HAMK
Hämeen ammatti-
korkeakoulu

Objektintunnistus tuotantolinjojen laaduntarkkailussa

Koneoppimismallien vertailu

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Kevät 2025

Rony Mölkänen

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus	Tietojenkäsittelyn koulutus	
Tekijä	Rony Mölkänen	Vuosi 2025
Työn nimi	Objektintunnistus tuotantolinjojen laaduntarkkailussa Koneoppimismallien vertailu	
Ohjaaja	Tommi Lahti	

Opinnäytetyön tarkoituksena oli vertailla kahta koneoppimismallia tuotantolinjojen laaduntarkkailun objektintunnistuksen kontekstissa. Tavoitteena oli löytää suorituskykyisempi malli näistä kahdesta. Tutkittavat koneoppimismallit olivat YOLOv11, jonka arkkitehtuuri perustuu konvoluutioneuroverkkoihin, ja RT-DETR, jonka arkkitehtuuri perustuu Transformer-arkkitehtuuriin. Opinnäytetyön toimeksiantaja oli Konekatse Oy.

Opinnäytetyön teoreettisessa osuudessa avataan työn kannalta keskeisiä käsitteitä kuten tekoäly, koneoppiminen sekä molempien käsitteiden alahaarautumia. Työn tietopohja perustuu ajankohtaisiin tieteellisiin julkaisuihin sekä koneoppimismallien virallisiin dokumentaatioihin. Opinnäytetyön tyyppi on toiminnallinen, ja siinä sovelletaan kokeellista tutkimusotetta. Koneoppimismallit koulutettiin tuotantoympäristöä vastaavalla datalla, ja niitä arvioitiin ja vertailtiin eri kvantitatiivisten mittareiden perusteella.

Työn tulosten perusteella havaittiin, että molemmat mallit suoriutuvat kiitettävällä tasolla objektintunnistuksesta. Kuitenkin RT-DETR suoriutui paremmin pienemmillä koulutusdatamäärillä, ja YOLO hyötyi suuremmista koulutusdatamääristä. Lopullisena johtopäätöksenä työn perusteella suositellaan, että toimeksiantaja valitsee jatkokehitykseen Transformer-arkkitehtuurin omaavaan RT-DETR-koneoppimismallin.

Avainsanat	Konenäkö, Koneoppiminen, Tekoäly, Objektintunnistus
Sivut	34 sivua ja liitteitä 4 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP	Degree Programme in Business Information Technology
Author	Rony Mölkänen
Subject	Object Detection for Quality Control in Production Lines Comparison of Machine Learning Models
Supervisor	Tommi Lahti

Year 2025

The purpose of this thesis was to compare two machine learning models in the context of object detection for quality control on production lines. The objective was to identify the more performant model of the two. The machine learning models under investigation were YOLOv11, which is based on a convolutional neural network architecture, and RT-DETR, which is based on the Transformer architecture. The thesis was commissioned by Konekatse Oy.

The theoretical part of the thesis introduces key concepts relevant to the work, such as artificial intelligence, machine learning, and their subfields. The knowledge base is built upon up-to-date scientific publications and the official documentation of the machine learning models. The thesis is functional in nature and employs an experimental research approach. The models were trained using data that reflects the target production environment, and their performance was evaluated and compared using various quantitative metrics.

Based on the results, both models performed well in the object detection task. However, RT-DETR showed better performance with smaller training datasets, whereas YOLOv11 benefited from larger amounts of training data. As a final conclusion, it is recommended that the client selects RT-DETR, the Transformer-based machine learning model, for further development.

Keywords	Computer Vision, Machine Learning, Artificial Intelligence, Object Detection
Pages	34 pages and appendices 4 pages

Sisällys

Sanasto

Tekoäly	Ihmisen älyä jäljittelevä tietokoneohjelma
Koneoppiminen	Tekoälyn osa-alue, jossa kone oppii datasta itsenäisesti
Data	Tiedot, aineisto
Konenäkö	Tekoälyn osa-alue, jossa kone tulkitsee kuvia tai videoita
Laadunvalvonta	Prosessi, jolla varmistetaan tuotteiden tai palveluiden laatu
Algoritmi	Joukko ohjeita ja sääntöjä
Iteraatio	Toistuva prosessi
Kielimalli	Koneoppimismalli, joka ymmärtää, tuottaa ja muokkaa luonnollista kieltä
Tekoälytalvi	Ajanjakso, jolloin tekoälytutkimukseen kohdistunut kiinnostus laski merkittävästi
Avoin lähdekoodi	Ohjelmistokehitysmenetelmä, jossa lähdekoodi on vapaasti kaikkien saatavilla
Hyperparametri	Koneoppimismallien oppimisprosessiin vaikuttava arvo

Kuvat

Taulukot

Ohjelmakoodit

Liitteet

1 Johdanto

Teollisilla tuotantolinjoilla liikkuu koko maailman mittapuulla suuria määriä erilaisia tuotteita vuoden jokaisena päivänä. Yhtenä haasteena tässä on laaduntarkkailu. Tuotteiden laaduntarkkailu tuotannossa syntyneiden virheiden vuoksi on ensiarvoisen tärkeää esimerkiksi virheellisten tuotteiden kustannuksien minimoimiseksi. Perinteiset laaduntarkkailumenetelmät, kuten ihmisen suorittama manuaalinen laaduntarkkailu on hidasta sekä täysin riippuvainen ihmisen havainnointikyvystä. Nykypäivän teknologinen kehitys mahdollistaa automaation luomisen laaduntarkkailuun tuotantolinjoille.

Tässä opinnäytetyössä perehdytään koneoppimisen maailmaan, ja tutkitaan koneoppimisarkkitehtuurien tarjoamia mahdollisuuksia laadunvalvontaprosessin automatisointiin tuotantolinjalla. Työssä pureudutaan laaduntarkkailun ensimmäiseen ja samalla tärkeimpään vaiheeseen, eli objektintunnistamiseen.

Opinnäytetyössä pyritään löytämään vastaukset alle oleviin tutkimuskysymyksiin:

- Kuinka tarkasti tutkittavat koneoppimismallit oppivat havaitsemaan objektit?
- Kuinka pienellä koulutusdatalla mallit oppivat tunnistamaan objektin?
- Mikä tutkittavista koneoppimismalleista on asetettujen kriteerien mukaan optimaalisin tuotantolinjojen objektien tunnistamiseen?

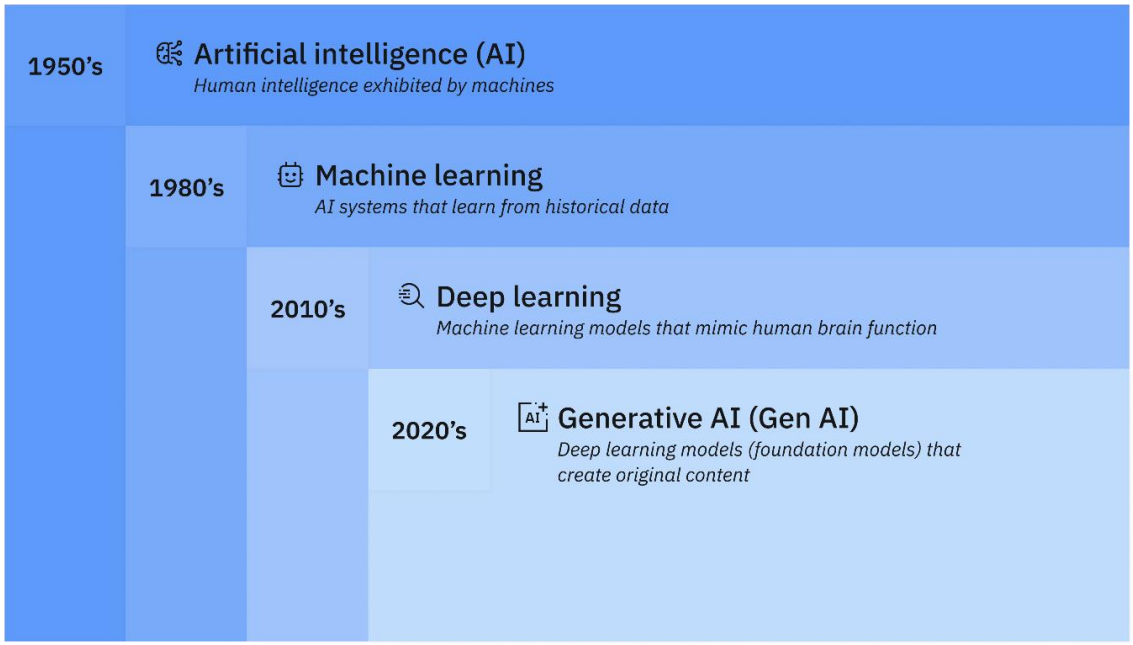
Opinnäytetyön toimeksiantajana toimii Konekatse Oy. Konekatse on hämeenlinnalainen, vuonna 2024 perustettu konenäköön ja sen sovellutuksiin erikoistunut yritys. Opinnäytetyön tavoitteena on tutkia ja löytää tarkin koneoppimismalli objektintunnistukseen tuotantolinjalla, joka oppii tunnistamaan uuden objektiluokan mahdollisimman pienellä määrällä dataa. Opinnäytetyön tuloksen avulla toimeksiantajayritys voi valita lopulliseen kaupalliseen tuotteeseen käytettävän koneoppimismallin, ja alkaa kehittämään lopullista kaupallista tuotetta.

