäköisyysprosentin sekä *Scored labels* kentässä suurimman todennäköisyysprosentin mukaisen luokan (kuvio 42).




```
{
  "Results": {
    "WebServiceOutput0": [
      {
        "category": "IV1",
        "id": 0,
        "Scored Probabilities_IV1": 0.9999980926513672,
        "Scored Probabilities_IV3": 6.005010391163523e-7,
        "Scored Probabilities_IV4": 0.000001112245740841899 5,
        "Scored Probabilities_IV5": 2.6697588850765896e-7,
        "Scored Probabilities_LJP1": 7.3869506067580915e-9,
        "Scored Probabilities_PP1": 4.04808773168952e-8,
        "Scored Labels": "IV1"
      }
    ] }
```


KUVIO 42. Esimerkki säätökaaviosta, joka on syötetty malliin ja siitä saatava tulos

Kuvien luokittelua suoritettiin myös Custom Vision ohjelmalla. Ohjelmassa ei pysty vaikuttamaan opetuksen hyperparametreihin tai valitsemaan käytettävää algoritmia. Opetusdataan lisätiin negatiivisia kuvia parantamaan mallin suorituskykyä, jonka lisäksi varsinaisiin luokkiin lisättiin uusia kuvia (kuvio 43).



Kuvio 43. Custom Vision opetusdatan tiedostojakauma

Taulukossa x on esitetty Custom Visionilla tehdyn mallin tulokset. Tulokset ovat saman suuntaisia kuin ResNet- ja DensetNet-malleilla, mutta mallin ylioppimista ei voida analysoida näiden tulosten perusteella. Custom Visionin graafinen käyttöliittymä tarjoaa mahdollisuuden käyttää mallia yksittäisillä kuvia. Mallin testaaminen ja verifiointi ulkopuolisella kuvajoukoilla ei onnistu automaattisesti käyttöliittymän kautta. Mallille suoritettiin pikainen muutaman kuvan testi manuaalisesti,

jonka tulokset olivat vaihtelevia. Malli tulkitsee IV4 luokan usein IV1 luokaksi, mutta tämä saattaa johtua satunnaisuudesta. Testiaineisto koostui 20 satunnaisesti valitusta kuvasta, jotka eivät sisältyneet opetusdataan.

TAULUKKO 7. Custom Vision kuvienluokittelu mallin tulokset

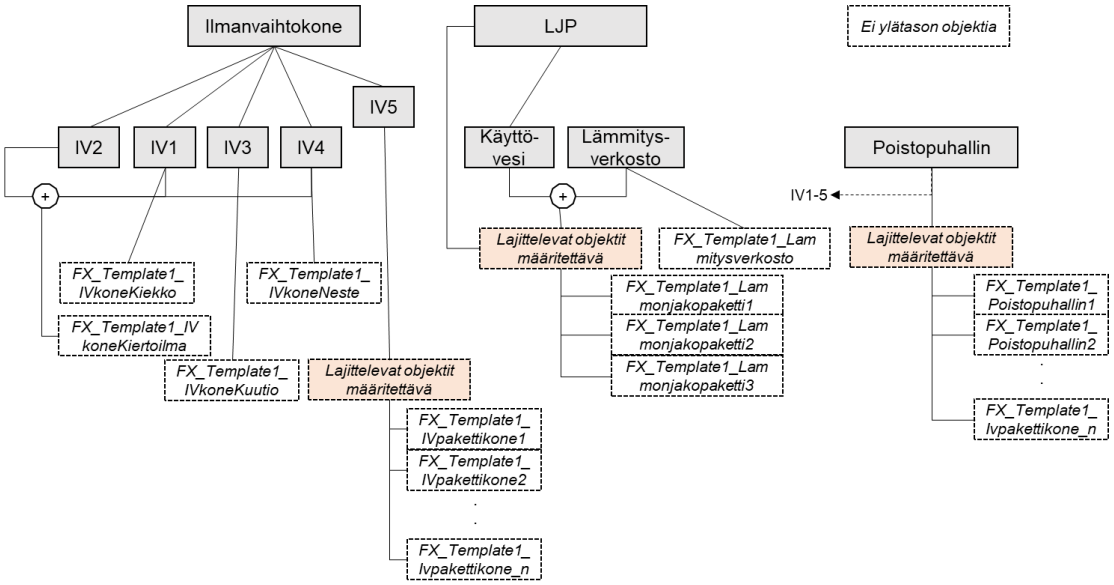
Luokka	**Precision**	**Recall**	**AP**
IV1	0,969	0,939	0,973
IV3	1	0,957	0,962
IV4	0,846	0,957	0,987
IV5	0,955	0,955	0,96
LJP1	0,941	0,889	0,994
PP1	0,774	0,828	0,912
Negatiivinen	0,898	0,863	0,952
Painotettu keskiarvo	0,905	0,905	0,961

Custom Visionilla ja Azure ML Designerillä tehdyt testit osoittavat, että mallipohjan luokittelu kuvasta suoraan on mahdollista. Kuvantunnistuksella tehtävä mallipohjan valinta on pohjimmiltaan opetusdatan keräämistehtävä. Luokkien lukumäärän lisääntyessä opetusdatan määrä tulee kasvamaan merkittävästi. Molempien ohjelmien käyttö on tehty helpoksi, mutta samaan aikaan hyvin rajatuksi. Hyperparametrien säätövaihtoehdot ovat Designerissa rajalliset ja Custom Visionissa olettamattomat. Täten ohjelmat soveltuvat ohjelmakehityksessä vain ideavaiheen demonstrointiin. Custom Visionin tulosten toistaminen on hankalaa, koska Microsoft ei julkisesti kerro, mitä algoritmeja ohjelma käyttää.

7.4.2 Objektin tunnistus

Mallipohjien lukumäärän kasvaessa tarvitaan enemmän opetusdataa. Tällöin päädytään hyvin nopeasti tilanteeseen, missä PI-kaavioiden väliset eroavaisuudet ovat hyvin pieniä tai vähemmän käytetylle mallipohjalle ei löydetä riittävästi PI-kaavioita. Case-tutkimuksessa ongelmaa lähdettiin ratkaisemaan objektin tunnistuksen kautta. Objektin tunnistuksessa on tarkoitus löytää PI-kaavioista avainsymbolit, joiden perusteella voidaan päätellä, mikä mallipohja sopii tähän PI-kaavioon. Symbolilla tarkoitetaan PI-kaaviossa olevia piirrosmerkkejä ja objekti on mallin havaitsema symboli. Kuviossa 44 on esitetty avainsymboleiden suhteet mallipohjiin.

Case-tutkimuksessa symbolit jaettiin ylä- ja alatason ryhmiin. Ylätasolla olevat symbolit määrittelevät mihin mallipohjaperheeseen löydetty objekti kuuluu. Ylätason symboli tulee olla määritelty niin, että niitä ei voi löytyä yhtä kappaletta enempää. Alatason symboli tarkentaa valittavaa mallipohjaa. Alatason symboli voi sisältyä useampaan mallipohjaan. Tällöin ylätason symboli on määräävä. Usean alatason symbolin tilanteita varten tulee luoda valintasäännöt. Ratkaisemattomassa tilanteessa tulee koneoppimissovelluksen palauttaa pääohjelmaan lista kaikista mallipohjista, jotka täyttävät löydettyjen objektien ehdot.



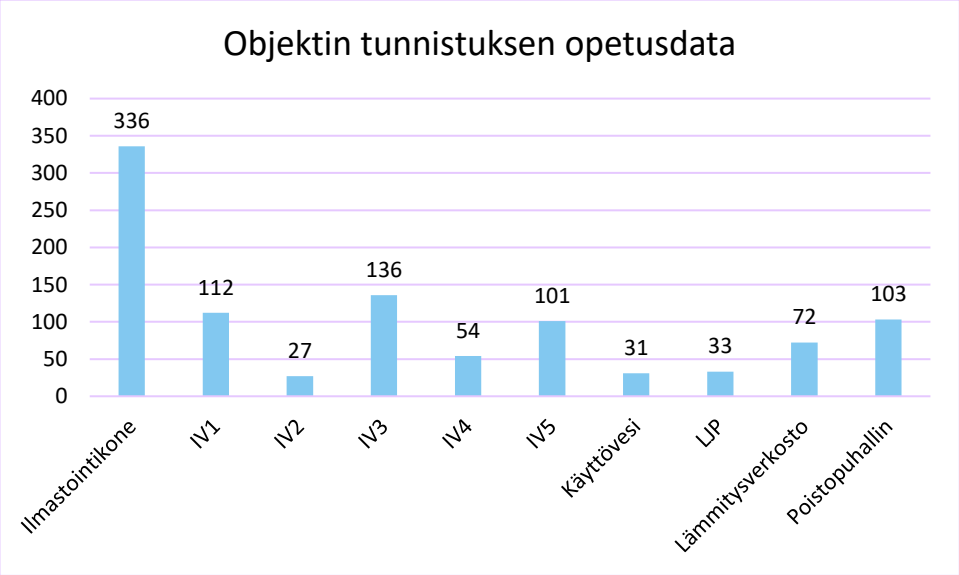
KUVIO 44. Objektien suhteet mallipohjiin, lajittelevan algoritmin perusidea

Kuvion 44 tulkintaesimerkkejä.

