Tampereen yliopisto

Jimi Eklund

KONEOPPIMINEN ELINTARVIKETEOLLISUUDEN KONENÄKÖSOVELLUKSESSA

Kandidaatintyö
Tekniikan ja luonnontieteiden tiedekunta
Tarkastaja: TkT Niko Siltala
Huhtikuu 2021

TIIVISTELMÄ

Jimi Eklund: Koneoppiminen elintarviketeollisuuden konenäkösovelluksessa
Kandidaatintyö
Tampereen yliopisto
Konetekniikan kandidaatin tutkinto-ohjelma
Huhtikuu 2021

Koneoppimisen hyödyntäminen teollisuuden konenäkösovelluksissa yleistyy tavoiteltaessa parempia tuloksia laaduntarkkailussa. Koneoppimisella korvataan aikaisemmin joko ihmisvoimin suoritettua laaduntarkastusta tai joustamattomilla ennalta määritellyillä arvoilla luokittelua. Työn tavoitteena on selvittää koneoppimisvaihtoehtoja yhteen elintarviketeollisuuden konenäkösovellukseen ja tehdä alustava ehdotus käytännön aloituspisteeksi. Sovelluksen toteutusalustana on LabVIEW-ympäristö ja kaikki käsitellyt vaihtoehdot ovat tällä toteutusalustalla toimivia. Työ suoritetaan kirjallisuuskatsauksena.

Työssä esiteltiin ensin käsiteltävä järjestelmä ja vaatimukset koneoppimissovelluksen toiminnalle. Tärkeää sovelluksen toiminnan kannalta on riittävä luokittelutarkkuus. Lisäksi tavoitteena olisi saada kuvien luokitteluun käytettävä sovellus toimimaan pelkästään ehjiä kappaleita sisältävän opetusmateriaalin avulla.

Seuraavaksi työssä käydään läpi yleisesti koneoppimista ja siihen liittyviä ongelmia, sekä koneoppimisen ja konenäön yhdistämistä kaksiulotteisissa (2D) ja kolmiulotteisissa (3D) tapauksissa. Koneoppimisen yleisin ongelma on ylisovittuminen, jolloin käytetty algoritmi toimii opetusaineistolla moitteettomasti, mutta käytännössä toiminta on heikkoa. Koneoppimisalgoritmeilla ei yleensä käsitellä kuvia ilman esiprosessointia ja piirre-erottelua. Esiprosessoinnissa käsiteltävää kuvaa muokataan niin, että jäljelle jää vain olennainen informaatio. Piirre-erottelussa kuva muunnetaan muotoon, jonka koneoppimisalgoritmi pystyy käsittelemään. Kaksi- ja kolmiulotteisten tapausten käsittelyyn vaaditaan erilliset algoritmit.

Kun koneoppimisen ja konenäön yhdistäminen on käsitelty, selvitetään kuvien luokitteluun soveltuvia koneoppimisalgoritmeja ja menetelmiä. Esiteltyjä algoritmeja ovat tukivektorikoneet, konvoluutioneuroverkot ja päätöspuut. Erityisesti konvoluutioneuroverkot ovat saavuttaneet suosiota kuvien luokittelussa. Algoritmien lisäksi esitellään luokittelijayhdistämistä, jonka avulla useampia algoritmeja voidaan käyttää yhden tehtävän ratkaisuun.

Lopulta päädyttiin tukivektorikoneiden käyttöön 2D-tiedonkäsittelyssä ja konvoluutioneuroverkkoihin 3D-tiedonkäsittelyssä. Algoritmien valinta perustuu kirjallisuudesta löydettyihin esimerkkeihin ja algoritmien ominaisuuksiin.

Avainsanat: koneoppiminen, konenäkö, tiedonkäsittely, ylisovittuminen, tukivektorikone, konvoluutioneuroverkko, päätöspuu

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

1. JOHDANTO ... 1
2. KÄSITELTÄVÄT KAPPALEET JA JÄRJESTELMÄ ... 3
2.1 Nykyinen järjestelmä ... 3
2.2 Kappaleet ja niistä tarkastettavat piirteet ... 4
2.3 Sovelluksen vaatimukset ... 5
3. KONEOPPIMINEN KONENÄKÖSOVELLUKSESSA ... 7
3.1 Koneoppimisen ja konenäön yhdistäminen ... 7
3.2 2D-tiedonkäsittely ja koneoppiminen ... 8
3.3 3D-tiedonkäsittely ja koneoppiminen ... 10
4. KONENÄKÖÖN SOVELTUVIA ALGORITMEJA JA MENETELMIÄ ... 13
4.1 Koneoppimisalgoritmeista ... 13
4.2 Tukivektorikoneet ... 14
4.3 Konvoluutioneuroverkot ... 15
4.4 Päätöspuut ... 18
4.5 Luokittelijayhdistelmät ... 19
5. SOVELLUS ... 20
5.1 Algoritmien vertailua ... 21
5.2 Johtopäätökset ... 23
5.2.1 2D-algoritmi ... 23
5.2.2 3D-algoritmi ... 24
6. YHTEENVETO ... 25
LÄHTEET ... 26

1. JOHDANTO

Konenäköä käytetään elintarviketeollisuudessa tuotteiden laadun varmistamiseen. Kontrolloiduissa ympäristöissä, joissa kappaleet ja niiden ympäristö ovat vakaita, konenäköä on perinteisesti käytetty ennalta määriteltyjen laatuparametrien, kuten leveyden, pituuden ja pinta-alan tarkastamiseen. Ympäristön ja kappaleiden ominaisuuksien tai orientaation ollessa vaihtelevia pelkällä konenäöllä suoritettavasta laaduntarkkailusta tulee hankalaa. Tietotekniikan kehitys tarjoaa mahdollisuuksia yhdistää konenäkö ja -oppiminen. Konenäön- ja oppimisen yhdistäminen on jo pidemmän aikaa ollut trendinä elintarviketeollisuudessa. Koneoppiminen mahdollistaa tuotteiden nopeamman ja tarkemman laaduntarkkailun.

Konenäön ja koneoppimisen yhdistäminen tuo laaduntarkkailujärjestelmän joustavuutta. Koneoppimisella pyritään luomaan yleistäviä malleja tunnistettavista piirteistä ja mallien avulla tunnistaa opetusmateriaalin ulkopuolelta tulevia syötteitä. Pelkällä konenäöllä on vaikeaa tunnistaa ennalta määrättyihin parametreihin kuulumattomia poikkeamia, kuten lohkeamia, halkeamia tai väärää muotoa jne. Algoritmeista ja sovellustavasta riippuen on mahdollista luoda monia erilaisia järjestelmiä, joita voidaan testata ja hyväksyä tai hylätä tavoitteeseen pääsemisen perusteella. Järjestelmiä ja algoritmeja voidaan myös yhdistellä.

Koneoppimisen avulla automatisoitu prosessi tuo inhimillisten virheiden ja koneiden tekemien virheiden lisäksi algoritmeista ja ohjelman opetuksesta johtuvat virheet. Toiveena vastaavissa sovelluksissa on, että ohjelman virheiden määrä on vähäisempi ja korjaustoimenpiteillä helpommin säädettävissä, kuin ihmisillä tai pelkillä konenäkösovelluksilla. Al-Audah et al. (2012) käyttivät rekisterikilpien tunnistamiseen automatisoitua koneoppimis- ja näköprosessia, jonka onnistumisprosentti ja nopeus ovat paljon suurempia kuin ihmispohjaisilla ratkaisuilla olisi mahdollista.

Konenäön ja koneoppimisen yhdistäminen laaduntarkkailussa on nykyisin yleinen tapa automatisoida ja optimoida prosessi. Konenäkö ja koneoppimista on sovellettu elintarviketeollisuudessa nesteiden epäpuhtauksien tarkkailuun (Li et

al. 2018), lohenlihan jäätymisen tunnistamiseen (Xu & Sun 2016) ja oluen laaduntarkkailuun (Gonzalez Viejo et al. 2016). Tässä työssä selvitetään koneoppimista hyödyntäviä algoritmeja ja menetelmiä liukuhihnalla kulkevien kappaleiden laaduntarkkailuun konenäkösovelluksen avulla. Selvitettyjen algoritmien ja menetelmien avulla annetaan myös ehdotus käytännön toteutuksen aloituspisteeksi. Kuvien muodostuminen kameralta rajautuu tämän työn ulkopuolelle.

Työssä tarkoituksena on keskittyä vain National Instrumentsin LabVIEW-ohjelmointiympäristössä toimiviin sovelluksiin. Ohjelmointialusta kykenee suoraviivaisesti yhdistämään koneoppimisen ja konenäön (Getting Started with LabVIEW, 2012). LabViewillä on mahdollista käyttää yleisimpiä koneoppimisalgoritmeja, joten yhteen ohjelmointikieleen keskittyminen ei ole rajoittava tekijä. Työ suoritetaan kirjallisuuskatsauksena, jonka pohjalta lopulta esitetään teoriapohjalla mahdollista käytännön ratkaisua.

Työssä käydään ensin läpi käsiteltävä järjestelmä, kappaleet ja sovelluksen vaatimukset. Tämän jälkeen käsitellään koneoppimisen ja konenäön yhdistämistä ja kaksi- ja kolmiulotteisen datan käsittelyä. Seuraavaksi käydään läpi konenäköön LabVIEW-ympäristössä sopivia algoritmeja ja menetelmiä. Esittelyssä on kolme erilaista algoritmia ja menetelmiä niiden yhdistämiseen. Viimeisenä käydään läpi johtopäätöksiä koneoppimissovelluksen käytännön toteuttamisen aloituspisteestä.

2. KÄSITELTÄVÄT KAPPALEET JA JÄRJESTELMÄ

Tässä luvussa esitellään käsiteltävä järjestelmä ja määritellään vaatimukset koneoppimissovelluksen toiminnalle.