2 Tekoäly

Tekoälyä (eng. Artificial Intelligence, AI) on vaikea määritellä. Edes tutkijat eivät ole pystyneet määrittelemään, mitä älykkyys varsinaisesti on ja miten se ilmenee. (Järvinen, 2023, s. 49) Järvinen (2023, s. 48) määrittelee tekoälyn olevan kattotermi erilaisille ohjelmointitekniikoille, joilla pyritään matkimaan ihmistä ja ihmisen eri kykyjä ja taitoja. Kysyttäessä tekoälyltä itsessään ”Miten määrittelet tekoälyn?”, Microsoft Copilot vastaa tekoälyn tarkoittavan tietokonejärjestelmää, joka jäljittelee ihmisen eri kognitiivisia toimintoja, kuten oppimista, päättelyä ja ongelmanratkaisua (Microsoft, 2024).

Tekoälyn historia ulottuu 1940-luvulle, jolloin myös ensimmäiset digitaaliset tietokoneet keksittiin. Kuitenkin varsinaisten tekoälysovellusten kehittäminen alkoi vasta seuraavalla vuosikymmenellä Yhdysvalloissa. (Suomen Koodikoulu, 2019, s. 6) Kuva 1 havainnollistaa tekoälyn historian 1950-luvulta lähtien ja sen kehityksen eri vaiheet nykyvuosikymmenelle asti. Kaikki alkoi 1950-luvulla, kun John MacCarthy sanoi ääneen ensimmäisen kerran sanan tekoäly. Koneoppiminen tuli mukaan 1970–1980 lukujen aikana. Myös neuroverkkoihin pohjautuvat tiedonkäsittelyn menetelmät alkoivat kehittyä, ja niitä otettiin laajemmin käyttöön. 2010-luvulla syväoppiminen mahdollisti edistyneempien koneoppimismallien luomisen. (Suomen Koodikoulu, 2019, ss. 9–10) Nykyvuosikymmenelle tultaessa generatiivinen tekoäly mahdollistaa jo useita asioita, kuten puheen tai esseen generoimisen, kuvien tunnistamisen, musiikin luomisen sekä ihmisellekin monimutkaisten ongelmien ratkaisemisen.

Kuva 1. Tekoälyn kehitysaskeleet (IBM, 2024)



Ei tekoälyn kehitys nykyhetkeen asti ole aina ollut aivan selvää. Tekoälyn tutkiminen ja kehittäminen vaatii paljon rahoitusta. Näin oli myös tekoälyn alkuvuosina. 1950-luvulla Yhdysvalloissa tutkittiin mahdollisuutta venäjänkielisten tekstien kääntämiseen tekoälyn avulla. Loppujen lopuksi tekoäly ei onnistunut tehtävässä odotusten lailla. Vuonna 1966 komitea julkaisikin raportin liittyen tekoälyyn, jossa todettiin, että tekstin kääntäminen tekoälyn avulla on hitaampaa, kalliimpaa sekä laadultaan huomattavasti huonompaa kuin ihmisen tekemänä. Yhdysvallat olivat rahoittaneet hanketta miljoonilla dollareilla, ja kaikki menivät hukkaan. Tästä seurasi niin sanottu tekoälytalvi. Tekoälytalvia seurasi useita toisensa jälkeen, kun tekoälystä ei saatu luotua järkevää kaupallista tuotetta. (Järvinen, 2023, ss. 56–57)

Tekoäly käsitteenä on erittäin laaja. Käsitteen tekoäly jaottelu helpottaa ymmärtämistä, koska tekoälyn osa-alueet eroavat merkittävästi toisistaan toiminnallisuuden ja käyttötarkoituksien osalta. Näistä osa-alueista kerron tarkemmin seuraavassa luvussa.

2.1 Tekoälyn osa-alueet

Heikolla tai kapealla tekoälyllä (eng. narrow AI) tarkoitetaan nykyaikaisia AI-sovelluksia tai järjestelmiä. Nämä sovellukset tai järjestelmät ovat suunniteltu tekemään muutamia erilaisia yksinkertaisia tehtäviä. Näitä heikon tekoälyn tehtäviä ovat esimerkiksi puheentunnistaminen tai tekstin kääntäminen toiselle kielelle. (Tekoälyaika, 2023) Järvinen (2023, s. 51) toteaa kuitenkin tekstissään, että vaikka kone pystyisikin tekemään vain yhden prosessin kerrallaan, pystyy se tekemään tämän erittäin tehokkaasti, luultavasti ihmistä tehokkaammin.

Vahva tai yleinen tekoäly (eng. Artificial General Intelligence, AGI) viittaa tekoälyyn, joka kykenee ajattelemaan ja toimimaan ihmisen tavoin. AGI kykenisi oppimaan uusia asioita ihmisen tasoisesti. AGI ymmärtäisi monen eri asian yhteyksiä, ja pystyisi tekemään täysin itsenäisiä johtopäätöksiä. (Järvinen, 2023, s. 51)

Kuten aikaisemmin todettiin, tällä hetkellä eletään heikon tekoälyn aikaa. Vahvaan tekoälyyn liittyvät tutkimukset etenevät hurjaa vauhtia, ja joitain esimerkkejä jo vahvan tekoälyn hyödyntämisestä löytyy (Tekoälyaika, 2023). Tutkijat ovat kuitenkin sitä mieltä, että AGI:n valjastamista käyttöön pidetään tällä hetkellä mahdottomana, koska se vaatisi merkittäviä lisäyksiä nykyisin saatavilla olevaan tietokoneiden laskentatehoon (IBM, 2024).

2.2 Tekoälyn etiikka

Tekoäly tuo mukanaan monenlaisia hyötyjä, mutta sen mukana esiin nousee myös uusia ennenäkemättömiä eettisiä ja yhteiskunnallisia haasteita. On tärkeää miettiä kuinka taataan tekoälyn käyttö oikeudenmukaisesti ja vastuullisesti. (Tekoälyaika, 2023) Yksi näistä haasteista on työpaikkojen mahdollinen katoaminen. Tekoälyn uskotaan korvaavan ihmisen manuaalisten työtehtävien suorittamisessa, jotka ovat toistuvia ja säännönmukaisia. Toisaalta, jos tekoälyllä voidaan automatisoida nämä tavanomaiset ja toistuvat työtehtävät, ihmiset voidaan ohjata työtehtävien pariin, jotka vaativat luovuutta ja inhimillisyyttä. (Viitaila, 2018)

Tekoälyn katsotaan vaikuttavan työelämään mullistavalla tavalla. Kuten aiemmin mainittiin, tekoäly ja sen integroiminen työmaailmaan tuo mukanaan sen, että työpaikkoja tulee

katoamaan. Viitaila (2018) kuitenkin toteaa kirjoituksessaan, että siinä missä tekoälyllä aiemmin ihmisen suorittamia työtehtäviä voidaan automatisoida, ja työtehtäviä tulee katoamaan, niin tekoäly samalla luo kuitenkin myös uusia työpaikkoja, joista monet ovat vasta kehittymässä ja osa on vielä täysin tuntemattomia.