- Löydetyt objektit ovat ilmanvaihtokone, IV ja poistopuhallin. Oikea mallipohja on FX_Template1_IVkoneKiekko, koska ylätason objekti määrittelee valinnan.
- Löydetyt objekti ovat IV1 ja IV5. Oikea mallipohja on jonkin IV5-luokan alta löytyvistä mallipohjista. Näihin mallipohjiin tulee määritellä tarkentava objekti. Ilman tarkentavan objektin määrittelyä pääohjelmaan palautetaan lista mallipohjista.
- Löydetty objekti on lämmitysverkosto. Valittava mallipohja on FX_Template1_Lammitysverkosto, koska ei löydetty käyttövesi- tai LJP-objektia.

Objektin tunnistukseen perustuvalla menetelmällä voidaan hallita mallipohjan valintaa paremmin kuin suoraan kuvasta tehtävällä luokittelulla. PI-kaavioista tunnistettavat objektit on määritettävä tarkasti ennen opetusdatan luomista. Opetusdatana voidaan käyttää myös kuvia, joita ei käytetty kuvan luokittelussa. Avainobjektien määrittely vaatii mallipohjien syvällistä tuntemusta.

Case-tutkimuksen objektin tunnistus toteutettiin Custom Vision sovelluksella. Kuviossa 44 harmaalla taustalla olevat laatikot edustavat tunnistettavia objekteja. Näitä on yhteensä 10kpl. Opetusdata sisälsi 540 kuvaa, joihin merkittiin kaikki löytyvät objektit. Löytyneiden objektien lukumäärät on esitetty kuviossa 45.



KUVIO 45. Objektin tunnistuksessa käytetyn opetusdatan objektien lukumäärät

Objektin tunnistusmallin luotettavuuden kynnysarvo asetettiin arvoon 0,50. Objektin tunnistusmallin $mAP@0.3$ arvoksi saatiin 0,953. Mallin tarkkuus oli 0,905 ja saanti 0,908. Mallin luokkakohtainen metriikka on esitetty taulukossa 8.

TAULUKKO 8. Objektin tunnistusmallin luokkakohtainen metriikka

Luokka	Precision	Recall	AP
Ilmastointikone	0,957	0,985	0,984
IV1	0,958	1,000	1,000
IV2	1,000	0,833	1,000
IV3	1,000	1,000	1,000
IV4	1,000	0,909	0,909
IV5	0,731	0,826	0,850
Käyttövesi	0,778	1,000	1,000
LJP	0,700	1,000	1,000
Lämmitysverkosto	1,000	1,000	1,000
Poistopuhallin	0,855	0,746	0,791

Objektin tunnistuksen tulokset ovat lupauksia antavia. Luokat IV5 ja poistopuhallin ovat haastavia mallille. Tämä johtuu luokan sisällä olevasta symboleiden variaatiosta. Molemmissa luokissa on useampi samaa asiaa kuvaa symboli, jotka on opetusdatassa merkattu samaksi objektiksi. Liikaa eroavat symbolit, jotka tarkoittavat samaa asiaa, tulisi objektin tunnistusvaiheessa jakaa kahteen objektiin. Tämä edellyttää opetusdatan kasvattamista. Mallipohjan valinta-algoritmissa voidaan yhdistää samaa tarkoittavat objektit.

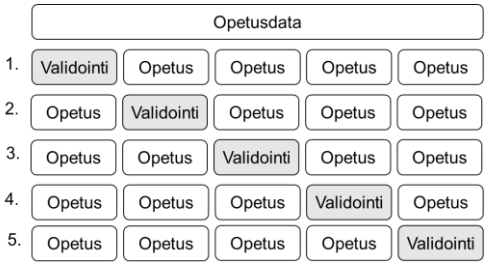
7.5 Ryhmävalintojen tekeminen koneoppimisen avulla

Ryhmävalinnat tehdään mallipohjakohtaisella moniluokkaisella mallilla. Mallien luokkien määrä vaihtelee mallipohjien välillä. Mallipohjat sisältävät 10–40 ryhmää, mutta kaikkien ryhmien arvoa ei voida päätellä säätökaavion PI-kaaviosta. Kaikkien ryhmien päättely vaatii koko säätökaavion tulkitsemista ja mahdollisti lisäinformaatiota muista dokumenteista. Ryhmävalintojen mallit toteutettiin ilmanvaihtokoneille (luokat IV1, IV3 ja IV4). Mallit toteutettiin python-kielellä TensorFlow-kirjastoa käyttäen.

Mallien opetusdatana käytettiin samoja kuvatiedostoja kuin kuvien luokittelussa sekä csv-tiedostoja, joihin oli määritelty kustakin kuvasta löytyvät muuttujat. Muuttujien arvot olivat binäärisiä. Kukin muuttuja viittaa tiettyyn ryhmätunnukseen kussakin mallipohjassa. Opetusdatan määrän oletetaan olevan riittämätön

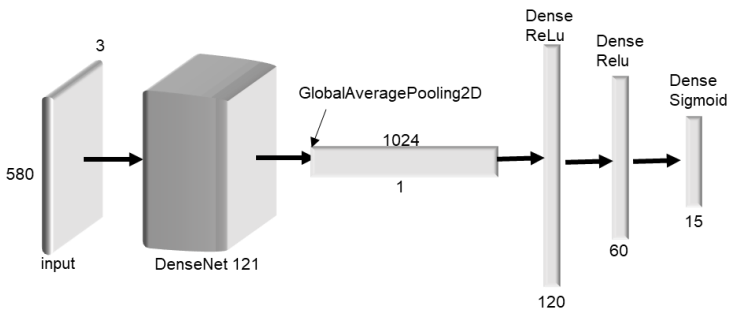
kaikkien ryhmien tunnistamiseen. Tästä syystä ryhmien lukumäärää jouduttiin rajoittamaan. Kuvat skaalattiin 600x600 pikselin kokoisiksi, jonka jälkeen ne leikattiin 580x580 kokoisiksi (*center crop*).

Käytössä olevalla opetusdatalla pyritään tutkimaan sitä, onko menetelmä käyttökelpoinen. Mallin arviointiin käytetään K-ositusmenetelmää satunnaisuuden vähentämiseksi. K-osituksessa opetusdata jaetaan k-osaan, joista yhtä käytetään mallin validointiin (kuvio 46). Opetus suoritetaan k-kertaa ja validointidata vaihtuu opetusten välillä. Lopullinen malli tulee opettaa koko opetusdatalla ja testata ulkopuolisella datalla.



KUVIO 46. K-ositus ristiinvalidoinnin periaate, K=5

LTO-kiekolla varustetun ilmanvaihtokoneen (IV1) mallissa hyödynnettiin DenseNet121-neuroverkkoa. DenseNet121-mallista poistettiin vakio ulostulokerros ja mallin perään lisättiin kaksi piilokerrosta ja ulostulokerros (kuvio 47). Piilokerrosten aktivointifunktiona käytettiin ReLua ja ulostulokerroksessa Sigmoid:ia. Mallin virhefunktiona käytettiin *Binary Cross Entropy:a* LTO-kiekolla varustetun ilmanvaihtokoneen (IV1) moniluokittelumallin rakenne oli kuvion 45 mukainen. Ulostulokerroksen neuroneiden lukumäärä on mallin ennustettavien muuttujien lukumäärä.