2.1 Nykyinen järjestelmä

Nykyinen järjestelmä ottaa tietyn aikavälin välein kuvia liukuhihnalla kulkevista kappaleista, ja laskee kuvien perusteella jokaisen kappaleen pinta-alan, leveyden ja pituuden. Kuvaukseen käytetään suoraan liukuhihnan yläpuolella olevaa kameraa, joka ottaa kuvia niin, että jokainen liukuhihnalla kulkeva kappale esiintyy kokonaisena ainakin yhdessä kuvassa. Kappaleita kulkee kuvausaseman läpi kymmeniä kappaleita sekunnissa ja sovelluksen täytyy saada kaikki kappaleet tarkastettua.

Järjestelmän ympäristö on pölyinen, mutta stabiili. Kameran linssin pölyisyys voi vaihdella, kun linssi likaantuu käytössä ja sitä puhdistetaan. Pölyisyys voi äärimmäistapauksissa aiheuttaa ongelmia kuvien selkeyden kanssa. Järjestelmän valaistus ja lämpötila pysyvät lähes muuttumattomina, joten muita ulkoisia haittatekijöitä ei ole.

Järjestelmä ei kykene tunnistamaan ja hylkäämään kappaleita, joissa on halkeamia, muita hylkäysperusteita tai jos kappaleet ovat päällekkäin, vaan hyväksyy tai hylkää kappaleet pelkästään edellä mainittujen laskettavien arvojen perusteella. Tellaeche ja Arana (2011) käsittelevät hyvin samanlaista systeemiä ja monet listatuista vikatekijöistä ovat samanlaisia kuin tässä työssä käsiteltävät. Järjestelmien erona on kuitenkin, että Tellaechen ja Arananan tapauksessa näytekappaleiden määrä yhtä kuvaa kohden on paljon suurempi kuin käsiteltävässä tapauksessa (n. 130 vs. 10-20).

Nykyisessä järjestelmässä laaduntarkkailukamera ottaa linjalla kulkevista kappaleista kuvia 10 millisekunnin välein. Sovelluksen hylkäykseen tai hyväksymiseen johtavat laskelmat täytyy suorittaa niin, että kameran ottaessa uuden kuvan on edellisen kuvan kaikista kokonaisista kappaleista tehty joko hyväksytty tai hylätty päätös. Jos järjestelmä ei kykene luokittelemaan jokaisen kuvan kokonaisia kappaleita, on mahdollista, että normaalisti hylkäykseen meneviä kappaleita voi päästä linjaston läpi.

2.2 Kappaleet ja niistä tarkastettavat piirteet

Koneoppimisen avulla pitäisi pystyä tarkastamaan kappaleen eheys, koko, reunojen kääntyneisyys, sekä kappaleen vääntyneisyys. Kappaleiden päämitat vaihtelevat välillä 20-50 mm, ja korkeus vaihtelee välillä 3-20 mm. Samaa erää olevien kappaleiden päämittojen vaihteluväli on kuitenkin pienempi. Väri voi riippuen raaka-aineesta olla vaalea tai tumma ja yksittäisten samaa erää olevien tuotteiden väri voi vaihdella lievästi.

Kappaleet ovat eheitä, silloin kun niissä ei ole reikiä tai halkeamia, ja reunat ovat muutenkin yhtenäiset koko matkalta. Reunoista pitäisi pystyä myös tunnistamaan, että ne on leikattu kunnolla. Tämä tarkoittaa, että kappaleen reunat on leikattu siististi, eikä niissä saa esiintyä rosoisuutta. Kappaleen koko ja oikeanlainen muoto riippuvat kappaleittain ja ohjelma täytyisi kouluttaa jokaiselle kappaletyypille erikseen. Lähtökohtaisesti jokaisen kappaletyypin pinta-ala heittelee +-10 % nimellisarvosta. Tärkeää systeemin toimivuuden kannalta olisi, että ohjelman toiminnan vaihtaminen nykyhetkellä valmistettavalle kappaleelle sopivaksi on helppoa.

1. Kappaleessa on reikä
2. Kappaleessa on halkeama
3. Kappaleessa on lohkeama
4. Kappaleen reunat eivät ole leikkaantuneet kunnolla
5. Kappale on väärän kokoinen

6. Kappale on väärän muotoinen
7. Kappale on liian vääntynyt
8. Kappaleen reunat ovat liian kääntyneet

Yllä on tiivistetysti listattuna kaikki mahdolliset laadulliset poikkeamat. Näistä yleisimpiä ovat kappaleiden kokoon ja muotoon liittyvät viat.

Käsiteltävät kappaleet voivat olla kaksiulotteiselta muodoltaan ympyröitä, ellipsejä, kolmioita tai lähes mitä vaan muita tasossa säännöllisiä kappaleita. kolmiulotteiselta muodoltaan ne taas voivat olla levymäisiä, kuperia tai koveria. Pääosa kappaleiden mahdollisista hylkäysvaihtoehdoista voidaan tunnistaa 2D-muodon perusteella, mutta 3D-muotoa tarvitaan kappaleen reunojen kääntyneisyyden ja kappaleen vääntymisen tarkastamiseen. Pyrkimyksenä on löytää ratkaisu, joka pystyy useiden eri 2D ja 3D-muotojen tunnistamiseen.

2.3 Sovelluksen vaatimukset

Suurin rajoittava tekijä sovellukseen sopivia algoritmeja valittaessa on niiden toimintanopeus. Algoritmien tarkkaa nopeutta tietyissä sovelluksissa ei voida tietää ennen kokeellista testausta, mutta suuntaa antavia hypoteesejä on mahdollista tehdä. Käsiteltävien kappaleiden yksinkertaisten geometristen muotojen ja taustan ja kappaleiden selvän erotettavuuden takia on mahdollista käyttää hyvin monenlaisia algoritmeja.

Sovelluksen täytyy optimitilanteessa tunnistaa hylättävät kappaleet opetusmateriaalin avulla, jossa esiintyy vain eheitä kappaleita. Tähän tavoitteeseen pääseminen vaatii, että koneoppimisalgoritmi pystyisi otettujen kuvien perusteella tunnistamaan kappaleiden 2D-tason geometrisien muotojen yhtenäisyyden. Jos algoritmia ei saada toimimaan pelkästään eheitä kappaleita sisältävällä opetusmateriaalilla, täytyy opetusmateriaalin joukkoon lisätä myös kuvia, joissa on viallisia kappaleita. Esimerkiksi lohkeamia, halkeamia ja rosoisia reunoja sisältäviä kappaleita. Jos opetusmateriaalin lisätään rikkinäisiäkin kappaleita sisältäviä kuvia, täytyy aineisto käsitellä niin, että käytettävä algoritmi tietää opetusvaiheessa missä kuvissa on ehjiä ja rikkinäisiä kappaleita ja missä vain ehjiä.

Kappaleiden käsittely 3D-tasossa ja tapauksen vaatima erillinen algoritmi hidastaa sovelluksen toimintaa ja on mahdollista, että 3D-muotojen tunnistus hidastaa sovelluksen toimintaa liikaa, jolloin järjestelmä ei kykene toimimaan nopeusvaatimuksen mukaisesti. Vaihtoehtoina tässä tapauksessa ovat uusien algoritmien tai niiden yhdistelmien kokeilu tai kappaleiden 3D-piirteiden tarkastamatta jättäminen.

Koneoppimista hyödyntävän laaduntarkkailusovelluksen täytyisi olla tarkempi tai kustannustehokkaampi, kuin alkuperäinen pelkkään konenäköön perustuva sovellus. Jos sovellus toimisi huonommin kuin pelkkä konenäkösovellus, ei sovelluksen implementointi olisi järkevää. Nykyisen järjestelmän kykenemättömyys tiettyjen vikaluokkien tunnistamiseen tarkoittaa, että jos koneoppimissovellus saadaan tunnistamaan enemmän vikaluokkia aikaisempaa tarkkuutta vastaavalla tarkkuudella, se on tehokkaampi ja tarkempi kuin alkuperäinen sovellus.

3. KONEOPPIMINEN KONENÄKÖSOVELLUKSESSA

Koneoppiminen on menetelmä, jossa kone oppii itsenäisesti saatavilla olevasta materiaalista ilman erillisiä toimintaohjeita. Koneoppimisella pyritään automatisoimaan päätöksentekoa käsiteltävästä datasta löytyvien piirteiden avulla (Baloglu et al. 2021). Se on erityisen hyödyllistä tilanteissa, joissa tarvitsee käsitellä suuria määriä dataa.

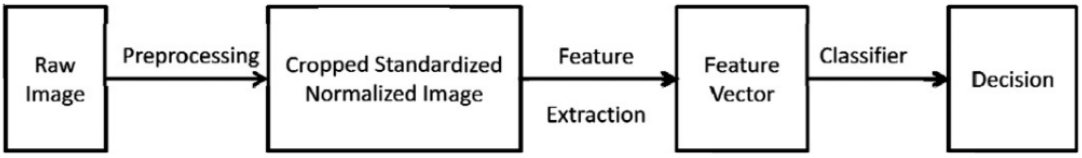
Ohjatussa oppimisessa algoritmi tai sovellus opetetaan valmiiksi luokitellulla opetusaineistolla ennen kuin sitä käytetään oikeassa ympäristössä. Esimerkiksi aineistossa esiintyvät kappaleet on jaettu ehjiin ja rikkinäisiin kappaleisiin. Ohjaamattomassa oppimisessa taas aineistoa ei ole luokiteltu valmiiksi, jolloin algoritmi löytää itsenäisesti luonnollisia luokkia aineiston materiaalien eri piirteiden perusteella (Sebastiani 2002). Jos käytettävä koneoppimisalgoritmi toimii binääriluokittelijana, eli algoritmi luokittelee käsiteltävän aineiston kahteen eri luokkaan, on mahdollista käyttää opetusvaiheessa ristivalidointia, jossa opetusaineisto jaetaan satunnaisesti mielivaltaiseen määrään osia, joista yhtä käytetään testaukseen ja muita opetukseen. Opettelua toistettaessa testaukseen käytettävää osaa vaihdetaan. Ristivalidoinnilla algoritmin opetukseen voidaan käyttää kaikkea saatavilla olevaa aineistoa eikä osa opetusaineistosta jää yli (Domingos 2012).