Toinen ja samalla hyvin erilainen näkökulma tekoälyn käytön etiikassa on sen valjastaminen useisiin kiistanalaisiin tarkoituksiin, kuten esimerkiksi valvontaan. Tänä päivänä kasvojentunnistusteknologiaa käytetään yhä laajemmin useimmissa suurimmissa kaupungeissa ympäri maailmaa julkisten tilojen valvontaan, ja tätä kautta esimerkiksi rikollisten tunnistamiseen. Vaikkakin teknologian käytön tavoitteet ovat periaatteessa hyvät, monet kyseenalaistavat sen luotettavuuden sekä pohtivat, onko ihmisten seuraaminen ja tämänkaltainen vakoilu eettisesti perusteltua ja hyväksyttävää. (Klusaitė, 2023)

Kolmas huomionarvoinen eettinen kysymys liittyy tekoälyyn ja sen hyödyntämiseen vaikuttamisessa ihmisten asenteisiin ja mielipiteisiin. Esimerkkinä tästä tapaus Cambridge Analytica. Cambridge Analytica oli brittiläinen poliittisia konsultointipalveluita tarjonnut yritys. Heillä oli näppinsä pelissä vuoden 2016 Yhdysvaltojen presidentinvaaleissa. Yritys työskenteli tuona aikana Donald Trumpille ja hänen vaalikampanjalleen. Yritys oli kerännyt yli 50 miljoonan äänestäjän Facebook profiileihin liittyvän datan. Yritys käytti koneoppimista työkaluna datan käsittelyssä, ja loi malleja, joilla he kohdistivat äänestäjiin poliittista mainontaa Trumpin hyväksi. Tässä tapauksessa puhutaan niin sanotusta mikrokohdentamisesta eli kerätyn datan ja tekoälyn avulla voitiin jopa yksilötasolla luoda täsmällistä henkilökohtaista mainontaa, tässä tapauksessa poliittista. (Coeckelbergh, 2024/2024, ss. 13, 59)

Coeckelbergh (2024/2024, s. 60) toteaa tekstissään: ”Nykyään Cambridge Analytica on oppikirjaesimerkki poliittisesta aineistoanalytiikan ja tekoälyn (väärin)käytöstä. Se osoittaa, miten tekoäly voi vaikuttaa suoraan politiikkaan, jopa kirkkaassa päivänvalossa järjestelmässä, joka kutsuu itseään demokraattiseksi.”

Tekoälyn ja sen etiikan monet ulottuvuudet osoittavat sen, kuinka tärkeää on yhteisesti tarkastella teknologian vaikutuksia sekä yhteiskunnan sekä yksilöiden näkökulmasta. Tekoäly tuo kiistatta mukanaan työelämään vaikuttavia asioita. Se tuhoaa työpaikkoja korvaamalla ihmisen säännöllisten ja toistuvien työtehtävien suorittamisessa. Tekoäly tuo myös mukanaan riskejä liittyen yksityisyyden suojaan ja jopa demokratian horjumiseen.

Erilaisten eettisten kysymyksien esille nostaminen on tärkeää, ja niiden ratkaiseminen vaatii monialaista yhteistyötä tekoälyteknologioiden kehittäjien, lainsäätäjien sekä ihmisten kesken.

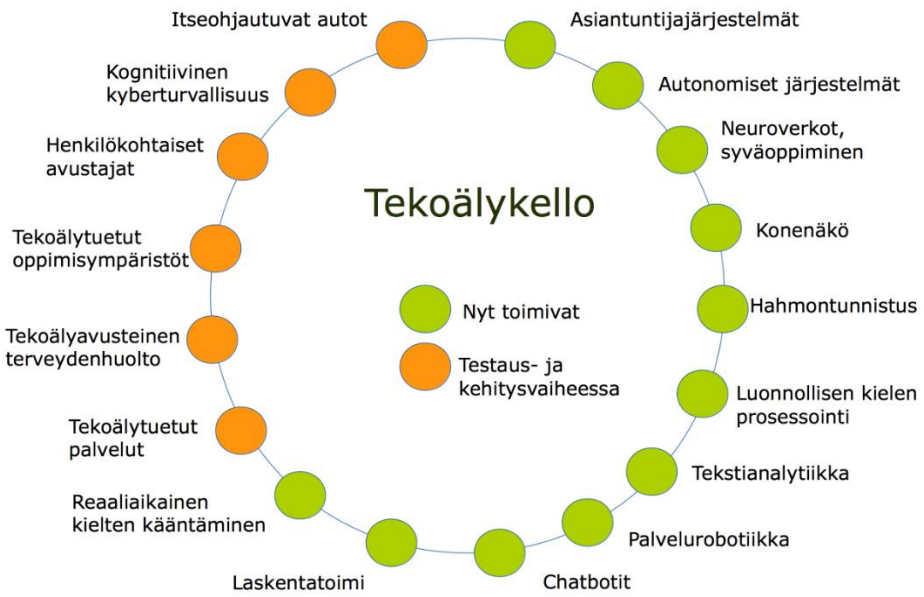
2.3 Tekoälyn tulevaisuus

Tätä opinnäytetyötä kirjoittaessa 2025 alkuvuodesta, tekoälyboomi on korkealla. Suuret globaalit AI-yritykset, kuten NVIDIA ja OpenAI viitoittavat tietä. Yhdysvaltain presidentin Donald Trumpin hallinto julkaisi 21. tammikuuta 2025 tiedotteen, jossa kerrotaan Stargate nimisen tekoäly-yrityksen perustamisesta. Yrityksen perustamisen takana on useita suuria tietotekniikkayrityksiä kuten aiemmin mainittu NVIDIA, OpenAI sekä Microsoft. Perustajayritysten kerrotaan tekevän alkusijoituksen 100 biljoonaa Yhdysvaltain dollaria kukin, ja lopullisen sijoitettavan rahaosuuden uskotaan olevan noin 500 biljoonaa Yhdysvaltain dollaria. (Duffy, 2025)

USA on tituleerannut itsensä johtavaksi AI-mahdiksi. Edellä mainittu Stargate-projekti mukaan lukien, myötäilee tätä titteliä. 20. päivä tammikuuta 2025, Kiinasta kuului kuitenkin jotain. DeepSeek-niminen AI-startup yritys julkaisi sarjan kielimalleja, jotka ohittivat kilpailevat mallit tehokkuudessa suoriutumisen ja kustannuksien osalta. DeepSeekin julkaisema laaja kielimalli (eng. Large Language Model, LLM) R1 syrjäytti kovimman kilpailijan OpenAI:n laajan kielimallin ChatGPT:n Applen App Storen ladatuimpien sovelluksien listalla. (Cui & Yang, 2025)

Kuva 2 näyttää tekoälyn kehityskaaren, ja missä tällä hetkellä kehityskaarta olemme. Tekoäly on tähän päivään mennessä jo todella kehittynyt ja se kehittyy koko ajan valtavin harppauksin. Tulevaisuudessa AI:n odotetaan integroituvan koko ajan vain syvemmälle jokaisen ihmisen tavalliseen elämään. Yhteiskunnallisesti on kuitenkin tärkeää, että tekoälyä ja sen kehitystä valvotaan, ja tarvittaessa rajoitetaan eri keinoin.

Kuva 2. Tekoälykello (Lehto ym., 2019)



3 Koneoppiminen

Koneoppiminen (eng. Machine Learning, ML) on tekoälyn osa-alue, jossa kone oppii itsenäisesti datasta, eikä sitä tarvitse erikseen ohjelmoida (SAP, n.d.). Koneoppimisessa konetta opetetaan valikoidun datan perusteella tekemään valikoitua tehtävää, esimerkiksi puulajien tunnistamista. Koneoppimisalgoritmeja ohjataan ihmisen toimesta etsimään malleja ja vastaavanlaisuuksia datasta, jonka perusteella kone rakentaa erilaisia malleja ja tätä kautta luo ennusteita (Klusaitė, 2023). Koneoppimisella pyritään jäljittelemään ihmistä niin havainnoimisessa kuin päätöksenteossa.