KUVIO 47. LTO-kiekko ilmanvaihtokonetta varten muokattu DenseNet121 pohjainen neuroverkko

Moniluokkaista mallia voidaan arvioida vertaamalla mallin tulosta ja tavoitetta. Tuloksen ja tavoitteen täsmätessä täysin käytetään EMR (*exact match measure*) -mittaria. EMR tunnetaan myös nimellä *0/1 loss*, joka voidaan laskea kaavalla 63. (Read ym. 2011.)

$$EMR = \frac{0}{1} LOSS = 1 - \frac{1}{N} \sum_{i=1}^{N} 1_{y^i = \hat{y}^i}, \tag{63}$$

missä *N* on instanssien lukumäärä. Mittarin huonona puolena pidetään sitä, että se ei anna kokonaiskuvaa mallin toimivuudesta. Yhdenkin luokan ollessa instanssia kohdin väärin, mittari antaa nollatuloksen instanssille. *Hamming Loss* suhtautuu lempeämmin yksittäisten luokkien vääriin tuloksiin (kaava 64). Jokainen oikea tulos luokkaa kohti parantaa tulosta. (Read ym. 2011.)

$$HAMMING\ LOSS = 1 - \frac{1}{NL} \sum_{i=1}^{N} \sum_{j=1}^{L} 1_{y_j^i = \hat{y}_j^i} \tag{64}$$

Moniluokkaisen mallin tarkkuus, jossa huomioidaan myös osittainen tulosten vastaavuus, voidaan laskea kaavalla 65. (Read ym. 2011)

$$ACCURACY = \frac{1}{N} \sum_{i=1}^{N} \frac{|y^i \wedge \hat{y}^i|}{|y^i \vee \hat{y}^i|} \tag{65}$$

Ilmanvaihtokone, LTO-kiekko (IV1)

LTO-kiekolla varustetun ilmanvaihtokoneen mallipohja sisältää kokonaisuudessa 38 ryhmätunnusta. Näistä noin 15 pystytään päättelemään PI-kaaviosta. Kuvassa 5 on esitetty csv-tiedostosta ladatun datan sisältöä. Malli laskettiin kaikille ominaisuuksille. Tämän lisäksi mallin laskentaa yksinkertaistettiin vähentämällä ominaisuuksia mallista enemmistösäännön avulla.

```
train = pd.read_csv('dataset/IV1_labels_v3.csv', sep=';')   # reading the csv file
train.head()      # printing first five rows of the file
```

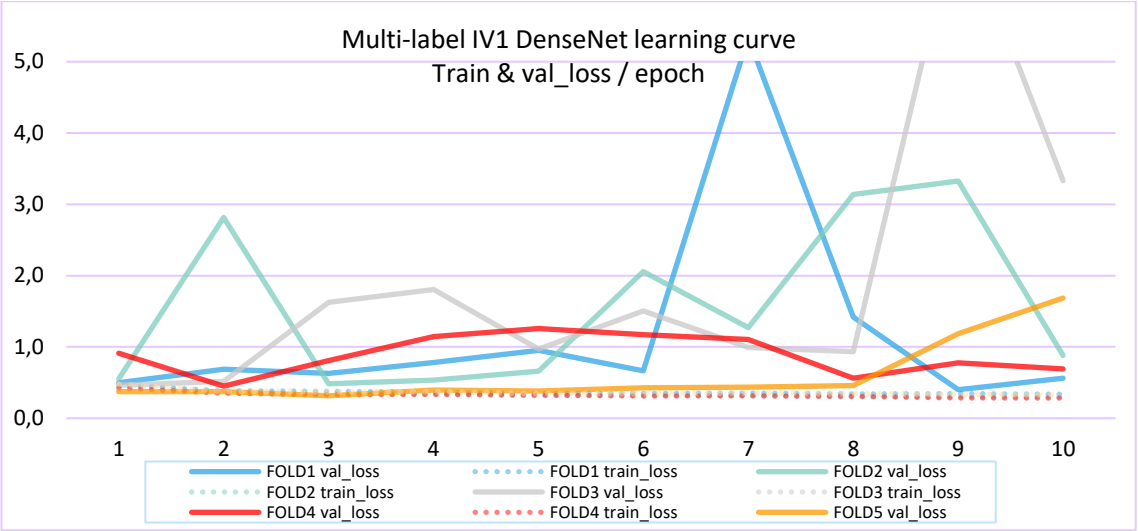
	ID	PuhallinSC_EC	Ilmamäärä	Kanavapaine	Kiekko	LP	JP	Lämpötila_LTO	Lämpötila_Lpjälkeen	Lämpötila_raitisilma	Lämpötila_jäteilma	Suodatin_tulo	Suodatin_poisto	Paine_erotLTO	CO2_poistoilma	Kosteus_poistoilma
0	4.PNG	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
1	8.PNG	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0
2	17.PNG	1	1	1	1	1	0	1	0	0	0	1	1	1	0	0
3	25.PNG	1	1	1	1	1	0	1	0	0	0	1	1	1	0	0
4	29.PNG	1	1	0	1	1	0	1	0	0	0	1	1	0	0	0

KUVA 5. Kuvaruutukaappaus, jossa IV1 mallin opetusdatan rakenne

Enemmistösääntöä sovellettiin sitten, että jokaiselle ominaisuudelle laskettiin enemmistön mukainen arvo ja kaikkien arvojen keskiarvo. Koska ominaisuudet ovat arvoiltaan binäärisiä, keskiarvo on samalla todennäköisyys sille, että ominaisuuden arvo on yksi. Opetusdatasta muodostettiin enemmistösäännön mukainen tulos,$y_{IV1}^{es1} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0]$. Opetusdatalle laskettiin tämän jälkeen tarkkuus, jos kaikki syötteet saisivat tuloksena y_{IV1}^{es1}. Pelkästään käyttämällä enemmistösäännön mukaista tulosta y_{IV1}^{es1} saadaan tarkkuudeksi 0,734. Taulukossa 9 on esitetty kaikki 14 ominaisuutta sisältävän mallin tulokset. Tarkkuus on lähellä enemmistösäännön mukaista tarkkuutta. Koneoppiminen ei ole tehnyt mallia yhtään paremmaksi. K-ositusten välillä on joitakin eroja, mutta ne eivät muuta tulkintaa mallin toimivuudesta. Kaikki ominaisuudet sisältävän mallin oppimiskäyrästä (kuvio 48) voidaan todeta, että opetusdataa ei ole riittävästi ongelman mallintamiseen, eikä käytetty malli ei ole riittävän kompleksinen omaksuakseen ilmiön luonnetta.

TAULUKKO 9. Mallipohja IV1 moniluokkamallin tulokset (14 ominaisuutta)

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,787	0,000	0,176	0,903	0,864	0,876
K-fold 2	0,769	0,061	0,192	0,909	0,832	0,862
K-fold 3	0,618	0,000	0,321	0,728	0,801	0,757
K-fold 4	0,726	0,031	0,208	0,816	0,887	0,835
K-fold 5	0,701	0,000	0,246	0,799	0,853	0,822
Avg	0,720	0,018	0,229	0,831	0,847	0,830
Std	0,067	0,027	0,058	0,076	0,033	0,046



KUVIO 48. Moniluokkaisen IV1-mallin virhefunktion oppimiskäyrät

Ominaisuusjoukossa on useita ominaisuuksia, joiden enemmistösäännön mukaisen arvon esiintyvyys on yli 90 % tapauksista. Osa ominaisuuksista saa saman arvon jopa 98 % tapauksista. Tällainen arvojen jakauma on täysin epätasapainoinen ja tätä kautta erittäin haastava mallin opetuksen kannalta. Ongelma vaihtuu luokittelusta poikkeamien tunnistamiseksi (*anomaly detection*). Tämän opinnäytetyön laajuudessa ei käsitellä poikkeamien tunnistamista. Opetusdatasta päätettiin rajata ulos ominaisuudet, joiden enemmistön mukaisen arvon esiintyvyys on yli 90 %.

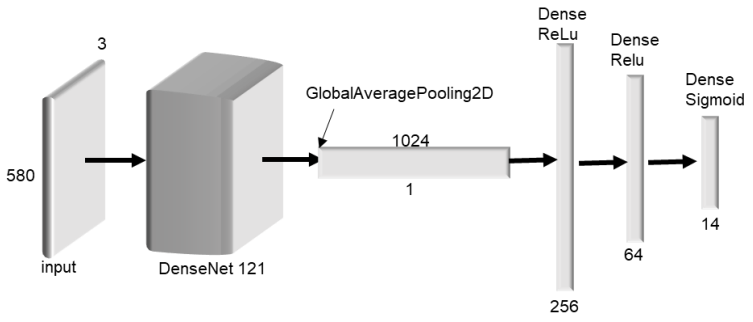
Luokan IV1 opetusdatasta rajattiin ulos seitsemän ominaisuutta. Näin jäljelle jääneiden muuttujien määrä on kahdeksan. Rajatusta opetusdatasta muodostettiin uusi enemmistösäännön mukainen tulos $y_{IV1}^{es2} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 0]$, jolle saatiin tarkkuudeksi 0,531. Mallin hyväksyttävyyttä arvioidaan tätä raja-arvo vasten. Rajatulla opetusdatalla opetetun mallin tulokset on esitetty taulukossa 10. K-ositusten tarkkuuden keskiarvo on melko lähellä enemmistösäännön mukaisesti saatua tarkkuutta.

TAULUKKO 10. Mallipohja IV1 moniluokkamallin tulokset (8 ominaisuutta)

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,513	0,061	0,318	0,725	0,636	0,646
K-fold 2	0,456	0,000	0,341	0,733	0,568	0,604
K-fold 3	0,542	0,121	0,288	0,632	0,798	0,679
K-fold 4	0,558	0,000	0,293	0,743	0,703	0,698
K-fold 5	0,452	0,156	0,363	0,515	0,667	0,568
Avg	0,504	0,068	0,321	0,670	0,674	0,639
Std	0,049	0,071	0,032	0,097	0,085	0,053

Ilmanvaihtokone, LTO-kuutio (IV3)

IV3-luokan mallissa olevien piilokerrosten neuroneiden lukumäärää kasvatettiin. Mallin rakenne esitetty kuviossa 49. Luokan IV3 mallin alkutilanteen tunnistettavien ominaisuuksien lukumäärä oli 14. Näille ominaisuuksille laskettiin enemmistösäännön mukainen tulos $y_{IV3}^{es1} = [1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0]$. Opetusdata testattiin tuloksella ja malli sai tarkkuudeksi 0,806. Luokka IV3 sisältää enemmän variaatioita kuin IV1-luokka. Variaatioista huolimatta viiden ominaisuuden vaihtelu oli hyvin pientä (muutaman instanssin luokkaa). Luokan IV3 rajatusta opetusdatasta muodostettu $y_{IV3}^{es2} = [1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1]$ sai tarkkuudeksi 0,720.