Koneoppimisen suurin ongelma on ylisovitus (overfitting), jolloin sovellus toimii opetusaineistolla moitteettomasti, mutta oikealla datalla heikosti. Tämä tarkoittaa, että algoritmi ei toimi koko aineiston yleistämiseen, vaan on löytänyt jonkun vain opetusaineistossa esiintyvän relaation (Gonçalves & Silva 2013). Tämä johtaa siihen, että sovelluksen päätökset datasta ovat vääriä.

3.1 Koneoppimisen ja konenäön yhdistäminen

Konenäkösovellusten toiminta voidaan jakaa kolmeen vaiheeseen: kuvankaappaukseen, prosessointiin ja tulosteeseen. Kuvankaappauksessa digitaalinen

kuva siirretään kameralta järjestelmään, jota käytettävä koneoppimisalgoritmi pystyy hyödyntämään. Prosessointivaiheessa kuvaa käsitellään ja siitä etsitään informaatiota esimerkiksi ennalta määriteltyjen parametrien tai koneoppimisen avulla. Tuloste on prosessointivaiheessa saadun tiedon avulla tehdyt havainnot.

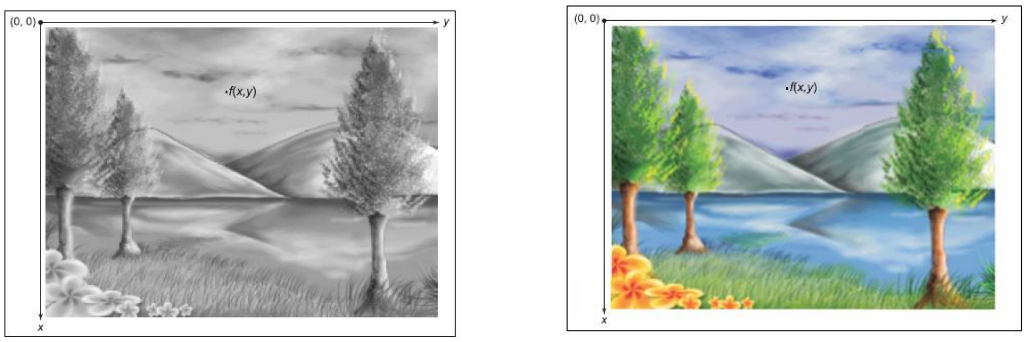


Kuva 1. *Yksi koneoppimisen ja konenäön yhdistystapa. (Ravikumar et al. (2011) s. 2 Kuva 1)*

Kuvassa 2 on esitelty koneoppimisen ja konenäön yhdistämisprosessi. Riippuen tilanteesta koneoppimissovellukset eivät välttämättä käytä raakoja kuvia, koska niissä voi olla häiriöitä tai tarpeettomia elementtejä. Kuvien muokkaukseen voidaan käyttää erilaisia esiprosessointimenetelmiä. Yleisiä käsittelymetodeja ovat kontrastin nostaminen, tarpeettomien piirteiden poisrajaus ja saturaation muuttaminen (Liu et al. 2016). Seuraavaksi siirrytään piirre-erotteluun (feature extraction), jossa kuvasta muodostetaan piirrevektori (feature vector). Esiprosessointi ja piirteiden poiminta kuuluvat molemmat prosessointivaiheeseen. Prosessointivaiheen lopuksi käsiteltävän kuvan data on valmis koneoppimisalgoritmin (classifier) käsiteltäväksi. Kun algoritmi on käsitellyt datan, saadaan tuloste.

3.2 2D-tiedonkäsittely ja koneoppiminen

Kaksiulotteisten kuvien esiprosessointi voidaan jakaa analogiseen ja digitaalisen kuvienkäsittelyyn. (Analog Image Processing vs. Digital Image Processing). Tälle työlle olennaista kuitenkin on vain digitaalinen kuvankäsittely, joten analogista kuvankäsittelyä ei käydä läpi. Digitaalisessa kuvankäsittelyssä pyritään tämän työn kontekstissa muokkaamaan kuvia koneoppimisalgoritmeilla helpommin käsiteltävään muotoon.



Kuva 2. *Harmaasävykuva ja RGB-kuva (Annadurai & Shanmugalakshmi 2006 s. 20 Kuva 2.1.)*

Digitaalinen kuva on sen resoluution (kuva 2 x ja y koordinaatit) määrittelemä joukko pikseleitä, joilla jokaisella on puna- sini- viherkanava arvot välillä 0-255. Eri värikanavien arvojen yhdistelmillä on mahdollisuus muodostaa 16 777 216 väriä. Jokaisesta värikanavasta on mahdollista muodostaa harmaasävykuva, joka vähentää värikanavien määrän kolmesta yhteen. Eri kanavailta johdetut harmaasävykuvat eivät ole identtisiä, vaan saatavan kuvan ulkonäkö riippuu valitusta värikanavasta.

Kuvien harmaasävyisiksi muuttamisen lisäksi on mahdollista muokata kuvien kokoa resoluutiota muuttamalla. Tämä on hyödyllistä erityisesti, jos käytettävä kuvamateriaali sisältää erikokoisia kuvia, mutta myös tilanteissa, joissa kaikki materiaalin sisältävät kuvat ovat samankokoisia, niitä voidaan pienentää käsittelyn nopeuttamiseksi (Zhang et al. 2009).

Kuvia otettaessa tapahtuu poikkeuksetta käytettävästä kamerasta ja sen antureista johtuvia häiriöitä. Kuvahäiriöiden poistossa kuvista pyritään suodattamaan näitä häiriöitä erilaisten algoritmien avulla. Kuvahäiriöiden poistoon voidaan käyttää monenlaisia toimintoja, kuten gaussin sumennusta, suodattimia ja jopa koneoppimista (Deng & Cahill 1993), (Arnal Barbedo 2013).

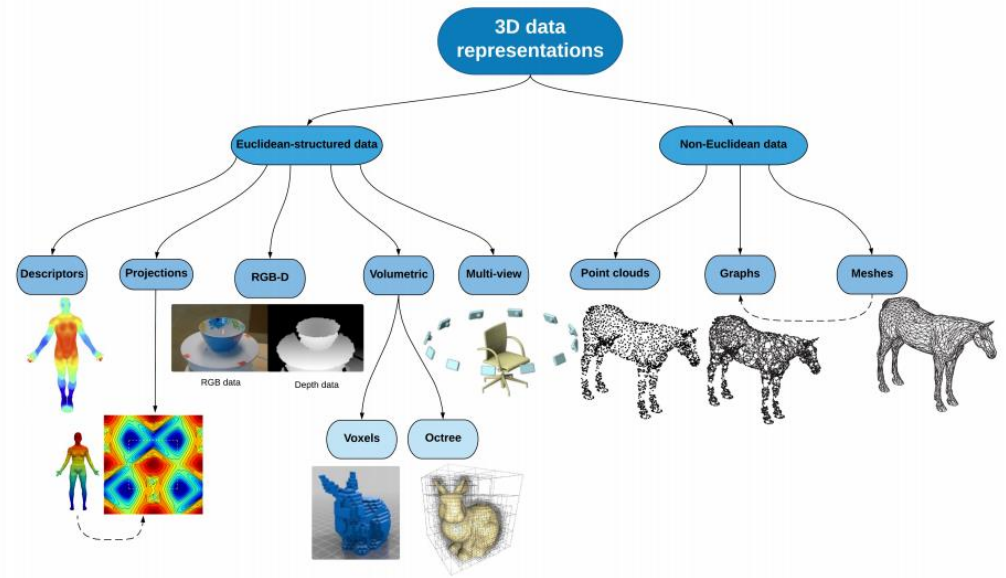
Kuvia käsiteltäessä voi olla myös hyödyllistä erotella kuvan eri alueet taustaksi ja käsiteltäviksi kappaleiksi. Erottelun tavoitteena on saada kuvan olennainen sisältö mahdollisimman helposti tunnistettavaksi. Kun kuvan tausta on eroteltu esiteltävistä kappaleista, sille voidaan antaa kokonaisuutena yksi väriarvo, joka yksinkertaistaa kuvaa huomattavasti (Remboski et al. 2018).

Kun kuvat on käsitelty, voidaan suorittaa piirre-erottelua. Piirre-erottelussa kuvassa esiintyvä informaatio muunnetaan piirrevektoreiksi. Piirrevektorit määrittelevät joukon polynomeja, joiden avulla algoritmi luo kuvasta informaatiota sisältävän vektorin. Tämän jälkeen vektori syötetään algoritmille, joka tekee lopulliset päätökset siitä, mihin luokkaan käsiteltävä kappale kuuluu (Ravimukar et al. 2011).

3.3 3D-tiedonkäsittely ja koneoppiminen

3D-koneoppiminen perustuu pitkälti samoihin menetelmiin kuin vastaavat 2D-dataa hyödyntävät sovellukset. Aikaisemmin 3D-koneoppimisen ongelmana on ollut saatavilla olevan opetusdatan rajoitukset kalliiden laitteistojen takia. Uusimpien 3D-konenäköön liittyvien edistysten takia saatavilla olevan datan määrä on kasvanut huomattavasti. 3D-koneoppimisen pääongelmana onkin nykyisin saatavilla olevan datan käsittely koneoppimisalgoritmeille sopivaksi. Kameralta tulevaa 3D dataa on mahdollista esittää erilaisilla kolmiulotteisilla esitystavoilla.