Koneoppiminen ei suinkaan ole uusi innovaatio. Jo vuonna 1959 Samuel Arthur kirjoitti ja julkaisi artikkelin IBM Journal of Research and Development -lehdessä. Artikkeli käsitteli koneoppimista, ja sen hyödyntämistä perinteisessä Tammi-pelissä. Arthur kirjoitti artikkelissaan koneoppimisesta, ja sen hyödyntämisestä pelissä tarkoituksenaan osoittaa, että tietokone voidaan koneoppimisen avulla opettaa pelaamaan Tammea paremmin kuin ihminen. (Theobald, 2017, s. 8)

Ensimmäiset roskapostisuodattimet hyödynsivät koneoppimista jo 1990-luvulla. 1990-luvulta aina näihin päiviin asti, roskapostisovellukset ovat todella edistyneitä, mikä johtuu täysin siitä, että nämä sovellukset ovat käsitelleet suuren määrän dataa eli sähköpostiviestejä useiden vuosien ajan, ja ovat oppineet datan perusteella tunnistamaan mikä on roskapostia, ja mikä ei ole. (Géron, 2023)

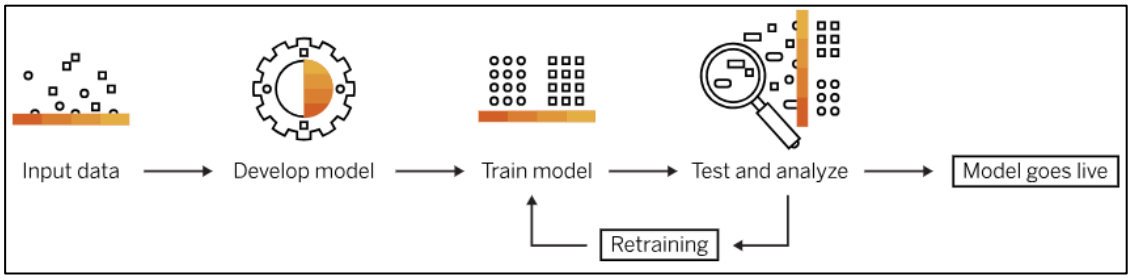
Nykymaailmassa koneoppiminen on hyvin paljon läsnä arjessamme. Hyvänä esimerkkinä voidaan esiin nostaa nykypäivän mainonta, ja vielä tarkemmin sanottuna kohdistettu mainonta, jossa koneoppimista hyödynnetään. Esimerkiksi lomamatkaa tulevalle kesälle suunnittelevalle henkilölle luultavimmin tulee eri matkatoimistojen mainoksia internettiä selatessa ja uutta talvitakkia verkkokaupoista etsivälle tulee jatkuvasti talvitakkimainoksia, vaikka hän ei juuri sillä hetkellä olisi missään verkkokaupassa. Kaikesta mitä teemme Internetissä kerätään jatkuvasti tietoa. Koneoppimisen avulla voidaan analysoida erittäin tarkasti ihmisten eri kulutustottumuksia, ja kohdistaa heihin tarkasti valittua mainontaa. (Klusaitė, 2023)

Kuva 3 näyttää koneoppimisprosessin kaikki eri vaiheet, jotka alkavat datan syöttämisestä koneoppimismallille, ja päättyvät lopullisen koneoppimismallin käyttöönottoon. Prosessi alkaa datan keräämisellä, joka on koko prosessin tärkein vaihe. Vaikka data esikäsitellään, sen tulee olla tarpeeksi laadukasta, koska muuten malli voi oppia heikosti, joka johtaa siihen, että lopullinen malli ei toimi halutulla tavalla.

Datavaiheen jälkeen koneoppimismalli kehitetään, ja koulutetaan valikoidun koulutusdatan perusteella. Koneoppimismallin luomis-, ja koulutusvaiheen jälkeen malli testataan valikoidulla testidatalla. Mallia analysoidaan eri mittareilla, jotta mallin suorituskykyä ja tarkkuutta voidaan tutkia ja arvioida onko se riittävällä tasolla.

Koneoppimismalli, jonka suorituskyky on asetettujen kriteerien mukainen, otetaan lopuksi käyttöön tuotantoympäristössä. Joissain tilanteissa on myös mahdollista, että koneoppimismallia tulee kouluttaa uudelleen. Tämänkaltainen tilanne voisi tulla esimerkiksi vastaan, jos mallin testaamisen ja analysoinnin tulokset eivät ole tarpeeksi hyvät. Mahdollista on, että mallin käyttökohteen olosuhteet muuttuvat merkittävällä tavalla, joka voisi vaikuttaa mallin suorituskykyyn. Esimerkiksi, jos tuotantolinjalla havaitaan kokonaan uudenlaisia tuotteita, alkuperäinen malli ei välttämättä enää pysty tunnistamaan kaikkia mahdollisia objekteja.

Kuva 3. Koneoppimisprosessi (SAP, n.d.)



3.1 Koneoppimisen osa-alueet

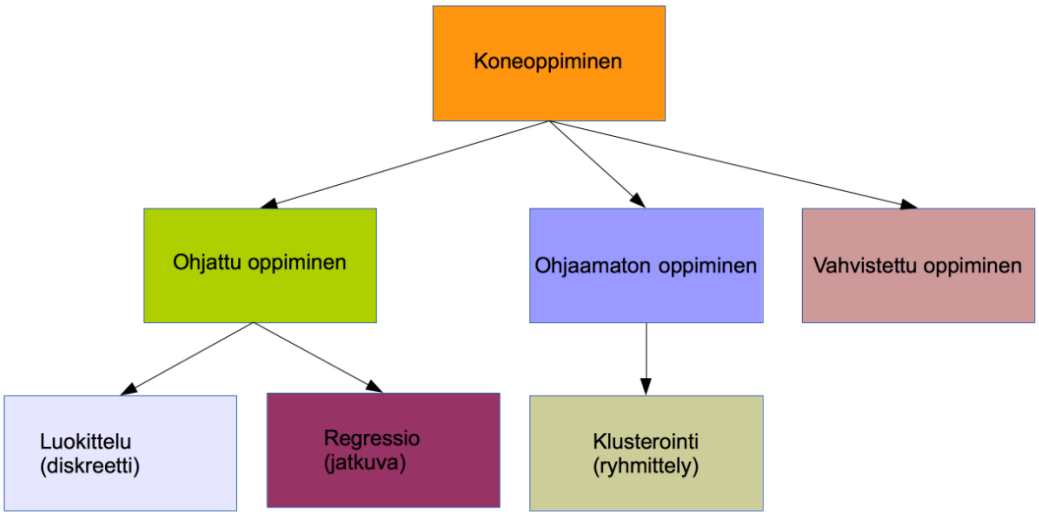
Koneoppiminen voidaan pilkkoa kolmeen eri osaan (Kuva 4). Nämä kolme osaa ovat ohjattu oppiminen, ohjaamaton oppiminen ja vahvistettu oppiminen. Näitä eri osia erottaa niiden erilaiset tavat oppia datasta. (Lehto ym., 2019)

Ohjatussa oppimisessa (eng. Supervised Learning) koneella on tiedossa sille tuleva syötedata sekä, mitä eri luokkia (labels) datasta löytyy, eli mitä datasta tulisi lopuksi ennustaa. Ohjatussa oppimisessa kone tunnistaa syötedatasta tietynlaisia kuviota, yhtäläisyyksiä ja säännönmukaisuuksia, jonka perusteella se rakentaa koneoppimismallin, jolla voidaan ennustaa uutta ennennäkemätöntä dataa. (Theobald, 2017, s. 18) Ohjattu oppiminen jakautuu vielä luokitteluun ja regressioon datan perusteella. Jos data voidaan luokitella tiettyihin ryhmiin, kyseessä on luokittelu. Luokittelutehtäviä on esimerkiksi sähköpostiviestien luokittelu roskapostiin ja läpi päästettäviin viesteihin. Regressiolla tarkoitetaan niin sanottua jatkuvaa dataa eli esimerkiksi lämpötilan tai pörssikurssien määrittelyä. (Lehto ym., 2019)

Ohjaamaton oppiminen (eng. Unsupervised Learning) eroaa ohjatusta oppimisesta siten, että siinä kone ei saa syötteeksi muuta kuin datan. Eli koneella ei ole tiedossa datasta ennustettavia luokkia samalla tavalla kuin ohjatussa oppimisessa. Ohjaamattomassa oppimisessa kone etsii ja tunnistaa syötedatasta yhtäläisyyksiä. (Alpaydin, 2020, s. 33) Kone pyrkii luokittelemaan yksittäiset syötedatat samaan joukkoon samankaltaisten yksittäisten syötedatojen kanssa (klusterointi), niin että tietyn datajoukon ominaisuudet ovat samankaltaisempia saman ryhmän muiden syötedatojen kanssa kuin toisen datajoukon. Tämä koneoppimisen oppimismuoto jäljittelee hyvin pitkälle ihmisen toimintaa ja oppimista (Lehto ym., 2019).