KUVIO 49. DensNet121-neuroverkko muokattu LTO-kuutio ilmanvaihtokoneille

Piilokerrokseen lisätyillä neuroneilla ei ollut vaikutusta mallin hyväksyttävyyteen (taulukko 11). Käyttämällä pelkästään enemmistösääntöä päästään parempaan tarkkuuteen kuin mallin avulla. Mallin oppimiskäyrä oli saman suuntainen kuin kuviossa 46 on esitetty. Validoinnin virhefunktion tulos huojuu merkittävästi ja saa usean kymmenen suuruisia arvoja muutamalla laskentakerralla. Tämä johtuu siitä, että opetusdataa ei ole tarpeeksi, jotta malli pystyisi omaksumaan ilmiön.

TAULUKKO 11. Mallipohja IV3 mallin tulokset (14 ominaisuutta)

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,718	0,000	0,214	0,928	0,770	0,829
K-fold 2	0,552	0,000	0,379	0,893	0,597	0,705
K-fold 3	0,673	0,000	0,283	0,985	0,685	0,796
K-fold 4	0,706	0,000	0,236	0,880	0,786	0,823
K-fold 5	0,711	0,043	0,220	0,770	0,902	0,824
Avg	0,672	0,009	0,266	0,891	0,748	0,795
Std	0,069	0,019	0,068	0,079	0,115	0,052

Rajatulle opetusdatalle lasketun mallin tulokset on esitetty taulukossa 12. Tuloksista käy ilmi, että malli ei saavuttanut hyväksyttyä tasoa.

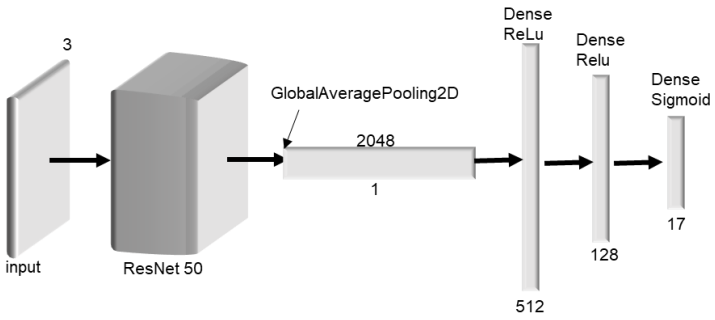
TAULUKKO 12. Mallipohja IV3 mallin tulokset (9 ominaisuutta)

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,595	0,000	0,333	0,911	0,648	0,724
K-fold 2	0,597	0,000	0,338	0,862	0,673	0,731
K-fold 3	0,611	0,000	0,343	0,968	0,629	0,741
K-fold 4	0,495	0,043	0,386	0,560	0,772	0,637
K-fold 5	0,669	0,000	0,266	0,846	0,783	0,793
Avg	0,593	0,009	0,333	0,829	0,701	0,725
Std	0,063	0,019	0,043	0,158	0,072	0,056

Näiden testien perusteella ei voida sanoa, että soveltuuko DenseNet käytettäväksi ilmiön mallintamiseen. Ilmiön mallinnuskokeiluja DenseNet-pohjaisilla neuroverkoilla on kuitenkin mahdollista jatkaa, jos opetusdataa saadaan kerätty enemmän.

Ilmanvaihtokone, LTO-neste (IV4)

Luokan IV4 mallin neuroverkon pohja-arkkitehtuuri vaihdettiin ResNet-pohjaiseksi (kuvio 50). ResNet-arkkitehtuureista käyttöön otettiin ResNet 50. Samalla kasvatettiin piilokerrosten neuroneiden lukumäärä, koska ResNetin ulostulokerros on suurempi kuin DenseNetin (1024 → 2048).



KUVIO 50. ResNet pohjaisen IV4 mallin rakenne.

Pohja-arkkitehtuurin muutos nosti merkittävästi opeteltavien parametrien lukumäärää. DenseNet-malleilla parametrien lukumäärä oli noin 7,2 miljoonaa. ResNet-mallissa parametrien lukumäärä oli 24,7 miljoonaa. ResNet-mallien laskenta-aika kasvoi noin 1,75 kertaiseksi.

Mallille IV4 laskettiin enemmistösäännön mukainen tulos $y_{IV4}^{es1} = [1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1]$. Opetusdatan testaaminen y_{IV4}^{es1} tuloksella antaa tarkkuudeksi 0,797. Tuloksesta voidaan todeta, että variaatioiden määrä opetusdatassa on hyvin pieni. Malli IV4 opetettiin kuvion 50 mukaisella neuroverkolla ja keskimääräiseksi tarkkuudeksi saatiin 0,719. Opetetulla mallilla ei päästä enemmistösäännön tulokseen.

TAULUKKO 13. Mallin IV4 tulokset ominaisuuksien määrän ollessa 17

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,780	0,000	0,182	0,900	0,859	0,872
K-fold 2	0,771	0,000	0,184	0,842	0,910	0,868
K-fold 3	0,669	0,000	0,251	0,826	0,773	0,788
K-fold 4	0,681	0,045	0,270	0,968	0,695	0,800
K-fold 5	0,696	0,000	0,230	0,848	0,814	0,804
Avg	0,719	0,009	0,223	0,877	0,810	0,826
Std	0,052	0,020	0,040	0,058	0,082	0,040

Opetusdatasta karsittiin pois ominaisuudet, joiden enemmistösäännön mukaisen arvon esiintyvyys oli 90 %. Näin saadun uuden tuloksen $y_{IV4}^{es2} = [1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ ]$ tarkkuus opetusdatalle testatessa oli 0,741. Näistä kahdeksalla ominaisuudella oli enemmistösäännön mukaisen arvon esiintyvyys yli 75 %. Malli opetettiin rajatulla opetusdatalla, jonka tulokset on esitetty taulukossa 14.

TAULUKKO 14. Mallin IV4 tulokset rajatulla opetusdatalla (ominaisuuksien määrä 12)

	Accuracy	**Exact Match Ratio**	**Hamming loss**	**Precision**	**Recall**	**F1 score**
K-fold 1	0,727	0,130	0,239	0,952	0,752	0,831
K-fold 2	0,685	0,045	0,280	0,926	0,727	0,804
K-fold 3	0,640	0,000	0,303	0,855	0,732	0,760
K-fold 4	0,683	0,000	0,277	0,837	0,803	0,800
K-fold 5	0,728	0,000	0,231	0,882	0,813	0,835
Avg	0,693	0,035	0,266	0,890	0,766	0,806
Std	0,037	0,057	0,030	0,048	0,040	0,030

Moniluokkamalleilla ei päästy hyväksyttäviin tuloksiin. Kaikissa tapauksissa enemmistösäännön mukaisella tuloksella opetusdataa testatessa saatiin parempi tarkkuus. Koneoppimismallit eivät pystyneet sisäistämään ilmiötä liian pienen opetusdatan takia. Mallien kompleksisuuden riittävyydestä ei päästy selkeään lopputulokseen.

Case-tutkimuksen opetusdatan kuvat skaalattiin ja leikattiin 580 pikselin kokoisiksi. Skaalauksen jälkeen kuvat ovat luettavissa, mutta kaikista yksityiskohdista ei saa selvää. Alkuperäinen kuvasuhde oli noin 1,4:1. Skaalauksessa kuva puristettiin 1:1 kuvasuhteeseen. Tämä vääristää kuvassa olevia symboleita ja tekee niistä vaikeammin tulkittavia. Kuvasuhde 1:1 tulee vaatimuksena ohjelmakirjastoista, joissa kuvan leveys ja korkeus pitää olla samat. Oleellista informaatiota on mahdollisesti menetetty liikaa. Jatkotutkimuksissa kuvien skaalaus täytyy tehdä ilman, että kuvasuhde muuttuu. Tämä voidaan tehdä lisäämällä kuvaan täytekehykset. Kehyksissä pikseleiden arvo on nolla, eivätkä täten sisällä konvoluutiossa louhittavia piirteitä. Lisäksi kuvien skaalausta on kevennettävä. Tätä ei pystytty

toteuttamaan tässä case-tutkimuksessa käytössä olleella tietokoneella. Tietokoneen muisti ei riittänyt isompien kuvien käsittelyyn *batch size* asetuksen ollessa neljä.