Nämä eri esitystavat voidaan jakaa euklidisiin ja ei-euklidisiin kategorioihin (Ahmed et al. 2019). Euklidiseen geometriaan perustuvissa esitysmuodoissa tutkittavat kappaleet sidotaan yhteiseen koordinaatistoon, kun taas ei-euklidisessa geometriassa esitystavat perustuvat pistepilveen ja mesh-malleihin.



Kuva 3. *3D-datan eri edustustapoja. (Ahmed et al. 2019 Kuva 1 s. 3)*

Ahmed et al. esittelevät erilaisia 3D-datan esitysmuotoja. Pistepilvi on kokoelma toisistaan riippumattomia pisteitä kolmiulotteisessa avaruudessa, jossa jokaiselle pisteelle on määritelty x-, y-, ja z-koordinaatit. Pisteille on myös mahdollista lisätä muita ominaisuuksia, kuten värejä. Pistepilvien ongelma koneoppimisen kannalta on niiden rakenteen puute, josta johtuen kappaleen ominaisuuksista, kuten esimerkiksi pinnanmuodoista on vaikeaa saada selvää pelkkää pistepilveä käyttämällä. Tätä ongelmaa on pyritty ratkaisemaan käsittelemällä pistepilviä eri metodeilla häiriöiden poistamiseksi ja geometristen piirteiden säilyttämiseksi (Su et al. 2015). Pistepilven avulla voidaan kuitenkin muodostaa erilaisia esitystapoja, kuten mesh-malleja ja vokseliverkkoja. Ne ovat yleensä helpompia käsitellä koneoppimissovelluksilla kuin pistepilvet (Ahmed et al. 2019).

Mesh-malli on yhteisiä sivuja jakavista kolmioista rakennettu pistepilven esitystapa, jolla voidaan approksimoida geometrisia pintoja. Mesh-mallia hyödyntämällä on myös mahdollista tuottaa monikulmakuvauksia. Mesh-malleja on käytetty esimerkiksi ihmiskasvojen kolmiulotteiseen esittämiseen (Nair & Cavallaro 2009).

Vokseliverkko on pistepilvestä johdettu esitysmuoto, jonka yksittäisillä rakennuspalikoilla on vakiotilavuus samalla tavalla kuin 2D-tasossa jokaisella pikselillä on vakiopinta-ala. Vokseleita voidaan pitää pikseleinä kolmiulotteisessa avaruudessa. Myös kappaleiden resoluution säätö on mahdollista samalla tavalla kuin kuvien resoluution säätö 2D-tasossa (Gezawa et al. 2020).

Monikulmakuvaus on kokoelma 3D-mallista otettuja kaksiulotteisia kuvia useammasta eri perspektiivistä. Monikulmakuvausta hyödyntämällä 3D-ongelma voidaan muuntaa 2D-tasossa käsiteltäväksi. Su et al. (2015) käyttivät monikulmakuvausta konvoluutioneuroverkon opetukseen ja saivat tulokseksi 8 % paremman tarkkuuden, kuin 3D-tietoa käyttämällä.

RGB-D on 3D-datan esitysmuoto, jossa kuva esitetään harmaasävyarvona esiteltävän syvyyskartan (D) ja 2D-tasossa olevan värikartan (RGB) avulla. RGB-D on halpa ja tehokas tapa esittää 3D-dataa, koska se voidaan käsitellä 2D-kuvana. Myös algoritmien testausta ajatellen RGB-D on erinomainen esitysmuoto, koska saatavilla olevien kirjastojen määrä on huomattavasti suurempi kuin muiden 3D-datakokoelmien, kuten pistepilvien tai mesh-mallien (Firman 2016).

Järjestelmän 3D-kameralla otetuista kuvista saadaan selville kappaleiden 3D-pistepilvi, jonka perusteella dataa voidaan käsitellä ja tehdä johtopäätöksiä kappaleiden vääntyneisyydestä tai reunojen kääntyneisyydestä. Toinen vaihtoehto kappaleiden 3D-ominaisuuksien analysointiin on pikselikokoon perustuva arviointi. Pikselikokoon perustuvassa arvioinnissa kappaleen ominaisuuksia määritellään arvioimalla sen kattavien pikselien määrää 2D- ja 3D-datasta (Gongal et al. 2017). Gongal et al. (2017) saivat omenien koon arvioinnissa 3D-kameroiden avulla keskimääräiseksi virheprosentiksi pistepilveä käyttämällä 30,9 % ja pikselikokoa käyttämällä 15,2 %. Tutkimus kuitenkin arvioi, että paremmalla kameran resoluutiolla ja pistepilven kartoituksen virheiden korjaamisella voisi parantaa tuloksia.

4. KONENÄKÖÖN SOVELTUVIA ALGORITMEJA JA MENETELMIÄ

Seuraavaksi esitellään kuvasyötteellä toimivia koneoppimisalgoritmeja ja menetelmiä, joita sovelluksessa olisi mahdollista käyttää. Tavoitteena on esitellä yksittäiset algoritmit pääpiirteittäin. Esitellyt algoritmit soveltuvat 2D- ja 3D-tapausten käsittelyyn.

4.1 Koneoppimisalgoritmeista

LabView Machine Learning Toolkit -työkalun LabView-alustalle tarjoamat koneoppimisalgoritmit voidaan jakaa neljään eri kategoriaan: klusterointiin (clustering), luokitteluun (classification), piirrevähennyksen (feature reduction) ja poikkeamatunnistukseen (anomaly detection). Jokainen kategoria sisältää useampia algoritmeja, joita voidaan hyödyntää. Näiden algoritmien lisäksi LabViewillä on mahdollista toteuttaa myös bayesialaisia verkostoja (Bayesian Networks), Naive-Bayes-luokittelua (Naive-Bayes classifier), konvoluutioverkostoja ja erilaisia neuraaliverkostoja (Algorithms in LabVIEW Analytics and Machine Learning Toolkit, 2018).

2D- ja 3D-piirteiden tiedonkäsittelyä varten vaaditaan molemmille tapauksilla omat algoritmit. Kappaleen piirteiden tunnistamiseen 2D-tasossa on yleisesti käytetty konvoluutioneuroverkostoja tai tukivektorikoneita 3D-piirteiden tunnistamiseen neuroverkot ovat olleet suosituimpia, mutta myös muut koneoppimismenetelmät ovat tiedon esiprosessoinnilla mahdollisia (Chowdhury et al. 2016).

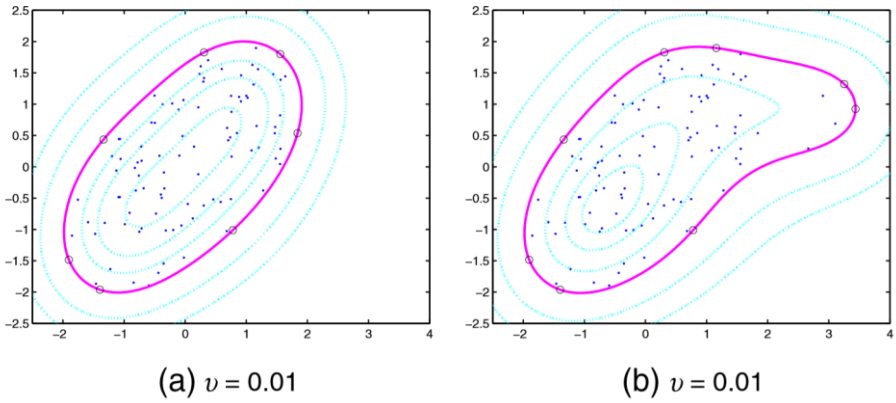
4.2 Tukivektorikoneet

Normaalit tukivektorikoneet (SVM, engl. Support Vector Machine) toimivat perinteisesti binääriluokittelijoina, jotka jakavat käsiteltävät datapisteet, (tässä tapauksessa käsiteltävät kappaleet) mahdollisimman tarkasti kahteen eri luokkaan (Esimerkiksi ehjät ja rikkinäiset kappaleet) Tukivektorikoneen toiminta voidaan esittää datapisteiden ja ne jakavien hypertasojen avulla (Noble 2006). Tukivektorikoneiden toimintaa voidaan myös manipuloida niin, että algoritmi toimii yksiluokkaisena ja ympäröi kaikki saatavilla olevat datapisteet mahdollisimman pienellä hyperympyrällä.

Kuva 4. *Tukivektorikoneprosessi kuvien luokittelussa (mukaillen Qing et al. 2002)*

Kuvassa 4 on esitelty tukivektorikoneiden toimintaketju kuvien luokittelussa. Esiprosessointivaiheessa voidaan käyttää luvuissa 3.2 ja 3.3 esiteltyjä metodeja datan muokkaamiseen haluttuun muotoon. Piirre-erottelussa muodostetaan luvussa 3.2 esitellyt piirrevektorit, joiden perusteella tukivektorikone pystyy luokittelemaan käsiteltävän datan.

Xiao et al. (2015) esittelevät tukivektorikoneen, joka toimii One Class Classification-periaattella (OCC). Algoritmi opetetaan pelkästään kohdejoukkoon kuuluvilla näytteillä, joiden perusteella se luo kohdejoukon ympärille hyperympyrän. Hyperympyrän avulla se pyrkii tunnistamaan kuuluvatko uudet näytteet haluttuun luokkaan vai sen ulkopuolelle. Algoritmin sovelluskohteet ovat kohdistettu erityisesti tapauksiin, joissa virheellisiä näytteitä sisältävää opetusmateriaalia on mahdotonta tai vaikeaa saada. Opetusmateriaalia valittaessa on lisäksi tärkeää välttää kappaleita, jotka ovat kaukana näytteiden kohdejoukon keskiöstä, sillä niillä on suuri vaikutus tukivektorikoneen päätösrajapinnaksi muodostuvan hyperympyrän muotoon.