Vahvistetussa oppimisessa (eng. Reinforcement Learning) kone oppii ja kehittää itseään aiemmista iteraatioista saadun palautteen perusteella. Tämä oppimistyyli eroaa ohjatusta- ja ohjaamattomasta oppimisesta, joissa molemmissa koneoppimisprosessi päättyy siihen, että malli rakennetaan perustuen koulutus- ja testidataan. (Theobald, 2017, ss. 24–25) Vahvistetun oppimisen aikana kone saa jatkuvasti ympäristöltä palautetta, joko negatiivista tai positiivista. Prosessin aikana kone tekee itsenäisiä päätöksiä niin, että positiivinen palaute koko ajan lisääntyy ja negatiivinen vähenee. (Lehto ym., 2019)

Kuva 4. Koneoppimisen osa-alueet (Lehto ym., 2019)



3.2 Syväoppiminen ja neuroverkot

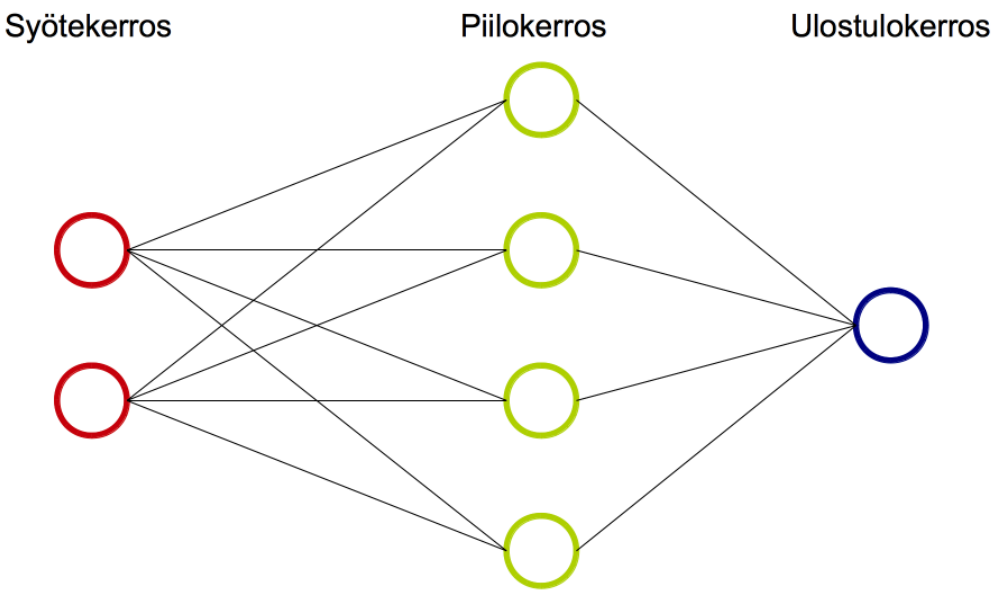
Syväoppiminen (eng. Deep Learning) on osa koneoppimista. Syväoppimisen perusperiaate on, että siinä pyritään mukailemaan ihmisaivojen toimintaa, ja varsinkin sitä miten ne prosessoivat tietoa (Viitaila, 2018). Termi syväoppiminen tulee siitä, että neuroverkoissa on todella paljon kerroksia. Nämä syvät neuroverkot ovat monikerroksisia verkkoja, jotka pystyvät tunnistamaan ja muodostamaan monimutkaisia piirteitä datasta. (Lehto ym., 2019)

Neuroverkko tai keinotekoinen neuroverkko (eng. Artificial Neural Network, ANN) tarkoittaa koneoppimismallia, joka jäljittelee ihmisaivojen oppimisprosessia. Neuroverkot on suunniteltu tunnistamaan ja oppimaan datassa piilevien kuvioiden avulla. Kuvioita hyödynnetään monenlaisissa tehtävissä, kuten luokittelussa ja regressiossa.

Neuroverkkojen käytön haasteena on se, että data on muunnettava numeeriseen muotoon ennen kuin neuroverkko voi vastaanottaa sitä. (Artasanchez & Joshi, 2020, ss. 469–470)

Neuroverkko koostuu syötetasosta, yhdestä tai useammasta piilotasosta ja ulostulotasosta kuvan (Kuva 5) mukaisesti. Jokainen taso sisältää solmuja. Solmuissa suoritetaan operaatioita, kuten laskutoimituksia, ja ne ovat yhteydessä toisiinsa. Signaalit liikehtivät kuvan (Kuva 5) mukaisesti vasemmalta syötekerroksesta sääntöjen mukaan, piilokerroksen tai kerroksien kautta ulostulokerrokseen. Signaalit voivat liikkua neuroverkossa takaisinpäin. Neuroverkkojen oppimiskyky pohjautuu niiden itseorganisoitavuuteen. Syötteiden kulkiessa neuroverkon läpi useita kertoja verkon eri yhteydet mukautuvat, jolloin se tunnistaa syötedatasta yhteyksiä, jonka perusteella se pystyy muodostamaan kaavoja. (Suomen Koodikoulu, 2019, ss. 21–22)

Kuva 5. Neuroverkko (Lehto ym., 2019)



3.3 Konenäkö

Konenäöllä (eng. Machine Vision) tarkoitetaan automatisoitua tiedon keräämistä kuvista tai videoista. Konenäköä yleensä valjastetaan objektien tunnistus- ja luokittelutehtäviin, joissa tarkoituksena on saada kone toimimaan ihmisen tavoin, eli tunnistamaan kohde ja luokittelemaan se silmänräpäyksessä. Tavoitteena konenäön sovelluksissa on mallintaa ja

automatisoida kone suorittamamaan ihmisen havainnointi- ja näköaistin toimintaa. (Suomen Koodikoulu, 2019, s. 29)

Konenäköjärjestelmää rakennettaessa täytyy ottaa huomioon monta erilaista asiaa. Järjestelmä koostuu useista komponenteista, joilla kaikilla on tärkeä rooli, jotta automatisoitu konenäköjärjestelmä saadaan toimimaan halutulla tavalla. Lehto ym. (2019) kertoo kirjassaan: ”Konenäköjärjestelmä koostuu valonlähteestä, kohteesta, kamerasta, tietokoneesta ja siinä toimivassa kuvankäsittelyohjelmasta, joka tulkitsee kuvan automaattisesti.”

3.4 Koneoppimismallit objektintunnistukseen

Tässä alaluvussa käsitellään mitä koneoppimismallit ovat ja mitä erityispiirteitä malleissa tulee ottaa huomioon, kun niitä käytetään objektintunnistamisessa. Koneoppimisen yhteydessä puhutaan usein eri malleista. Koneoppimismalleja objektintunnistukseen on olemassa useita erilaisia. Etsimisen ja tutkimisen jälkeen, tämän opinnäytetyön koneoppimismalleiksi valikoituivat seuraavat mallit, jotka esitellään tulevissa alaluvuissa.

3.4.1 YOLO11

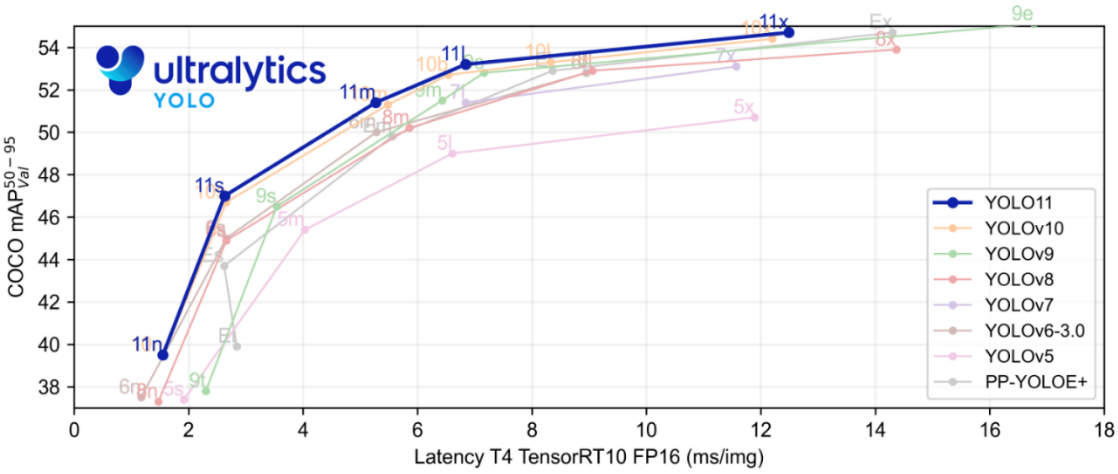
YOLO (You Only Look Once) on avoimeen lähdekoodiin perustuva objektintunnistusmallisarja. Ensimmäisen kerran YOLO-konsepti tuotiin julkiseksi vuonna 2016 Joseph Redmonin toimesta. Jo ensimmäisessä versiossaan YOLO oli paljon muita sen aikaisia objektintunnistusmalleja nopeampi sekä tarkempi, ja siitä tuli melko nopeasti konenäköprojektien standardi. (Kejriwal, 2023)