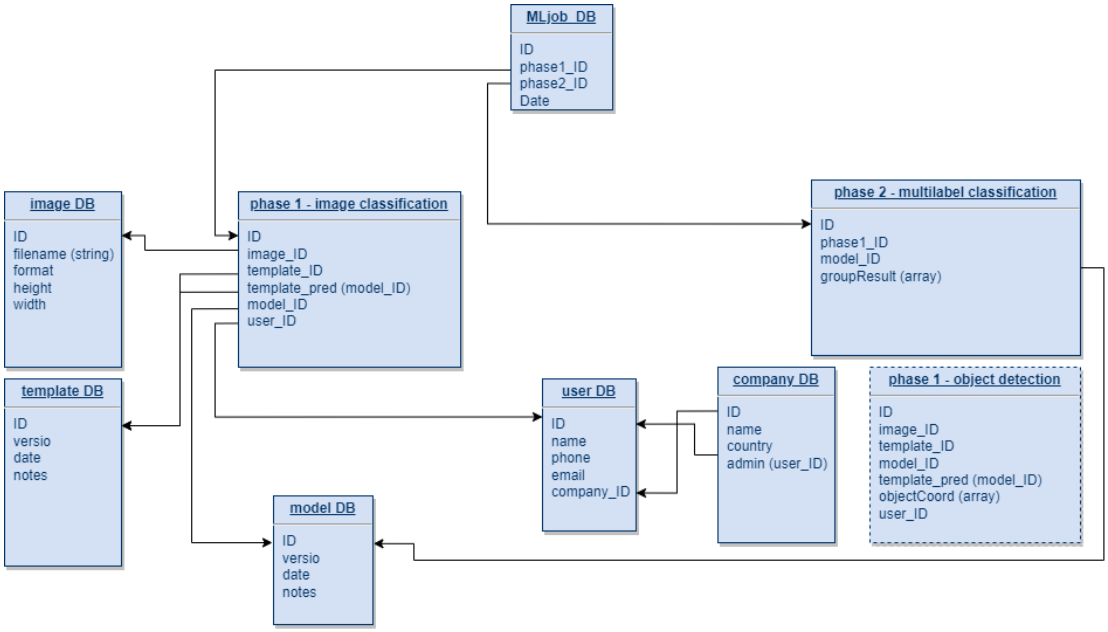
7.6 Integrointi osaksi FX-editorin toimintoja

Koneoppimisen käyttäminen FX-editorissa edellyttää yhteyttä pilveen. Koneoppimismoduulin lisäksi pilviä voidaan käyttää päivitysten jakeluun. Fidelix-järjestelmän on kolme keskeistä komponenttia, joissa on jatkuvaa versio kehitystä. Näitä ovat FX-editori, ala-aseman BIN-versio sekä IEC/Template-ohjelmat. Kaikkien näiden päivitykset ovat tällä hetkellä käyttäjän vastuulla. Päivitykset on tehtävä manuaalisesti. Osittain tästä syystä useiden Fidelix-järjestelmän ohjelmoitsijoiden versiot ovat todella vanhoja, joka aiheuttaa sen, että samoista jo korjatuista ongelmista tulee jatkuvasti palautetta. Päivitysten automatisointi on pilvi-integraation yksi tärkeimmistä hyödyistä.

Koneoppimismoduulin käyttö edellyttää, että FX-editorin kykenee vastaanottamaan esitellyn mukaisen kutsukoodin. Kutsukoodin pohjalta avataan mallipohjan muokkausnäkymä, jossa käyttäjä voi tehdä viimeiset valinnat. Tehdyt valinnat ja kuva tulee tallentaa. Tallennus voidaan toteuttaa paikallisesti tai ladata tallennettu tieto suoraan pilveen. Paikallinen tallennus on tarpeen mahdollista offline-käyttöä varten. FX-editorin käynnistyessä tarkastetaan pilviyhteyden tila ja jos yhteys on kunnossa, ladataan tallennetut tiedot pilveen. Mallipohjien käyttäminen tulee olla mahdollista myös offline- sekä online-tilassa ilman, että tietoja tallennetaan pilveen. Tietojen tallennuksesta pilveen tulee lisätä sopimusteksti ohjelman käyttöehtoihin. Koneoppimismoduulin toimintaperiaate on kuvattu tarkemmin liitteessä 1.

Pilvessä sijaitsevan tietokannan rakenne tulee olla suunniteltu skaalautuvaksi sekä robustiseksi. Opinnäytetyössä toteutettujen osien tietokanta oli erillään toisistaan. Tuotantomallien tietokanta tulee olla yhtenäinen ja yhdestä paikkaa hallittavissa. Tämä vaatimus rajaa Custom Vision ohjelman käytön pois tuotantomallien luonnista. Azure ML Designer mallit vaativat tarkasti määritellyllä tavalla järjestetyn opetusdatan. Designeriin pystyy luomaan omia moduuleita, mutta samalla työmäärällä koko prosessi voidaan tehdä ohjelmakoodilla (python).

Relaatiotietokannan rakenne on hahmoteltu kuviossa 51. Tietietokantaan tulee tallentaa kaikki oleellinen informaatio mallin uudelleenopetusta varten. Tilastointia varten tarvitaan päivämäärä ja tieto käytetystä mallista. Ensimmäisen vaiheen mallin uudelleenopetusta varten tarvitaan tieto kuvasta ja vahvistetusta mallipohjasta. Toisen vaiheen mallien uudelleen opetusta varten tulee tallentaa myös ryhmävalinnat. Tallennettavat ryhmävalinnat voivat sisältää kaikki mallista löytyvät ryhmät. Näin saadaan tilastoja muiden kuin PI-kaaviosta selviävin ryhmien valinnoista. Tätä informaatiota voidaan hyödyntää tulevissa kehityshankkeissa. Käyttäjäinformaatiota voitaisiin käyttää tilanteissa, jossa halutaan poistaa tietyn käyttäjän luomat datat tietokannasta syystä tai toisesta.



KUVIO 51. Relaatiotietokannan hahmotelma

Mallipohjan valinnan tapahtuessa objektin tunnistuksen kautta, on tietokantaan tallennettava löydettyjen objektien x-y koordinaatit. Löydösten oikeellisuus on tarpeen tarkastaa ennen kuin niitä käytetään mallin uudelleenopetuksessa.

8 JOHTOPÄÄTÖKSET JA POHDINTA

8.1 Opetusdata ja menetelmät

Opetusdatan kerääminen ja hallinta on koneoppimisprojektin onnistumisen pohja. Hallinnalla tarkoitetaan datan esikäsittelyä, karsimista ja tasapainoisten datajoukkojen muodostamista, jotta mallien opetusalgoritmit toimisivat tehokkaasti. Datan kerääminen oli tämän case-tutkimuksen hitain ja työllistävin vaihe. Datan noutaminen tietovarannoista tapahtui manuaalisena työnä, joka on hieman ristiriidassa sen kanssa, minkälaista järjestelmää koneoppimissovellusta varten olisi tarkoitus rakentaa. Datan keräämistä tulisi tehostaa ja automatisoida, jotta työaikaa voitaisiin käyttää enemmän datan hallintaan.

Dataa kerättiin projektiputken toteutusvaiheesta (projektipäälliköt ja projektinhoitajat). Jälkikäteen mietittynä dataa olisi pitänyt kerätä projektiputken alkupäästä, eli myynniltä. Näin data saadaan koneoppimisprojektin käyttöön pienemmältä määrältä ihmisiä, jolloin kommunikointi on tehokkaampaa. Tämän lisäksi saataisiin dataa kaikista projekteista, myös niistä mitkä eivät mene toteutukseen asti. Osittain tästä syystä case-tutkimukseen kerätty data oli jossain määrin puolueellista tai vinoutunutta. Data oli puolueellista siinä mielessä, että isojen suunnittelijatoimistojen kuvat olivat yliedustettuina. Opetukseen valitut kuvat olivat luonteeltaan ja laadultaan sellaisia, että ne sopivat käytettäväksi ohjelmakirjaston kanssa. Tulisikin kiinnittää enemmän huomiota sellaisiin kuviin, mitkä eivät sovi käytettäväksi. Analysoimalla ei-sopivia kuvia voidaan mahdollisesti oppia jotain uutta, joka antaa uusia ideoita ohjelmakirjaston kehittämiseen.

Kuvien luokitteluun tarvitaan paljon opetusdataa jokaista luokkaa kohden. Kuvien luokittelu toteutettiin Azure ML Designerissä, jossa on rajalliset mahdollisuudet säätää algoritmien hyperparametrejä. Kuvien luokittelu tulisi toteuttaa vapaammassa ympäristössä, jossa pystyttäisiin tekemään esimerkiksi datan augementointia. Olemassa olevista kuvista voitaisiin generoida 1/10-suhteella lisää kuvia.