Kuva 5. *Datapisteiden vaikutus tukivektorikoneen hyperympyrän muotoon. (Xiao et al. 2015 s. 3 Kuva 1)*

Lisäksi he esittelevät metodin, jolla kuvan 5 b-kohdassa esiintyvät häiriötekijät on mahdollista suodattaa pois ja välttää niiden kuulumista tukivektoreiden datapisteiden joukkoon. Yksinkertaisin tapa erotella häiriötekijät opetusmateriaalista on manuaalisesti poistaa häiriötä aiheuttavat datapisteet opetusmateriaalista. Tämä ei kuitenkaan ole helppoa, koska häiriötekijöitä sisältäviä näytteitä ei voida tunnistaa ennen kokeellista testausta. He ehdottavat, että metodin avulla tunnistetaan mahdollisia häiriötä aiheuttavia datapisteitä automaattisesti, siten, että kaikki mahdolliset häiriötekijät esiintyvät tunnistetussa datasetissä. Tämän datasetin avulla voidaan poistaa koulutusvaiheessa tarpeettomat datapisteet ja kun tukivektorikoneiston hyperympyrä on saatu selville, voidaan hyperympyrää vääristävät datapisteet erotella rajapinnan avulla ja suodattaa pois.

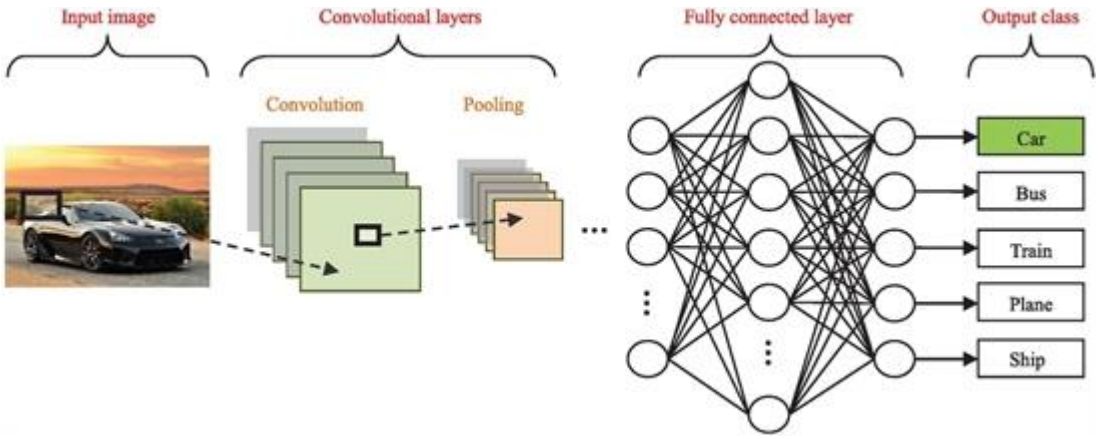
Käytännössä tukivektorikone siis muodostaa pelkkiä ehjiä kappaleita sisältävästä opetusmateriaalista hyperympyrän, joka sisältää rajapintojensa sisäpuolella kaikki opetusmateriaalin datapisteet. Kun testausvaiheessa algoritmille esitetään sekoitus ehjiä ja rikkinäisiä kappaleita, se sijoittaa optimitilanteessa kaikki rikkinäiset kappaleet hyperympyrän ulkopuolelle, jolloin ne hylätään ja kaikki hyväksytyt kappaleet hyperympyrän sisäpuolelle, jolloin ne hyväksytään.

4.3 Konvoluutioneuroverkot

Neuroverkot ovat ihmisen aivosolujen toiminnasta inspiraationsa saanut koneoppimisen menetelmä. (Guo et al. 2017). Ne koostuvat toisiinsa yhdistetyistä

neuronikerroksista. Joiden syötteet on yhdistetty edellisen kerroksen vasteeseen. Neuroverkot kuuluvat ohjatun oppimisen piiriin. Riittävällä määrällä opetusesimerkkejä neuroverkko oppii kuvaamaan funktiota, joka kuvaa syötteen ja vasteen välisen riippuvuuden.

Konvoluutioneuroverkostot ovat neuroverkkojen erikoistapaus, jotka toimivat kerroksissa ja hyödyntävät syväoppimista kuvien piirteiden arvioimiseen. Ne ovat viime vuosina nousseet suosituksi erityisesti kuvien 2D-piirteiden tunnistamisessa (Schmidhuber 2014).



Kuva 6. *Yksi mahdollinen konvoluutioneuroverkon toimintamalli (Rawat ja Wang, 2017, s. 4 Kuva 1)*

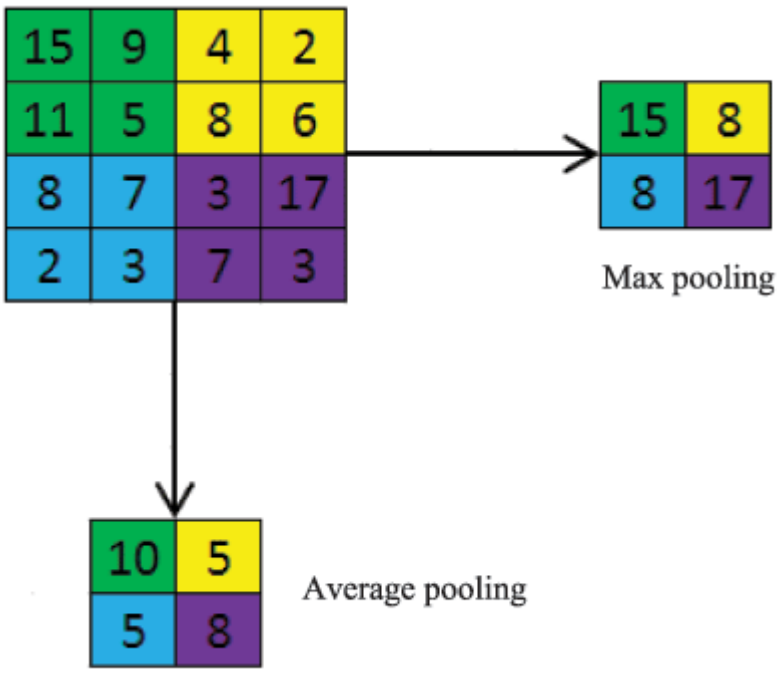
Kuvassa 6 on esitelty pelkistetty konvoluutioneuroverkon toimintamalli. Täysyhteystasojen (fully connected layer), konvoluutiotasojen (convolutional layer) ja vasteluokkien (output class) määrä voi vaihdella algoritmeista ja sovelluskohteista riippuen. Rawat ja Wang (2017) esittelevät konvoluutioneuroverkkojen käyttöä kuvien luokitteluun. Konvoluutioneuroverkon toiminta voidaan jakaa neljään eri tasoon:

1. Konvoluutiotaso
2. Pooling-taso
3. Täysyhteystaso
4. Vastetaso

Konvoluutiotasossa kaikille kuvan värikanaville annetaan painotusmatriisi, jota kutsutaan myös siemeneksi tai suodattimeksi, ja matriisia siirrellään koko kuvan

alueella, kunnes kuvan informaatiomatriisi on muunnettu halutuksi. Jos kuva on harmaasävykuva ei käsiteltävänä ole kuin yksi värikanava, joka vähentää tarvittavaa laskentatehoa. Konvoluutiotason tuloste riippuu käytettävästä painotusmatriisista ja sitä muuttamalla voidaan vaikuttaa pooling-tasolle siirtyvään kuvaan. Kuvien luokittelussa konvooluutiota käytetään piirre-erotteluun. Konvoluutioneuroverkkoja opetettaessa algoritmi valitsee satunnaisen alueen kuvasta, jonka piirteiden perustella painotusmatriisi muodostetaan. Painotusmatriisia siirrellään koko kuvan alueella, jolloin saadaan eroteltua painotusmatriisin mukaiset piirteet koko kuvasta. Konvoluutiotasoja voi olla useampi peräkkäin, joista jokaisella on tietystä kuvan alueesta saatuja piirteitä vastaava painotusmatriisi (Al-Saffar et al. 2017, Chang & Sha 2017).

Painotusmatriisin eri arvoilla ja koolla voidaan vaikuttaa konvoluutiotason toimintaan. Piirre-erottelun lisäksi konvoluutiolla voidaan esimerkiksi sumentaa tai terävöittää kuvia tai muuntaa niiden väritystä (Albawi. et al. 2017).



Kuva 7. *Pooling-tason toiminta (Rawat ja Wang, 2017, s. 6 Kuva 2)*

Myös Pooling-tasossa kaikki mahdolliset värikanavat käsitellään erikseen. Tavoitteena on vähentää tällä hetkellä 2x2 matriisissa olevat värikanavien matriisit 1x1 matriiseiksi. Erilaisia pooling-menetelmiä ovat mm. kuvassa 7 esitellyt max-

pooling, jossa valitaan jokaisen matriisin suurin arvo tai average-pooling, jossa valitaan 2x2 matriisin kaikkien arvojen keskiarvo (Rawat & Wang 2017).