YOLO-mallit yhdistävät objektintunnistuksen sekä objektinluokittelun yhtenäiseen neuroverkkoarkkitehtuuriin. Tällä tavoin malli pystyy käsittelemään nämä molemmat tehtävät tehokkaasti samanaikaisesti. YOLO-mallien arkkitehtuuri voidaan jakaa kolmeen keskeiseen osaan, jotka ovat backbone, neck sekä head. **Backbone** toimii malleissa peruspiirteiden erottimena. Se käyttää konvoluutioneuroverkkoja, ja muuntaa niiden avulla raakakuvadataa usean mittakaavan piirrekartoiksi. **Neck**-osa toimii malleissa väliprosessointivaiheena, jossa se yhdistää ja parantaa näitä piirteitä eri mittakaavoilla erikoistuneiden kerrosten avulla. **Head**-osassa tapahtuu ennustaminen. Siellä tuotetaan

lopulliset tulokset, joiden perusteella objektien sijainnin määrittäminen ja luokittelu tapahtuu. (Khanam & Hussain, 2024, s. 3)

YOLO-sarjan viimeisin versio on Ultralyticsin kehittämä YOLOv11-malli. Se julkaistiin syyskuussa 2024. Malli käyttää syviä neuroverkkoja paikantamaan ja tunnistamaan objekteja kuvista reaaliajassa. YOLOv11-malii tuo markkinoille mukanaan merkittäviä parannuksia aikaisempaan YOLOv10-malliin verrattuna. Mallin uudistettu ”neck” ja ”backbone” -rakenne maksimoi objektien tunnistustarkkuuden sekä mahdollistaa vaativienkin tehtävien suorittamisen tehokkaasti. Mallin päätösaika on 2 % nopeampi kuin aikaisemmalla mallilla, mikä tekee siitä monikäyttöisen useille eri alustoille. (Mukherjee, 2024) Kuva 6 näyttää aikaisempien eri YOLO mallien suorituskyvyt, ja miten uusin malli vertautuu aikaisempiin verrattuna. Y-akselilla käytetään mittarina COCO $mAP_{50:95}$, jolla havainnoidaan eri YOLO-mallien tarkkuutta eri IoU-arvoilla välillä 0,5–0,95. Mitä korkeampi arvo, sitä parempi tunnistustarkkuus. X-akselilla kuvaajassa on Latency T4 TensorRT 10 FP16, jolla kuvataan mallin nopeutta tunnistaa yksi kuva. Mitä pienempi arvo, sitä tehokkaammin malli kuvan tunnistaa.

Kuva 6. YOLO mallien vertailu (Ultralytics, n.d.b)



YOLO11-malli tarjoaa mahdollisuuden suorittaa useita eri konenäköön liittyviä tehtäviä. Taulukko 1 näyttää YOLO-koneoppimismallin pääkäyttötarkoitukset oikean elämän esimerkkien tukemana. (Khanam & Hussain, 2024, s. 5)

Taulukko 1. YOLO11-mallin eri käyttötarkoitukset

Tehtävä	**Esimerkki**
Objektin tunnistaminen	Malli tunnistaa ja paikantaa objektit kuvasta sekä videosta rajauslaatikoita (eng. bounding boxes) avuksi käyttäen. Ominaisuutta voi käyttää valvontajärjestelmissä.
Instanssisegmentointi	Malli kykenee erottamaan yksittäiset objektit kuvasta tai videosta, jopa pikselitasolle asti. Tämä on hyödyllistä lääketieteessä erilaisten elimien ja kasvaimien tutkimuksissa.
Kuvien luokittelu	Malli voi luokitella kuvia eri kategorioihin ennalta määrättyjen luokkien perusteella. Tätä ominaisuutta voidaan käyttää verkkokaupoissa tuotteiden kategorisoimisessa.
Asennon arviointi	Malli kykenee havainnoimaan kuvasta tai videosta avainpisteitä (eng. key points), jonka perusteella voidaan havainnoida syötteen liikettä ja erilaisia asentoja. Tämä mahdollistaa esimerkiksi ihmisen vartalon tarkan analysoimisen urheilusuorituksen aikana.
Suunnattujen objektien tunnistaminen	Malli pystyy tunnistamaan objektit niiden orientaation eli kiertokulman perusteella. Tästä on hyötyä ilmakuvien analysoimisessa.
Objektien seuranta	Malli tunnistaa ja pystyy seuraamaan objektien liikettä kuvassa tai videossa. Tämä ominaisuus on hyödyllinen liikenteenvalvonnassa.

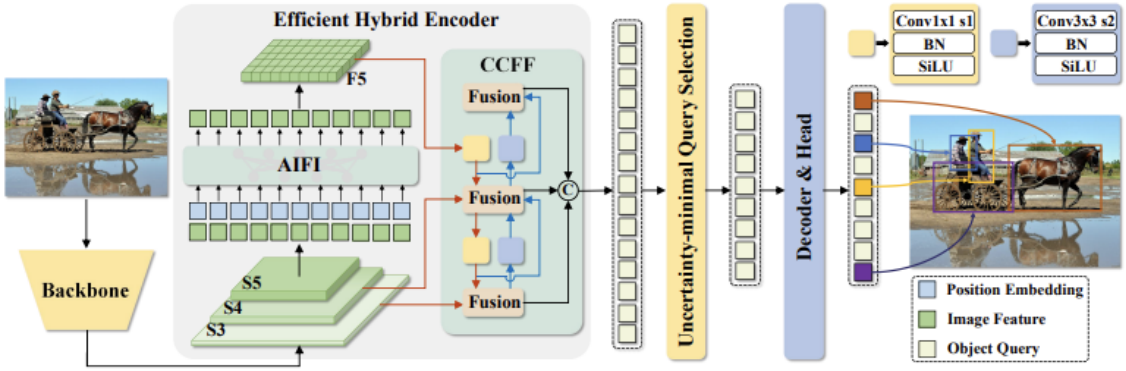
3.4.2 RT-DETR

RT-DETR eli Real-Time Detection Transformer on kiinalaisen monikansallisen teknologiayrityksen Baidun kehittämä koneoppimismalli objektintunnistukseen. Malli on kehitetty mukaillen DETR-ajattelumallia, jossa ei käytetä NMS eli Non-Maximum Supression -menetelmää mallin mahdollisten päällekkäisten ennustuksien poistamiseen. Malli siis oppii tunnistamaan objektin suoraan kuvasta ilman ylimääräisiä suodatuksia, joka tekee siitä nopean ja tehokkaan. (Ultralytics, n.d.a)

Mallin rakenne koostuu kolmesta osasta (Kuva 7), jotka ovat backbone, hybrid encoder ja transformer decoder. **Backbone** ottaa vastaan syötteen, poimii syötteestä keskeisiä piirteitä ja lähettää kolmen viimeisen vaiheen (S3, S4, S5) piirteet eteenpäin Hybrid Encoderille. **Hybrid Encoder** muuntaa nämä monitasoiset piirteet yksinkertaiseksi ja yhtenäiseksi syötteeksi mallin seuraavalle osalle. Tämän suorittamiseen malli käyttää Attention-based Intra-Scale Feature Interactionia (AIFI), joka analysoi piirteitä tietyn tason sisällä ja Cross-scale Feature Fusionia (CCFF), joka yhdistelee tietoja eri tasojen välillä.

Näiden vaiheiden jälkeen Uncertain-minimal Query Selection valitsee ennalta määritetyn lukumäärän hybrid encoderin eri piirteitä. **Transformer Decoder** osassa näitä piirteitä käytetään tunnistamaan ja kategorisoimaan annetut alkuperäiset syötteet. Decoder käyttää työkalunaan Auxiliary Prediction Headieja, jotka parantavat huonosti näkyvien kohteiden tunnistamista. (Zhao ym., 2024)

Kuva 7. RT-DETR-mallin rakenne (Zhao ym., 2024)



4 Konenäkö ja teollinen laadunvalvonta

Teollisuuden alojen tuotantomäärät ovat kasvaneet merkittävällä tavalla automaation kehityksen seurauksena. Samalla teollisuuden tuotteiden laadun ja niiden valvomisen kriteerit ovat tiukentuneet. Konenäkö tarjoaa kustannustehokkaan vaihtoehdon ihmissilmän suorittamaan laadunvalvontaan. (Provendor Oy, n.d.) Konenäköä voi soveltaa useilla eri teollisuuden aloilla taulukon (Taulukko 2) mukaisesti.