Objektin tunnistusta varten ei välttämättä tarvitse hankkia uusia kuvia, koska mallipohjien kannalta ei-sopivia kuvia voidaan käyttää objektien tunnistukseen. Objektin tunnistuksen suurin haaste on pohtia mitkä ovat mallipohjan valinnan kannalta oleelliset objektit. Objektien merkitseminen ei ole haastavaa, mutta se on erittäin työlästä. Objektin tunnistus antaa enemmän mahdollisuuksia oikean mallipohjan määrittämiseen valinta-algoritmin kautta. Tämän lisäksi valinta-algoritmiin on mahdollista tehdä muutoksia ilman mallin uudelleenopetusta. Kuvan luokittelussa muutokset tapahtuvat säätämällä opetusdataa.

Case-tutkimus oli rajattu muutamaan mallipohjan valintaan. Tästä huolimatta opetusdataa oli lopulta liian vähän ongelman kompleksisuuteen nähden. Tuotantosovellusta varten opetusdatan määrä tulisi moninkertaistaa ja datan tulisi olla monipuolisempaa.

8.2 Säätökaaviot

Säätökaavioihin on sisällytetty paljon informaatiota. Toimintaselostuksessa kuvataan järjestelmän toimintaa sanallisesti. Sanalliseen kuvaukseen liittyy usein tulkinnanvaraisuus. Säätökaavion toimintaselostuksen tulkinta oli rajattu case-tutkimuksen ulkopuolelle. Case-tutkimuksessa pyrittiin löytämään menetelmä, jolla PI-kaaviosta voidaan tunnistaa tarpeelliset elementit ja muodostaa niistä kutsukoodi. Kutsukoodilla generoidaan järjestelmän grafiikka ja ohjelmakoodi. Kutsukoodin ajatus pohjautuu siihen, että ohjelmakirjastoa voidaan käyttää sellaisenaan järjestelmän ohjelmointiin. Tässä on ristiriita siinä, miten rakennusautomaatiota on Suomessa totuttu tekemään.

Suomessa suunnittelutoimistot tekevät hyvin omannäköisiä säätökaavioita. Järjestelmien toiminnassa on eroja, vaikka I/O-pisteet olisivat samat. Muualla Euroopassa on käytössä enemmän vakioituja *de-facto-standard* tyylisiä säätökaaviota. Tämän lisäksi tilaajat hyväksyvät helpommin laitetoimittajan ja urakoitsijan omia säätökaavioita. Tämän kaltaisiin suunnitelmiin on merkittävästi helpompi tehdä toimiva ohjelmakirjasta, koska variaatioiden lukumäärä on paljon rajatumpi kuin Suomessa.

Säätökaavioissa on yhteisiä elementtejä, mutta suunnittelutoimistojen välillä on kuitenkin isojakin eroja. Suurin ja haastavin ero on prosessin symboleiden piirtämisjärjestyksessä. Suuressa osassa kaavioita prosessin alkupää on vasemmalla ylhäällä, palvelualue on oikealla ja prosessin loppupää vasemmalla alhaalla. Toisaalta järjestys saattaa olla myös jotain ihan muuta. Tämä aiheuttaa ongelman tulkitessa symboleiden välisiä yhteyksiä. Tulkitsemalla vain kahta symbolia ei aina päästä oikeaan tulokseen vaan prosessia pitää tulkita kokonaisuutena.

Rakennusautomaatioon liitettäviin laitteisiin (anturit, pumput, venttiilit jne.) liittyy aina laitetunnus. Laitetunnuksien alkuosa (TE, ME, QE jne.) on jossain määrin standardisoitu, mutta loppuosalle (TE10, TE16, TE19 jne.) löytyy useita käytössä olevia merkintätapoja. Laitetunnuksen alkuosa määrittelee laitteen tyypin; esimerkiksi TE (*temperature element*) on lämpötila-anturi. Laitetunnuksen numeroosa määrittää yleensä laitteen toiminnallisuuden prosessi. Laitteiden numerointiin on käytössä useita eri merkintätapoja. Merkintätavan määrittää yleensä kiinteistön omistaja tai suunnittelijatoimisto.

PI-kaaviosta pystytään koneellisesti lukumaan laitetunnukset ja luetteloimaan ne, mutta semantiikkaa määrittämistä ei pystytä määrittämään pelkän tunnuksen perusteella. Täytyy olla tiedossa käytetty merkintätapa. Osa säätökaavioista sisältää laite- ja/tai pisteluettelon, johon on usein merkitty laitteen tarkoitus. Tunnistamalla tältä sivulta sama laitetunnus, kuin kaaviossa, ja analysoimalla siihen liitettyä tekstiä, voidaan liittää tunnukselle sille tarkoitettu toiminallisuus. Tekstintunnistaminen on mahdollista koneoppimisen keinoin. Case-tutkimuksessa tätä ei tutkittu.

8.3 Ohjelmakirjaston käytön kehittäminen

Ohjelmakirjastoa ei ole kehitetty koneoppimisen hyödyntämistä huomioiden. Tämä näkyy siinä, että lähes samoista järjestelmistä on useita eri mallipohjia. Mallipohjat eroavat toisistaan vain marginaalisesti. Eroavaisuus saattaa olla niin pieni, että pelkästään PI-kaaviota tulkitessa eroa ei ole. Tällaisten mallipohjien tunnistaminen kuvantunnistuksella on lähtökohtaisesti mahdotonta. Koneoppimissovelluksen näkökulmasta nämä mallipohjat tulisi yhdistää ja niissä olevat eroavaisuudet tulisi olla parametrein muokattavissa.

Case-tutkimuksen mukaisen koneoppimissovelluksen hyöty rakennusautomaatioprojektissa on hyvin pieni. Käyttäjiä ohjelmalla olisi hyvin rajoitettu joukko. Suomen mittakaavassa käyttäjiä on muutamia satoja maksimissaan. Teoreettisesti voidaan ajatella, että ajansäästö ohjelmoitavaa järjestelmää kohden on muutaman minuutin luokkaa. Toimivan sovelluksen aikaansaamiseksi resursseja tulisi sitoa projektiin merkittävästi. Näin pienelle käyttäjäkunnalle suunnattuun ohjelmaan ei ole välttämättä perusteltua käyttää rajallisia resursseja. Tuotosten laatua ja yhdenmukaisuutta pystytään parantamaan myös ilman koneoppimista. Käytössä oleville resursseille saadaan parempi vastine ohjaamalla niitä muun muassa seuraaviin kehityshankkeisiin.

Mallipohjien vakiovalinnat olisi hyvä muuttaa enemmistösäännön mukaisiksi. Näin tarvittavat muutokset mallipohjan käytössä keskimäärin vähenevät. Mallipohjien käyttöä tulisi tilastoida. Tilastointi tulisi tapahtua automaattisesti mallipohjia käytettäessä. Tämä vaatii tilastointimoduulin rakentamisen pääohjelmaan. Tilastoitavia asioita ovat mallipohja ja siihen liitetyt ryhmävalinnat. Mallipohjien vakiovalintoja voidaan muuttaa tilastojen pohjalta määräajoin.

Mallipohjien ryhmien nimet ja toiminallisuus tulisi yhdenmukaistaa ja järkeistää. Ryhmien nimet ovat tällä hetkellä hyvin vaihtelevia ja sama toiminallisuus voi olla eri nimellä mallipohjien välillä. Ryhmien nimien tulisi lisäksi olla kuvaavia, mutta lyhyitä. Järkevät ja yhdenmukaiset ryhmätunnukset mahdollistavat helpommin lähestyttävän dokumentaation laadinnan.

Keskeisin kehitysaihe mallipohjien käytössä on pääohjelman käyttöliittymän kehittäminen. Käyttöliittymän käytettävyyteen tulee kiinnittää aikaisempaa enemmän huomiota. Tarvittavat muutokset ovat pieniä, mutta niillä voidaan saavuttaa lähes sama ajansäästö kuin koneoppimisella. Lisäksi hyvin toimiva käyttöliittymä lisää mallikirjaston käytön houkuttelevuutta.