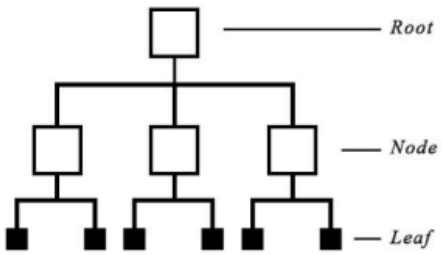
Pooling-tason jälkeen päästään täysyhteystasolle, joiden lukumäärä riippuu valitusta verkosta, jonka jälkeen siirrytään vastetasolle. Täysyhteystasolla kaikilla algoritmin neuroneilla on yhteys kaikkiin aikaisemman tason neuroneihin. ja ne toimivat normaalin neuroverkon tapaisesti (Chang & Sha 2017).

Vastetasossa algoritmi on saanut koko kuvan käsiteltyä ja tekee ennusteen, että mihin opetusmateriaalin ennalta määritellyistä luokista näytekuva todennäköisimmin kuuluu. Luokkien määrä riippuu opetusmateriaalissa erotelluista luokista (Rawat & Wang 2017).

Riskinä neuroverkkojen käytössä on niiden helppo opetusaineistoon ylisovittuminen. Ongelmaa on pyritty ratkaisemaan esimerkiksi ristivalidoinnilla tai useamman neuroverkon käytöllä siten, että ne vahvistavat toistensa tuloksia.

4.4 Päätöspuut

Päätöspuut ovat ohjattua oppimista hyödyntäviä koneoppimisalgoritmeja. Päätöspuu koostuu kerroksittaisesta rakenteesta, joka on jaettu useampiin tasoihin: juureen, sisäsolmuun ja lehtisolmuun. Tasot on liitetty toisiinsa oksilla, jotka kuvaavat luokitteluprosessin suuntaa (Salzberg & Kingsford 2008).



Kuva 8. *Tyypillinen päätöspuun rakenne (Alassar, 2020 s. 2 Kuva 1)*

Kuten kuvasta 8 nähdään, päätöspuun rakenne alkaa juuresta ja jatkuu sisäsolmuihin. Jokaisella päätöspuun tasolla käsiteltävälle syötteelle tehdään piirteidensä mukaan testi, jonka perusteella siirrytään seuraavalle tasolle. Juurisolmulle annetaan syöte, josta se jatkaa sisäsolmuihin. Sisäsolmut edustavat päätöspisteitä, joiden perusteella käsiteltävä syöte luokitellaan. Eri tasoilla olevat

solmut on liitetty toisiinsa oksilla, jotka edustavat aikaisemmalla solmun tasolla tehtyjä päätöksiä. Viimeisenä tasona on lehtisolmut, jotka kuvaavat algoritmin ulostuloja.

Päätöspuut perustuvat useisiin algoritmeihin, jotka yhdessä muodostavat matemaattisen pohjan niiden toiminnalle. Erilaisia päätöspuiden rakennusalgoritmeja ovat Shannonin entropia, C4.5 ja ID3 (Alassar 2020).

Päätöspuiden heikkoutena on epävakaus piirteiden tai harjoitusmateriaalin muuttuessa. Joissain tapauksissa päätöspuut ovat myös hyvin monimutkaisia ja vaativat merkittäviä määriä laskentavoimaa. Lisäksi päätöspuun syventyessä niiden heikkous ylisovittua kasvaa. Ylisovittumista voidaan estää yhdistämällä useita puita metsiksi (random forest) (Norouzi et al. 2015).

4.5 Luokittelijayhdistelmät

Yksittäiset luokittelualgoritmit eivät usein ole riittävän tehokkaita tai tarkkoja halutun lopputuloksen saavuttamiseksi. Mahdollisten koneoppimisalgoritmien määrän lisääntyessä luokittelijayhdistelmistä on tullut houkuttelevampi vaihtoehto parhaan yksittäisen algoritmin selvittämisen sijaan. Luokittelijayhdistelmissä useamman eri algoritmin päätökset yhdistetään ja luokittelupäätös tehdään yhdistelmän perusteella. Yksinkertainen esimerkki luokittelijayhdistelmästä on tilanne, jossa kolme algoritmia äänestää mihin luokkaan käsiteltävän kappaleen piirteet kuuluvat, ja suurimman määrän ääniä saanut vaihtoehto on luokittelijayhdistelmän ulostulo (Tulyakov et al. 2007). Äänestävien algoritmien määrä ei ole vakio, vaan sitä voidaan muuttaa sovelluskohteesta riippuen.

Boosting-menetelmää voidaan käyttää useiden heikkojen luokittelijoiden yhdistämiseen yhdeksi vahvaksi luokittelijaksi luokittelijoiden lineaarikombinaation avulla (Kittler et al. 1998). Menetelmän sisällä eri algoritmien painotusta voidaan säätää vakiokertoimien avulla. Tämä mahdollistaa algoritmien joustavan muokkaamisen ja optimoinnin käsiteltävälle prosessille sopivaksi.

5. SOVELLUS

Tärkeää sovelluksen toimivuuden kannalta on algoritmin tarkkuus tehtävää suoritettaessa ja myös algoritmin toimintanopeus. Algoritmi, joka ei pysty luokittelemaan kuvamateriaalia oikein riittävällä onnistumisprosentilla tai riittävän nopeasti ei ole hyödyllinen tälle sovelluskohteelle. Kuten aikaisemmin jo käsiteltiin, erilaisten algoritmien nopeuksia tietyissä sovelluskohteissa voi olla vaikeaa arvioida tarkasti ennen kokeellista testausta, joten riippumatta käytettävästä algoritmista on tärkeää pyrkiä kokeellistesti etsimään sovelluskohteelle sopivia ratkaisuja.

Käsiteltävien kappaleiden ja taustan helpon eroteltavuuden takia kuvien esiprosessointi voitaisiin todennäköisesti pitää minimaalisena. Kuvien lukuselkeyden nostamiseksi tarvittaisiin korkeintaan kuvien muuttamista harmaasävyisiksi ja taustan erottelua, jolloin jäljelle jäävä aika voi keskittyä vain algoritmin toimintaan. Tämä voi kuitenkin rajoittaa kappaleiden väriin perustuvien piirteiden luokittelua, kuten esimerkiksi kypsyysasteen arviointia. Myös harmaasävykuvista on kuitenkin mahdollista tulkita käsiteltävien kappaleiden kypsyysastetta niiden tummuuden muuttuessa.
Kuten jo aikaisemmin mainittiin, tavoitteena on opettaa sovellus pelkästään eheitä näytteitä sisältävien kuvien avulla. Tässä tapauksessa opetusmateriaalissa esiintyy vain yksi luokka, ehjät kappaleet, joten käytettävän algoritmin täytyy onnistua ehjien ja viallisten kappaleiden erottelussa vain yhden luokan avulla. Sovelluksen opetus vain ehjien kappaleiden avulla tarkoittaa myös, että opetusmateriaali ja testausmateriaali ovat sisällöltään perusteellisesti erilaisia, joten ristivalidoinnista ei tässä tapauksessa ole merkittävää hyötyä. Jos jostain syystä sovellusta ei saada toimimaan pelkkiä ehjiä kappaleita sisältävän opetusmateriaalin avulla ja opetusmateriaalia muutetaan sisältämään kahta eri luokkaa, ehjiä ja viallisia kappaleita, on mahdollista käyttää ristivalidointia.

5.1 Algoritmien vertailua

Widanapathirana et al. (2012) vertailivat erilaisten koneoppimisalgoritmien ominaisuuksia. Heidän havaintojensa mukaan päätöspuut, neuroverkot ja tukivektorikoneet ovat kaikki luokittelunopeudeltaan erinomaisia. Merkittävimmät erot eri algoritmien toiminnassa ovat keskittyvät niiden opetuksen nopeuteen ja opetukseen vaadittavan opetusmateriaalin määrään. Tukivektorikoneet tarvitsevat toimiakseen suuria määriä materiaalia, kun taas neuroverkkojen vaatima materiaalimäärä on pienempi. Päätöspuiden vaatima materiaalin määrä sijoittuu tukivektorikoneiden ja neuroverkkojen välialueelle. Ylisovittumista vastaan parhaiten toimivat tukivektorikoneet, kun taas neuroverkot ja päätöspuut ovat molemmat huomattavasti helpommin ylisovittuvia, josta voi johtua ongelmia sovelluksen toiminnalle.

Koska eri algoritmien toimintatehokkuus sovelluskohteelle on vaikeaa määrittää ennen kokeellista testausta, joudutaan johtopäätösten teossa turvautumaan muiden samankaltaisia järjestelmiä käsitteleviin tutkimuksiin. Sture et al. käyttivät tukivektorikoneita lohien lihanlaadun arviointiin. Pääperusteluna tukivektorikoneiden käyttöön 2D- ja 3D-ominaisuuksien arviointiin he mainitsevat tukivektorikoneiden hyvän ylisovittumisvastaisuuden. Tässä tapauksessa opetusmateriaali sisälsi kahta eri luokkaa, joten sovelluksen opetukseen käytettiin ristivalidointia. Tulokseksi he saivat haavojen tunnistuksessa n. 90 % onnistumistarkkuuden ja epämuodostumien tunnistuksessa 86 % onnistumistarkkuuden. Ravikumar et al. (2011) käyttivät naarmujen ja viiltojen tunnistamiseen konekappaleiden pinnasta C4.5 algoritmilla rakennettua päätöspuualgoritmia. Tässä sovelluskohteessa päätöspuiden luokittelutarkkuus riippui käytettävän opetusmateriaalin määrästä huomattavasti. Parhaassa tilanteessa päätöspuilla päästiin kuitenkin 90,4 % luokittelutarkkuuteen verrattuna Naive Bayes-luokittelijalla saavutettuun 82,7 % tarkkuuteen.