Taulukko 2. Konenäön ja laadunvalvonnan sovelluskohteet (Provendor Oy, n.d.)

Teollisuuden ala	**Käyttökohde-esimerkki**
Elektroniikkateollisuus	Tuotteiden laadunvalvonta
Elintarviketeollisuus	Elintarvikkeiden laadunvalvonta sekä lajittelu
Lääketeollisuus	Lääkkeiden lajittelu ja laadunvalvonta
Tekstiiliteollisuus	Kankaiden ja lankojen värien tulkinta ja laadunvalvonta
Koneteollisuus	Moottoreiden ja muiden osien seuranta
Metsäteollisuus	Sahatavaran laaduntarkkailu
Kumi- ja muoviteollisuus	Kumi- ja muovituotannon laadunvalvonta

Ihmisen korvaaminen koneella laadunvalvonnassa on erittäin hyödyllistä. Kone parantaa tuotantolinjan kriittisten parametrien mittaamisen, laadunvalvonnan, huoltotarpeiden tunnistamisen sekä muiden prosessien järjestelmällisyyttä, tarkkuutta ja kustannustehokkuutta. Kun teolliselta tuotantolinjalta saadaan koneen tuottamia tarkkoja tietoja, voidaan prosessit optimoida ja tätä kautta saavuttaa merkittäviä säästöjä kustannuksissa. (VTT, n.d.)

5 Opinnäytetyön menetelmät

Tässä luvussa käsitellään opinnäytetyössä käytettyjä ohjelmistoja ja kehitysympäristöä, projektinhallinnan toteutusta sekä tekoälyn roolia tässä opinnäytetyöprojektissa.

5.1 Käytetyt työkalut ja ohjelmistot

Koneoppimismallien luominen on tässä opinnäyteyössä toteutettu Python-ohjelmointikielellä. Python on suosittu ohjelmointikieli koneoppimisessa. Pythonin suosio perustuu sen selkeään syntaksiin eli kirjoitusasuun sekä laajoihin kirjastoihin. Kirjastoilla tarkoitetaan Python-ohjelmointikieleen valmiiksi sisäänrakennettuja eri funktioita, joita ohjelmistokehittäjät voivat vapaasti hyödyntää. Alla seuraavaksi esitellään muutamia yleisesti tunnettuja ja koneoppimisessa käytettyjä Python-kirjastoja (Taulukko 3).

Taulukko 3. Python-kirjastot koneoppimisessa

Kirjasto	**Käyttötarkoitus**
NumPy	Matriisilaskenta ja numeerinen analyysi
Pandas	Datan käsittely ja sen analysoiminen

Matplotlib / Seaborn	Datan visualisointi
scikit-learn	Koneoppimisalgoritmit
Tensorflow / Pytorch	Neuroverkkojen luominen ja niiden kouluttaminen
OpenCV	Kuvankäsittely ja konenäkö

Projektin kehitysympäristönä toimii työn toimeksiantajan tarjoama JupyterLab-palvelin. Palvelimella toteutetaan käytännössä koneoppimismallien luominen, kouluttaminen ja testaaminen. Mallien kouluttamisprosessien visualisoinnit toteutetaan Tensorboardia hyväksi käyttäen. Tensorboard on työkalu, joka mahdollistaa mallien oppimisen seuraamisen jopa reaaliaikaisesti. Lopulliset mallien suorituskykyjen tulokset visualisoidaan Pythonin Pandas- sekä Matplotlib-kirjastoja avuksi käyttäen.

5.2 Tekoälyn käyttö opinnäytetyössä

Opinnäytetyön tekemisen aikana tekoälyä on hyödynnetty useilla eri tavoilla. Ensimmäisenä ja suurimpana apuna tekoäly on ollut oikolukijana. Tekoälyn käytön avulla kirjoitetun tekstin oikeinkirjoitus sekä kielioppi on saatu varmistettua. Tekoäly on toiminut myös tiedonhankintamenetelmänä tieteellisten julkaisuiden rinnalla. Työn toiminnallisessa osassa eli koneoppimismallien luomisessa, kouluttamisessa ja testaamisessa tekoäly on ollut myös apuna Python-ohjelmakoodin kirjoittamisessa.

Vaikka opinnäytetyössä on käytetty tekoälyä apuna, kaikki työn sisältö on itse kirjoitettua eikä työssä ole käytetty suoraan tekoälyn generoimaa tekstiä. Tekoäly on toiminut työn kirjoittamisen ohessa hyödyllisenä apurina opinnäytetyöprosessin eri vaiheiden aikana.

Tekoälyn ja sen tarjoamien erilaisten työkalujen avulla on kyetty parantamaan tuotetun tekstin laatua. Tekoäly on siis ollut arvokas työkalu osana tätä opinnäytetyötä.

5.3 Projektinhallinta

Opinnäytetyön aikana pidetään viikoittain tapaamisia ohjausryhmän kesken. Ryhmään kuuluu opinnäytetyön ohjaaja sekä muita opinnäyteyötä tekeviä opiskelijoita. Tapaamisissa käsitellään avoimesti ryhmäläisten opinnäytetöitä, niiden aikataulua ja edistymistä sekä annetaan palautetta toisillemme, jonka avulla työtä voidaan kehittää.

Projektin toteutuksen aikana järjestetään viikoittaisia palavereita myös työn toimeksiantajan, Konekatse Oy:n edustajan kanssa. Näiden palavereiden tarkoituksena on tarkastella yhteisesti työn vaihetta, ja keskustella tulevista tavoitteista. Lisäksi toimeksiantaja tarjoaa mahdollisuutta saada tukea mahdollisiin työssä ilmenneisiin haasteisiin. Kommunikaatio opiskelijan ja toimeksiantajan välillä tapaamisten ulkopuolella tapahtuu Slack-viestintäalustan kautta, jonne on luotu oma työtila lopputyöhön liittyvää viestintää varten.

Opinnäytetyön aikana kirjoitetaan päiväkirjaa, johon kirjataan ylös päivittäin tehdyt työt, kehitysaskeleet ja mahdolliset ongelmat. Päiväkirjan avulla seurataan työn etenemistä yksityiskohtaisesti. Lisäksi päiväkirjan pitäminen tarjoaa mahdollisuuden kehittää työn tekijän kirjoitustaitoa asiantuntijakirjoittajana.

6 Koneoppimismallien kehittäminen objektintunnistukseen

Tässä luvussa esitellään opinnäytetyöhön valikoitujen koneoppimismallien kehittäminen ja lopullinen testaaminen. Luvussa esitellään myös data, jolla mallit koulutetaan ja testataan.

6.1 Data ja datan esikäsittely

Datan valitseminen ja sen esikäsittely on tämänkaltaisessa koneoppimisprojektissa ensimmäinen, ja samalla myös kaikista tärkein vaihe. Ilman kunnollista dataa työ luultavasti epäonnistuisi. Työn dataksi valikoitui toimeksiantajan suosituksesta Roboflow-verkkosivustolta saatavilla oleva avoin datasetti (Veerman, 2024). Datasetti koostuu kuvista

eri elintarvikkeista liukuhihnalla. Kuvia datasetistä löytyy yhteensä 3977 kappaletta. Kuvasta (Kuva 8) nähdään esimerkkikuvia datasta, jota työssä käytetään.

Kuva 8. Kuvakollaasi käsiteltävästä datasta

Dataa tarkasteltaessa ja sen kehitysympäristöön lataamisen jälkeen huomataan, että kaikki kuvat sijaitsevat yhdessä kansiossa ja että ne on nimetty epäselvästi. Datan käsittelyn selkeyttämistä ja helppokäyttöisyyttä ajatellen luotiin Python-ohjelmointikielellä muutama ohjelmakoodi. Ohjelmakoodi 1 uudelleennimeää tiedostot luettavampaan muotoon. Koodi määrittää kansiot, joissa halutut tiedostot sijaitsevat. Tämän jälkeen koodi hakee kaikki kuva sekä label tiedostot, ja lajittelee ne järjestykseen. Tämän jälkeen koodi tarkastaa, että kuva ja label tiedostojen kokonaislukumäärä täsmää. Jos lukumäärät täsmäävät, tiedostot uudelleennimetään. Kuvatiedostot nimetään logiikalla img_001.jpg, img_002.jpg, ja label tiedostot nimetään img_001.txt, img_002.txt. Lopuksi ohjelma tulostaa vielä viestin tiedostojen uudelleennimeämisen onnistumisesta.