LÄHTEET

Aghdam, H. H. Heravi, E. J. 2017. Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification. Springer. Viitattu 15.8.2021. Vaatii käyttöoikeuden. https://masterworkshop.skillport.com/skillportfe/assetSummaryPage.action?assetid=RW$3091:_ss_book:138106#summary/BOOKS/RW$3091:_ss_book:138106

Alfarraj, M. Alregib, G. 2018. Petrophysical-property estimation from seismic data using recurrent neural networks. SEG Technical Program Expanded Abstracts 2018. Viitattu 11.8.2021. Vaatii käyttöoikeuden. https://doi.org/10.1190/segam2018-2995752.1

Arroyo, E. Hoernicke, M. Rodríguez, P. Fay, A. 2016. Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams. Computers & Chemical Engineering, Volume 92, 2016, Pages 112-132. Viitattu 27.11.2021. https://doi.org/10.1016/j.compchemeng.2016.04.0

Asaithambi, S. 2018. Why, How and When to apply Feature Selection. toward data science blogi 31.1.2018. Viitattu 6.8.2021. https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2

Basha, S.H.S. Vinakota, S. K. Dubey, S. R. Pulabaigari, V. Mukherjee, S. 2020. AutoFCL: Automatically Tuning Fully Connected Layers for Handling Small Dataset. Neural Comput & Applic 33, 8055–8065 (2021). Viitattu 10.11.2021. https://arxiv.org/abs/2001.11951

Bonaccorso, G. 2018. Machine Learning Algorithms – Second Edition. Packt Publishing. Viitattu 26.7.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/machine-learning-algorithms/9781789347999/

Brownlee, J. 2017. How Much Training Data is Required for Machine Learning? Machine Learning Mastery blog 24.7.2017. Viitattu 9.11.2021. https://machinelearningmastery.com/much-training-data-required-machine-learning/

Brownlee, J. 2019a. How Do Convolutional Layers Work in Deep Learning Neural Networks? Machine Learning Mastery blog 17.4.2019. Viitattu 24.11.2021. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Brownlee, J. 2019b. How to use Learning Curves to Diagnose Machine Learning Model Performance. Machine Learning Mastery blog 27.2.2019. Viitattu 21.11.2021. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Chappell, D. 2015. Introducing Azure Machine Learning. Chappell & Associates. Viitattu 10.11.2021. http://download.microsoft.com/download/3/B/9/3B9FBA69-8AAD-4707-830F-6C70A545C389/Introducing_Azure_Machine_Learning.pdf

Cogito. 2019. How Much Training Data is Required for Machine Learning Algorithms? Cogito Tech blog 7.10.2019. Viitattu 9.11.2021. https://www.cogitotech.com/blog/how-much-training-data-is-required-for-machine-learning-algorithms

Costa, V. C. 2019. Understanding the Structure of a CNN. medium.com blogi 18.4.2019. Viitattu 15.8.2021. https://viniciuscantocosta.medium.com/understanding-the-structure-of-a-cnn-b220148e2ac4

Dabbura, I. 2017. Gradient Descent Algorithm and Its Variants. toward data science blogi 21.12.2017. Viitattu 22.8.2021. https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Deng, J. Wang, H. 2018. Modeling and Optimizing Building HVAC Energy Systems Using Deep Neural Networks. 2018 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), 2018, pp. 181-185. Viitattu 20.11.2021. https://doi.org/10.1109/ICSGCE.2018.8556684

Dertat, A. 2017. Applied Deep Learning - Part 4: Convolutional Neural Networks. toward data science blogi 8.11.2017. Viitattu 15.8.2021. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Devasia, A. 2021. An Overview of IEC 61131-3 in Industrial Automation System. tekninen artikkeli 25.3.2021 Control Automation. Viitattu 5.11.2021. https://control.com/technical-articles/an-overview-of-iec-61131-3-Industrial-Automation-Systems/

Donges, N. A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks. built in blogi 29.7.2021. Viitattu 11.8.2021. https://builtin.com/data-science/recurrent-neural-networks-and-lstm

Du K. & Swamy M.N.S. 2014. Neural Network and Statistical Learning. Springer. Viitattu 10.8.2021. Vaatii käyttöoikeuden. https://doi.org/10.1007/978-1-4471-5571-3

Elements of AI. n.d. Reaktor & Helsingin yliopisto. Verkkosivu. Viitattu 22.7.2021. https://www.elementsofai.com/

Esposito, D. & Esposito, F. 2020. Introducing Machine Learning. Microsoft Press. Viitattu 20.7.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/introducing-machine-learning/9780135588338/

Fidelix. 2021. Template-kirjaston säätökaaviot. Julkaistu yrityksen sisäiseen käyttöön. Opinnäytetyön tekijän hallussa.

Fidelix. n.d. Fidelix esittely. Verkkosivu. Viitattu 12.8.2021. https://www.fidelix.fi/

Girshick, R. Donahue, J. Darrel, T. Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580–587). Viitattu 25.11.2021. https://arxiv.org/abs/1311.2524

Hayduk, G. Kwasnowski, P. Mikòs, Z. 2016. Building management system architecture for large building automation systems. IEEE. Viitattu 30.7.2021. Vaatii käyttöoikeuden. DOI: 10.1109/CarpathianCC.2016.7501100

He, K. Zhang, X. Ren, S. Sun, J. 2015. Deep Residual Learning for Image Recognition. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778. Viitattu 29.11.2021. https://arxiv.org/abs/1512.03385

Hosmer, D. W. Jr. Lemeshow, S. Sturdivant, R. X. 2013. Applied Logistic Regression, 3rd Edition. Wiley. Viitattu 21.8.2021. Vaatii käyttöoikeiden. https://learning.oreilly.com/library/view/applied-logistic-regression/9781118548356/

Hu, J. 2016. Discriminative Transfer Learning with Sparsity Regularization for Single-Sample Face Recognition. Image and Vision Computing. 60. Viitattu 10.11.2021. https://doi.org/10.1016/j.imavis.2016.08.007

Huang, G. Liu, Z. Van Der Maaten, L. Weinberger, K. Q. 2017. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 4700-4708. Viitattu 29.11.2021. https://arxiv.org/abs/1608.06993

Hulstaert, L. 2018. A Beginner's Guide to Object Detection. DataCamp tutorial 19.4.2018. Viitattu 23.11.2021. https://www.datacamp.com/community/tutorials/object-detection-guide

IBM. 2020. What are convolutional neural networks? IBM Cloud Education 20.10.2020. Viitattu 15.8.2021. https://www.ibm.com/cloud/learn/convolutional-neural-networks

Jiao, L. Zhang, F. Liu, F. Yang, S. Li, L. Feng, Z. Qu, R. 2019. A Survey of Deep Learning-based Object Detection. IEEE access, 7, 128837–128868. Viitattu 25.11.2021. https://arxiv.org/abs/1907.09408

John, K. H. Tiegelkamp, M. 2021. IEC 61131-3: Programming Industrial Automation Systems 2nd Edition. Springer. Viitattu 5.11.2021. Vaatii käyttöoikeuden. https://doi.org/10.1007/978-3-662-07847-1

Jung, A. 2021. Machine Learning: Basic Principles. Viitattu 27.7.2021. https://arxiv.org/abs/1805.05052

Kadam, S. 2020a. CNN Series Part 1: How do computers see images? Analytics Vidhya artikkeli 7.7.2020. Viitattu 23.11.2021. https://medium.com/analytics-vidhya/cnn-series-part-1-how-do-computers-see-images-32462a0b33ca

Kadam, S. 2020b. CNN Series Part 2: What is meant by Convolution? Analytics Vidhya artikkeli 20.7.2020. Viitattu 23.11.2021. https://shwetarkadam25.medium.com/cnn-series-part-2-what-is-meant-by-convolution-c504f38c42b

Kang, S.-O. Lee, E.-B. Baek, H.-K. 2019. A Digitization and Conversion Tool for Imaged Drawings to Intelligent Piping and Instrumentation Diagrams (P&ID). Energies 2019, 12, 2593. Viitattu 27.11.2021. https://doi.org/10.3390/en12132593

Kapur, S. 2017. Computer Vision with Python 3. Packt Publishing. Viitattu 23.11.2021. Vaatii käyttöoikeiden. https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=4987525#

Kim, H. Lee, W. Kim, M. Moon, Y. Lee, T. Cho, M. Mun, D. 2021. Deep-learning-based recognition of symbols and texts at an industrially applicable level from images of high-density piping and instrumentation diagrams. Expert Systems with Applications, Volume 183, 2021, 115337. Viitattu 27.11.2021. https://doi.org/10.1016/j.eswa.2021.115337

Kusiak, A. Li, M. Tang, F. 2010. Modeling and optimization of HVAC energy consumption. Applied Energy Volume 87, Issue 10, October 2010, Pages 3092-3102. Viitattu 20.11.2021. https://doi.org/10.1016/j.apenergy.2010.04.008

Lapan, M. 2020. Deep Reinforcement Learning Hands-On - Second Edition. Packt Publishing. Viitattu 5.8.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/deep-reinforcement-learning/9781838826994/

Li, S. Ren, S. Wang, X. 2013. HVAC Room Temperature Prediction Control Based on Neural Network Model. 2013 Fifth International Conference on Measuring Technology and Mechatronics Automation, 2013, pp. 606-609. doi: 10.1109/ICMTMA.2013.151

Liedes, R. Härkönen, P., Mikkola, J. Piikkilä, V., Pusa, K., Sahala, A., Sahlstén, T., Sandström, B., Sirviö, A., Spangar, T., Sulku, J. 2018. Rakennusautomaatiojärjestelmät. ST-Käsikirja 17. 6. uudistettu painos. Espoo: Sähkötieto ry

Manning, C. D. Raghaven, P. Schütze, H. 2008. Introduction to Information Retrieval. Cambridge University Press. Viitattu 25.11.2021. https://doi.org/10.1017/CBO9780511809071