Kuten jo aikaisemmin mainittiin, erityisesti konvoluutioneuroverkot ovat viime vuosina nousseet suosiossa käsiteltäessä kuvia syötteenään käyttäviä järjestelmiä. Nanda et al. (2019) testasivat konvoluutioneuroverkon tarkkuutta käyttämällä CIFAR-10 kuvakokoelmaa. Kokoelma koostuu 60 000 eri kuvasta, jotka sisältävät autoja, lentokoneita, laivoja ja erilaisia eläimiä. Neuroverkon tarkkuus oli

riippuen tunnistettavasta luokasta 57 % ja 90 % välillä, malli kuitenkin toimi huomattavasti paremmin erilaisille ajoneuvoille niiden säännöllisten muotojen takia verrattuna eläinten luokitteluun. Autojen, lentokoneiden ja laivojen luokittelutarkkuus oli alimmillaan 80 % ja korkeimmillaan 90 %. Parannusehdotuksina he esittivät kerrosten lisäämistä malliin, kuvien lisäämistä opetusaineistoon ja saatavilla olevan laskentatehon lisäämistä.

Tarvittaessa on myös mahdollista käyttää luokittelijayhdistämistä useampien eri algoritmien yhdistämiseen. Tellaeche ja Arana (2012) saivat nallien laaduntarkkailussa keskimäärin parempia tuloksia erilaisilla algoritmiyhdistelmillä, kuin pelkillä yksittäisillä algoritmeilla. Artikkelin tapauksessa parhaiten toimi bayesialaisen verkoston, K:n lähimmän naapurin ja C4.5 päätöspuualgoritmin yhdistelmä, joka sai aikaan 94,33 prosentin onnistumisprosentin tarkastettavien kappaleiden arvioinnissa verrattuna pelkkään bayesialaiseen verkostoon, jonka onnistumisprosentti oli 88,33 prosenttia. Myös muut artikkelissa esitellyt algoritmien yhdistelmät toimivat keskimäärin paremmin kuin yksittäiset algoritmit. Boosting-algoritmien ja vastaavien luokittelijayhdistelmien ominaisuudet riippuvat käytettäväksi valittujen luokittelijoiden ominaisuuksista.

2D- ja 3D-piirteitä käsittelevien algoritmien yhdistämiseen voidaan käyttää Boolen logiikan mukaista AND-operaattoria (Zohuri & Moghaddam, 2017), joka hyväksyy käsiteltävän kappaleen vain, jos se läpäisee 2D- ja 3D-piirteiden käsittelyn. Sovellus kävisi erillisten algoritmien tai niiden yhdistelmien avulla läpi 2D- ja 3D-kuvat ja hyväksyisi kuvissa esiintyvät kappaleet vain, jos kappaleet läpäisevät molemmat tarkastusvaiheet. 2D- ja 3D-piirteiden tarkastuksen ei pitäisi pidentää sovelluksen läpäisyaikaa huomattavasti, jos toiminnot voitaisiin ajaa rinnakkain, jolloin aikaa ei menisi hukkaan verrattuna tilanteeseen, jossa saatavat syötteet tarkastettaisiin sarjassa. Piirteiden tarkastus rinnakkain vaatii kuitenkin enemmän laskentatehokapasiteettia verrattuna piirteiden tarkastamiseen sarjassa.

5.2 Johtopäätökset

Tässä kappaleessa esitetyt johtopäätökset on jaettu 2D- ja 3D-tapauksille erikseen. Kaikkia tässä työssä esiteltyjä algoritmeja on mahdollista soveltaa One Class Classification-periaatteella, jolloin opetusmateriaali ei sisältäisi kuin yhden luokan ja mahdollinen testausmateriaali sisältäisi opetusmateriaalissa olevaan luokkaan kuuluvia ja kuulumattomia kappaleita (Chalapathy et al. (2019), Xiao et al. (2016), Itani et al. (2020)).

Esimerkkejä kaikkien käsiteltyjen koneoppimisalgoritmien toiminnasta LabVIEW-ympäristössä löytyy kirjallisuudesta. Abdullah et al. (2011) käyttivät LabVIEW Laboratory Virtual Instrument Engineering Workbench-sovellusta ja tukivektorikoneita aivojen MRI-kuvien luokitteluun. Cruz de la Cruz & Vera Ramirez (2020) käyttivät konvoluutioneuroverkkoja avokadojen luokitteluun. Demetgul & Demetgul (2013) käyttivät päätöspuita tuotantosysteemien vikojen tunnistamiseen, vaikkei kyseessä ole kuvien luokittelua käyttävä sovelluskohde, he osoittavat, että LabView-ympäristössä voidaan toteuttaa myös päätöspuurakenteita. Yhteenvetona tukivektorikoneet ja konvoluutioneuroverkot soveltuvat varmasti LabVIEW-alustalla kuvien luokitteluun. Lisäksi voidaan olettaa, että myös päätöspuiden hyödyntäminen kuvien luokitteluun on mahdollista.

5.2.1 2D-algoritmi

Kappaleiden yksinkertaisten geometristen muotojen takia käytettävän 2D-algoritmin ei tarvitse soveltua monimutkaisten muotojen tunnistamiseen. Tämän takia konvoluutioneuroverkot, jotka ovat nykyisin ansioituneet erityisesti monimutkaisempien piirteiden tunnistamisessa kuvista, eivät välttämättä ole tarpeellisia (Hansen et al. 2020). Päätöspuut voivat myös olla tarpeettoman monimutkaisia ja ylisovittua liian helposti. Siksi lähtökohtana käytännön testaamiselle suositeltaisiin tukivektorikoneista aloittamista. Testattujen algoritmien perusteella päätöspuita, konvoluutioneuroverkkoja tai muita algoritmeja voitaisiin lisätä luokittelijayhdistelmien avulla sen ollessa tarpeellista.

5.2.2 3D-algoritmi

Lähtökohtaista 3D-algoritmia valittaessa työssä esitellyistä vaihtoehdoista on haastavampi erotella selvää lähtökohtaa, josta aloittaa testaaminen. Esimerkkejä kaikista esitellyistä algoritmeista 3D-piirteiden tunnistamisesta löytyy kirjallisuudesta (Griffiths & Boehm, 2019), vaikka myös tässä kategoriassa konvoluutioneuroverkot ovat saavuttaneet viime aikoina suosiota.

3D-dataa käsiteltäessä on hyvä muistaa, että käytettävien menetelmien toimivuus riippuu hyvin paljon siitä, miten dataa on käsitelty ja missä muodossa se esitetään koneoppimisalgoritmille. Algoritmin toiminnan helpottamiseksi 3D-tapauksen muuntaminen 2D-tasoon monikulmakuvauksen tai RGB-D:n avulla voisi mahdollistaa enemmän erilaisten algoritmien käyttöä.

3D-datan monimutkaisemman geometrian ja suuren esiprosessointivalikoiman takia konvoluutioneuroverkot ovat sopivin aloituskohta 3D-algoritmin testaamiselle. Jos kuitenkin päädytään käyttämään esiprosessointimenetelmiä, jotka muuntavat 3D-datan kaksiulotteiseksi, voitaisiin 2D- ja 3D-datan käsittelyyn käyttää samaa algoritmia.

6. YHTEENVETO

Tämän kirjallisuusselvityksen tavoitteena oli tutusta koneoppimisratkaisuihin, joita voitaisiin käyttää elintarviketeollisuuden konenäkösovelluksessa ja saatavilla olevan kirjallisuuden perusteella tehdä alustavia ehdotuksia käytännön toteutuksen vaihtoehdoista.

Nykyisillä 2D-kuvien ja 3D-datan käsittelyyn soveltuvilla koneoppimisalgoritmeilla on mahdollisuus muodostaa monenlaisia tiettyihin kohteisiin räätälöityjä sovelluksia. Algoritmien ja niitä hyödyntävien sovellusten toiminta on tapauskohtaista, ja yhdessä tilanteessa toimiva algoritmi ei välttämättä toimi toisessa tilanteessa. Tästä johtuen konenäkösovelluksissa käytettävät koneoppimismenetelmät ovat hyvin monimuotoisia, eivätkä tässä työssä esitellyt algoritmit ole ainoita mahdollisia kuvia sisääntulonaan käyttäviä algoritmeja. Tällä hetkellä suosituimpia kuvia ja 3D-dataa sisääntulonaan ottavia algoritmeja ovat neuroverkot ja niiden eri variaatiot, joita voidaan räätälöidä tarpeen mukaan tarkoitukseen sopivaksi.

Tutkimuksen tavoitteena oli selvittää koneoppimista hyödyntäviä algoritmeja ja menetelmiä liukuhihnalla kulkevien kappaleiden laaduntarkkailuun konenäkösovelluksen avulla. Tulokseksi saatiin esiteltyä kolme eri algoritmia ja menetelmiä niiden yhdistelemiseksi. Toisena tavoitteena oli alustava ehdotus käytännön toteutukselle, jossa päädyttiin tukivektorikoneisiin 2D-tiedonkäsittelyssä ja konvoluutioneuroverkkoihin 3D-tiedonkäsittelyssä. Työn tavoitteisiin päästin onnistuneesti. Nämä valinnat ovat kuitenkin vain aloituspisteitä sovelluksen käytännön testaamiselle, jonka pohjalta todelliset valinnat tehtäisiin. Kaikille esitellyille koneoppimisalgoritmeille löytyi esimerkkejä LabVIEW-alustalla toimivista sovelluksista ja kahdelle kolmesta kyseessä oli kuvien luokitteluun rakennettu sovellus.

Seuraava jatkotutkimuskohde olisi sovelluksen käytännön toteutus, jossa tarkasteltaisiin tarkemmin algoritmien toimintaa ja tehtäisiin lopullisia ratkaisuja käsitellylle järjestelmälle parhaasta koneoppimismenetelmästä.

LÄHTEET

Abdullah, N. Ngah, U. & Aziz, S. A. (2011) Image classification of brain MRI using support vector machine. IEEE International Conference on Imaging Systems and Techniques. IEEE. s. 242–247.

Ahmed, E., Saint, A., Shabayek A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D. & Ottersten, B. (2018) A Survey on Deep Learning Advances on Different 3D Data Representations. arXiv:1808.01462.

Alassar, Z. (2020) Decision Tree as An Image Classification Technique. Saatavilla: https://www.researchgate.net/publication/342707557_DECISION_TREE_AS_AN_IMAGE_CLASSIFICATION_TECHNIQUE

Albawi, S., Mohammed. T., Tareq, A. & Al-Zawi, S. (2017) Understanding of a convolutional neural network International Conference on Engineering and Technology (ICET). IEEE. s. 1–6.

Al-Saffar, A. A. M. Hai, T., Talab, M. A. (2017) Review of deep convolution neural network in image classification. International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). IEEE. s. 26–31.

Al-Audah, Y. K., Al-Juraifani, A. K., & Deriche, M. A. (2012). A Real-Time License Plate Recognition System for Saudi Arabia Using LabVIEW. 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA). s. 160–164.

Analog Image Processing vs. Digital Image Processing. Saatavilla: https://www.javatpoint.com/analog-image-processing-vs-digital-image-processing Vierailtu: 20.4.2021.

Annadurai, S. & Shanmugalakshmi, R. (2006) Fundamentals of digital image processing. 1st edition. Pearson.

Arnal Barbedo, J. G. (2013) Digital image processing techniques for detecting, quantifying and classifying plant diseases. SpringerPlus. Vol. 2 (1), s. 1–12.

Baloglu, O., Samir, Q. L., & Nazha, A. (2021) What Is Machine Learning? Archives of Disease in Childhood. Education and Practice Edition: 2021;0: s. 1-3.

Chalapathy, R., Menon, A. K., & Chawla, S. (2018). Anomaly Detection Using One-Class Neural Networks. arXiv:1802.06360

Chang, J. & Sha, J. (2017) An efficient implementation of 2D convolution in CNN. IEICE Electronics Express. Vol. 14 (1), s. 1–8.

Chowdhury, A., Kautz, E, Yener, B & Lewis, D. (2016) Image driven machine learning methods for microstructure recognition. Computational materials science. Vol. 123 s. 176–187.

Gonçalves, I., and Sara, S. (2013). Balancing Learning and Overfitting in Genetic Programming with Interleaved Sampling of Training Data. Genetic Programming. Vol. 7831. Berlin, Heidelberg: Springer Berlin Heidelberg. s. 73–84.

Cruz de la Cruz, J. E. & Vera Ramirez, O. J. (2020) Convolutional neural networks for the Hass avocado classification using LabVIEW in an agro-industrial plant. IEEE XXVII International Conference on Electronics, Electrical Engineering and Computing (INTERCON). s. 1–4.

Deng, G. & Cahill, L. (1993) An adaptive Gaussian filter for noise reduction and edge detection. IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference. Vol.3. s. 1615–1619

Demetgul, M. & Demetgul, M. (2013) Fault diagnosis on production systems with support vector machine and decision trees algorithms. International journal of advanced manufacturing technology. Vol. 67 (9), s. 2183–2194.

Domingos, P. A Few Useful Things to Know About Machine Learning. (2012). Communications of the ACM Vol. 55: s. 1078–87.

Firman, M. (2016) RGBD Datasets: Past, Present and Future IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). s. 661–673.

Gongal, A., Karkee, M., & Amatya, S. (2018) Apple Fruit Size Estimation Using a 3D Machine Vision System. Information Processing in Agriculture Vol. 5.4: s. 498–503.

Gonzalez, V., Gonzalez, C., Fuentes, S., Torrico, D. D., Howell, K. & Dunshea F. R. (2018). Assessment of Beer Quality Based on a Robotic Pourer, Computer Vision, and Machine Learning Algorithms Using Commercial Beers. Journal of food science Vol. 83.5: s. 1381–1388.

Guo, T., Dong, J., Li, H. & Gao, Y. (2017). Simple Convolutional Neural Network on Image Classification. IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017. s. 721–724.

Griffiths, D. & Boehm, J. A Review on Deep Learning Techniques for 3D Sensed Data Classification. (2019). Remote sensing Vol. 11 (12): s. 1499.

Gezawa, A. S., Abubakar, S., Zhang, Y., Wang, Q. & Yunqi, L. (2020) A Review on Deep Learning Approaches for 3D Data Representations in Retrieval and Classifications. IEEE access. Vol. 8 s. 57566–57593.

Hansen, O. L. P., Svenning, J., Olsen, K., Dupont, S., Garner B. H., Iosifidis, A., Price, B. W. & Hoye, T. T. (2020). Species-level Image Classification with Convolutional Neural Network Enables Insect Identification from Habitus Images. Ecology and Evolution Vol. 10.2: s. 737–747.

Itani, S., Lecron, F., & Fortemps, P. (2020). A One-Class Classification Decision Tree Based on Kernel Density Estimation. Applied soft computing Vol. 91: Article 106250.

Kittler, J., Hatef, M., Duin, R. P. W. & Matas, J. (1998) On combining classifiers. IEEE transactions on pattern analysis and machine intelligence. Vol. 20 (3), s. 226–239.

LabVIEW TM. (2013). Getting Started with LabVIEW Getting Started with LabVIEW Saatavissa: https://www.ni.com/pdf/manuals/373427j.pdf

LabVIEW TM. (2018) Algorithms in LabVIEW Analytics and Machine Learning Toolkit. Saatavissa: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000kJXLSA2

Li, X., Qiao, T., Pang, Y., Zhang, H. & Yan, G. (2018). A New Machine Vision Real-Time Detection System for Liquid Impurities Based on Dynamic Morphological Characteristic Analysis and Machine Learning. Measurement: Journal of the International Measurement Confederation Vol. 124: s. 130–137.

Nanda, A., Kumar, P. & Rawat, S. (2019). Implementing Convolutional Neural Networks for Simple Image Classification. International journal of engineering and advanced technology Vol. 9.2: s. 3616–3619.

Nair, P. & Cavallaro, A. (2009) 3-D Face Detection, Landmark Localization, and Registration Using a Point Distribution Model," in IEEE Transactions on Multimedia, Vol. 11, no. 4, s. 611-623.

Noble, W. S. (2006). What Is a Support Vector Machine? Nature biotechnology Vol. 24.12: s. 1565–1567.

Norouzi, M., Collins, M. D., Johnson, M., Fleet, D. J. & Kohli, P. (2015) Efficient non-greedy optimization of decision trees. arXiv:1511.04056

Qing, S., Wenjie, H., & Wenfang, X. (2002). Robust Support Vector Machine with Bullet Hole Image Classification. IEEE transactions on systems, Man and Cybernetics. Part C, Applications and Reviews Vol. 32.4: s. 440–448.

Ravikumar, S., Ramachandran, K. I., & Sugumaran, V. (2011). Machine Learning Approach for Automated Visual Inspection of Machine Components. Expert systems with applications Vol. 38.4: s. 3260–3266.

Rawat, W. & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural computation Vol. 29.9: s. 2352–2449.

Remboski, T., de Souza, W., de Aguiar, M. & Ferreira Jr, P. (2018) Identification of fruit fly in intelligent traps using techniques of digital image processing and machine learning, in Proceedings of the 33rd Annual ACM Symposium on applied computing. s. 260–267.

Salzberg, S. L. & Kingsford, C. (2008) What are decision trees? Nature biotechnology. Vol. 26 (9), s. 1011–1013.

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview.” Neural networks Vol. 61: s. 85–117.

Sebastiani, F. (2002). “Machine Learning in Automated Text Categorization.” ACM computing surveys Vol. 34.1: s. 1–47.

Sture, Ø., Øye, E., Elling, R., Skavhaug, A. & Mathiassen, J. R. (2016). A 3D Machine Vision System for Quality Grading of Atlantic Salmon. Computers and electronics in agriculture Vol. 123: s. 142–148.

Su, H., Maji, S., Kalogerakis, E. & Learned-Miller, E. (2015). Multi-View Convolutional Neural Networks for 3D Shape Recognition. IEEE International Conference on Computer Vision (ICCV). Vol. 2015-12. s. 945–953.

Tellaeche, A., & Arana, R. (2011). Three-Dimensional Machine Vision and Machine-Learning Algorithms Applied to Quality Control of Percussion Caps. IET Computer Vision Vol. 5.2: s. 117–124.

Tulyakov, S., Jaeger, S., Govindaraju, V. & Doermann, D. (2007). Review of Classifier Combination Methods. Machine Learning in Document Analysis and Recognition. Vol. 90. s. 361–386.

Widanapathirana, C. (2012). Automated Inference System for End-To-End Diagnosis of Network Performance Issues in Client-Terminal Devices.” International Journal of Computer Networks & Communications Vol. 4.3: s. 37–56.

Xiao, Y., Wang, H., Xu, W. & Zhou, J. (2016). Robust One-Class SVM for Fault Detection. Chemometrics and Intelligent Laboratory Systems Vol. 151: s. 15–25.

Xu, J. & Sun, D. (2017). Identification of Freezer Burn on Frozen Salmon Surface Using Hyperspectral Imaging and Computer Vision Combined with Machine Learning Algorithm. International Journal of Refrigeration Vol. 74: s. 151–164.

Zhang, G., Cheng, M., Hu, S., Martin, R. R. (2009) A Shape-Preserving Approach to Image Resizing. Computer graphics forum. Vol. 28 (7), s. 1897–1906.

Zohuri, B. & Moghaddam, M. (2017) What Is Boolean Logic and How It Works. Business Resilience System (BRS): Driven Through Boolean, Fuzzy Logics and Cloud Computation. Cham: Springer International Publishing: s. 183–198.