Marsland, S. 2014. Machine Learning, 2nd Edition. Chapman and Hall/CRC. Viitattu 22.7.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/machine-learning-2nd/9781466583283/

McAllester, D. A. 1999. Some PAC-Bayesian Theorems. Machine Learning volume 37, pages355–363 (1999). Viitattu 9.10.2021 https://doi.org/10.1023/A:1007618624809

Murphy, K P. 2012. Machine Learning: A Probabilistic Perspective. MIT Press. Viitattu 22.7.2021. Vaatii käyttöoikeuden. https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=3339490

Michelucci, U. 2019. Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection. Apress. Viitattu 15.8.2019. Vaatii käyttöoikeuden. https://doi.org/10.1007/978-1-4842-4976-5_6

Microsoft. n.d. Azure Machine Learning documentation. Viitattu 22.11.2021. https://docs.microsoft.com/fi-fi/azure/machine-learning/

Microsoft. n.d. Custom Vision documentation. Viitattu 22.11.2021. https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/

Mishra, S. Glaws, A. Cutler, D. Frank, S. Azam, M. Mohammadi, F. Venne, J-S. 2020. Unified architecture for data-driven metadata tagging of building automation systems. Automation in Construction Volume 120, December 2020, 103411. Viitattu 20.11.2021. https://doi.org/10.1016/j.autcon.2020.103411

Moradzadeh, A., Mansour-Saatloo, A., Mohammadi-Ivatloo, B., & Anvari-Moghaddam, A. 2020. Performance evaluation of two machine learning techniques in heating and cooling loads forecasting of residential buildings. Applied Sciences, 10(11), 3829. Viitattu 20.11.2021. http://dx.doi.org/10.3390/app10113829

Muralidhar, KSV. 2021. Learning Curve to identify Overfitting and Underfitting in Machine Learning. towards data science artikkeli 9.2.2021. Viitattu 21.11.2021. https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5

Ng, H-W. Nguyen, V. D. Vonikakis, V. Winkler, S. 2015. Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning. ICMI '15: Proceedings of the 2015 ACM on International Conference on Multimodal Interaction November 2015, Pages 443-. Viitattu 10.11.2021. https://doi.org/10.1145/2818346.2830593

Nurminen, J. K. Rainio, K. Numminen, J-P. Syrjänen, T. Paganus, N. Honkoila, K. 2019. International Conference on Computer Recognition Systems. Springer, Cham, 2019. p. 27-36. Viitattu 27.11.2021. https://tuhat.helsinki.fi/ws/portalfiles/portal/128457068/Object_Detection_in_Design_Diagrams_with_ML.pdf

Osowski, S. Siwek, K. Markiewixz, T. 2004. MLP and SVM network – a Comparative study. Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004, 2004, p.37-40. Viitattu 10.8.2021. Vaatii käyttöoikeuden. DOI: 10.1109/TMAG.2010.2043657

Perlich, C. 2010. Learning Curves in Machine Learning. Encyclopedia of Machine Learning, pages 577–488. Springer. Viitattu 22.11.2021. http://dx.doi.org/10.1007/978-0-387-30164-8_452

Raydan, K. 2021. An Intuitive Guide to Convolutional Neural Networks. artikkeli 22.7.2021. Viitattu 24.11.2021. https://medium.com/@kamalraydan1/an-intuitive-guide-on-convolutional-neural-networks-cnns-9efaded5b2dc

Read, J. Pfahringer, B. Holmen, G. Frank, E. 2011. Classifier chains for multi-label classification. Mach Learn 85, 333 (2011). Viitattu 30.11.2021. https://doi.org/10.1007/s10994-011-5256-5

Rudin, W. 1976. Principles of Mathematical Analysis 3. edition. McGraw-Hill

Runkler, T A. 2012. Data Analytics: Models and Algorithms for Intelligent Data Analysis. Springer Vieweg. Viitattu 2.8.2021. Vaatii käyttöoikeuden. DOI 10.1007/978-3-8348-2589-6

Russakovsky, O. Deng, J. Su, H. Krause, J. Satheesh, S. Ma, S. Huang, Z. Karpathy, A. Khosla, A. Bernstein, M. Berg, A. C. Fei-Fei, L. 2014. ImageNet Large Scale Visual Recognition Challenge. Viitattu 23.11.2021. https://arxiv.org/abs/1409.0575

Russell, S. Norvig, P. 2016. Artificial intelligence a modern approach 3rd edition. Pearson. Viitattu 18.8.2021. Vaatii käyttöoikeuden. https://www.vlebooks.com/Vleweb/Product/Index/861765?page=0

Sarkis, A. 2020. Training Data for Machine Learning. O'Reilly Media, Inc. Viitattu 9.11.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/training-data-for/9781492094517/

Scikit-learn. n.d. User Guide. Viitattu 29.11.2021. https://scikit-learn.org/stable/user_guide.html

Sewak, M. Karim, Md. R. Pujari, P. 2018. Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python. Packt Publishing. Viitattu 15.8.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/practical-convolutional-neural/9781788392303/

Sugiyama, M. 2015. Statistical Reinforcement Learning. CRC Press. Viitattu 5.8.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/statistical-reinforcement-learning/9781439856895/

Sutton, R. Barto, A. 2016. Reinforcement Learning: An Introduction Second editon, in progress. The MIT Press. Viitattu 5.8.2021. https://web.stanford.edu/class/psych209/Readings/SuttonBartoRL.pdf

Tabian, I. Fu, H. Sharif Khodaei, Z. 2019. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors 2019, 19, 4933. Viitattu 24.11.2019. https://doi.org/10.3390/s19224933

Tan, W. C. Chen, I-M. Tan, H. K. 2016. Automated identification of components in raster piping and instrumentation diagram with minimal pre-processing. 2016 IEEE International Conference on Automation Science and Engineering (CASE), 2016, pp. 1301-1306. Viitattu 27.11.2021. Vaatii käyttöoikeuden. https://doi.org/10.1109/COASE.2016.7743558

Trask, Mr. 2018. General vs Narrow AI. Hackermoon blogi 2.6.2018. Viitattu 20.7.2021. https://hackernoon.com/generalvs-narrow-ai-3d0d02ef3e28

Tuominen, H. & Neittaanmäki, P. 2019. Tekoälyn perusteita ja sovelluksia. Informaatioteknologian tiedekunta Jyväskylän yliopisto. Pdf-dokumentti. Viitattu 20.7.2021. http://urn.fi/URN:ISBN:978-951-39-7796-2

Wang, W. Brambley, M. R. Kim, W. Somasundaram, S. Stevens, A. J. 2018. Automated point mapping for building control systems: Recent advances and future research needs. Automation in Construction Volume 85, January 2018, Pages 107-123. Viitattu. 20.11.2021. https://doi.org/10.1016/j.autcon.2017.09.013

Witten, I. H. Frank, E. Hall, M. A. Pal, C. J. 2017. Data mining practical machine learning tools and techniques, 4th edition. Morgan Kaufmann. Viitattu 16.8.2021. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/data-mining-4th/9780128043578/

Wortmann, T. Waibel, C. Nannicini, G. Evins, R. Schroepfer, T. Carmeliet, J. 2017. Are Genetic Algorithms Really the Best Choice for Building Energy Optimization? Viitattu 20.11.2021. http://dx.doi.org/10.22360/simaud.2017.simaud.006

Yang, S. Wan, M. P. Chen, W. Ng, B. F. Dubey, S. 2020. Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization. Applied Energy Volume 271, 1 August 2020, 115147. Viitattu 20.11.2021. https://doi.org/10.1016/j.apenergy.2020.115147

Ying, X. 2019. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series, Volume 1168, Issue 2. Viitattu 23.11.2021. https://doi.org/10.1088/1742-6596/1168/2/022022

Yohanandan, S. 2020. mAP (mean Average Precision) might confuse you. toward data science blog 9.6.2020. Viitattu 25.11.2021. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

Yu, E. Jae-Min Cha. Lee, T. Kim, J. Mun, D. 2019. Features recognition from piping and instrumentation diagrams in image format using a deep learning network. Energies, 12(23), 4425. Viitattu 27.11.2021. http://dx.doi.org/10.3390/en12234425

Zanaty, E.A. 2012. Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. Egyptian informatics journal, 2012-11, Vol.13 (3), p.177-183. Viitattu 10.8.2021. https://doi.org/10.1016/j.eij.2012.08.002

Zeng, N. 2018. An Introduction to Evaluation Metrics for Object Detection. blog artikkeli 16.12.2018. Viitattu 25.11.2021. https://blog.zeng-gyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/

Zhu, X. Vondrick, C. Fowlkes, C. C. & Ramanan, D. 2015. Do We Need More Training Data? International Journal of Computer Vision volume 119, pages76–92 (2016). DOI https://doi.org/10.1007/s11263-015-0812-2

LIITTEET

Liite 1. FX-editor / Template Manager ML concept