Ohjelmakoodi 1. Datan uudelleennimeäminen Python-ohjelmointikielellä

```
import os

# määritä kansiot
image_dir = 'data/train/images'
label_dir = 'data/train/labels'
```

```
# hae kaikki kuvat ja merkintätiedostot
image_files = sorted([f for f in os.listdir(image_dir) if
f.endswith('.jpg')])
label_files = sorted([f for f in os.listdir(label_dir) if
f.endswith('.txt')])

# tarkista, että kuvien ja merkintätiedostojen määrä täsmää
assert len(image_files) == len(label_files), "Kuvien ja
merkintätiedostojen määrä ei täsmää!"

# uudelleennimeä tiedostot
for i, (image_file, label_file) in enumerate(zip(image_files,
label_files)):
new_name = f"img_{i+1:03d}"
new_image_name = f"{new_name}.jpg"
new_label_name = f"{new_name}.txt"

# uudelleen nimeä kuvatiedosto
os.rename(os.path.join(image_dir, image_file), os.path.join(image_dir,
new_image_name))

# uudelleen nimeä merkintätiedosto
os.rename(os.path.join(label_dir, label_file), os.path.join(label_dir,
new_label_name))

print(f"Uudelleen nimetty: {image_file} -> {new_image_name}")
print(f"Uudelleen nimetty: {label_file} -> {new_label_name}")

print("Tiedostojen uudelleen nimeäminen onnistui.")
```

Kuvien uudelleennimeämisen jälkeen, data jaetaan koulutus-, validointi sekä testausjoukkoihin. Liitteessä (Liite 2) luodaan Python-ohjelmakoodi, jossa data jaetaan 70/15/15 jaolla, jossa 70 prosenttia datasta toimii koulutusdatana. Ensimmäinen 15 prosentin osuus datasta toimii validointidatana. Toinen 15 prosenttia datasta toimii testausdatana, ja se siirretään koulutuksen ajaksi sivuun. Lopullinen mallien testaaminen suoritetaan tällä datalla, jota malli ei ole nähnyt koulutus-, tai validointivaiheessa. Tällä varmistetaan mallien suorituskyvyn mittaamisen oikeellisuus ja vertailukelpoisuus.

70/15/15 datajaon jälkeen luodaan viedä kolmas Python-ohjelmakoodi, jolla 70 prosentin koulutusdata jaetaan pienempiin osiin 10 prosentin välein. Tällä tavoin pystytään seuraamaan kuinka paljon koulutusdatan määrän vaihtelut vaikuttavat koneoppimismallien oppimiseen. Yksi toimeksiantajan määrittämistä tutkimuskysymyksistä oli, että kuinka pienellä datalla malli pystyy oppimaan, ja tällä tavoin lähdetään tutkimaan vastausta tähän kysymykseen.

6.2 Mallien kouluttaminen

Datan tutkimisen ja sen esikäsittelyn jälkeen edetään seuraavaan vaiheeseen koneoppimismallien luomisessa, mallien kouluttamiseen. Seuraavissa luvuissa käydään läpi tutkittavien koneoppimismallien kouluttamisprosessit.

6.2.1 YOLO11

YOLO11-koneoppimismalli on Ultralyticsin kehittämä malli, joten mallin lataaminen ja käyttöönottaminen onnistuu helposti Ultralyticsin YOLO-kirjaston kautta (Ohjelmakoodi 2).

Ohjelmakoodi 2. YOLO11-mallin lataaminen ja määrittely

```
from ultralytics import YOLO

model = YOLO("yolo11n.pt")
```

Mallin lataamisen jälkeen tulee kehitysympäristöön luoda data.yaml -tiedosto. YOLO11-malli tarvitsee tämän tiedoston mallin koulutusta varten. Tiedostossa tulee määritellä koulutus-, validointi- ja testidatan tiedostopolut sekä datasetin eri luokkien lukumäärä sekä niiden nimet. Ohjelmakoodi 3 näyttää työssä käytetyn data.yaml -tiedoston. Koodissa määritellään mistä tiedostopoluista löytyvät koulutus-, validointi- sekä testidata. Määritetään myös, että datasta löytyy kuusi eri luokkaa, ja minkä nimisiä ne ovat.

Ohjelmakoodi 3. data.yaml -tiedosto

```
train: train/images
val: valid/images
test: test/images

nc: 6
names: ['Bamiblok', 'Frikandel', 'Kaassouffle', 'Kipcorn', 'Kroket', 'Pikanto']

roboflow:
  workspace: niels-veerman
  project: pythonyolov5
  version: 7
  license: CC BY 4.0
  url: https://universe.roboflow.com/niels-veerman/pythonyolov5/dataset/7
```

Mallin koulutuksen tavoitteena on tutkia, kuinka pienellä datamäärällä YOLO-malli kykenee saavuttamaan hyvän suorituskyvyn objektientunnistamisessa. Kuten aiemmin kerrottiin, koulutusdata jaettiin 10 prosentin välein pienempiin datasetteihin. Koulutus toistetaan jokaisella datasetillä, jotta voidaan seurata mallin oppimista koulutuksen aikana.

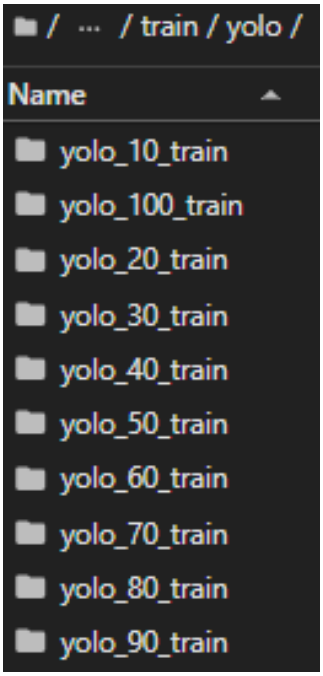
Ohjelmakoodi 4 näyttää, miten YOLO11-mallille tulee antaa hyperparametreja, ja miten koulutus saadaan aloitettua. Data-parametri ohjaa mallin data.yaml tiedostoon, josta löytyvät tiedostopolut dataan sekä tunnistettavat luokat. Epochit tarkoittavat montako kierrosta koulutus kestää, tässä tapauksessa 50 kierrosta. Patience-hyperparametri keskeyttää koulutuksen, jos mallin tarkkuus ei kehity 10 epochin aikana. Tämän avulla säästetään aikaa koulutuksen aikana. Batch määrittää montako kuvaa muistissa käsitellään kerralla, tässä tapauksessa käsitellään 16 kuvaa kerrallaan. Imgsz määrittää käsiteltävien kuvien koon pikseleinä, tässä tapauksessa koko on 640x640. Project-hyperparametri määrittää projektin kansion, johon jokaisen koulutuksen tulokset tallennetaan, tässä tapauksessa "runs/train/yolo" kansioon. Name-hyperparametri antaa nimen koulutusajolle, tässä tapauksessa "yolo_{ratio}_train". Tämä auttaa jatkotutkimuksissa sekä testiajojen suorittamisessa.

Ohjelmakoodi 4. YOLO11-mallin kouluttaminen

```
model.train(
            data="datasets/data.yaml",
            epochs=50,
            patience=10,
            batch=16,
            imgsz=640,
            project="runs/train/yolo",
            name=f"yolo_{ratio}_train"
        )
```

Mallin koulutuksen jälkeen ohjelma tallentaa jokaisen koulutussession tulokset omaan kansioon alla olevan kuvan (Kuva 9) mukaisesti. Tämä selkeyttää ja helpottaa tuloksien tarkastelua sekä mallin jatkokehitystä. Tämän jälkeen mallin koulutus on suoritettu.

Kuva 9. YOLO-mallin koulutuksen tuloksien kansiorakenne



Seuraavaksi suoritetaan jokaiselle koulutusdatajoukolle eli jokaiselle mallille validointi ohjelmakoodin (Ohjelmakoodi 5) mukaisesti. Validoinnilla tarkoitetaan mallien sen hetkisen suorituskyvyn arvioimista. Validointia varten luotiin alussa erillinen datajoukko, jota ei käytetty koulutuksessa. Tällä varmistetaan validoinnin tuloksien oikeellisuus. Validoinnilla varmistetaan mallien mahdolliset puutteet ja saadaan selville, jos mallia tulee kouluttaa uudelleen vielä ennen mallin lopullista testausta. Mallin testauksen tuloksista kerrotaan luvussa 7.

Ohjelmakoodi 5. YOLO-mallin validointi

```
from ultralytics import YOLO

yaml_path = "datasets/data.yaml"

model = YOLO("yolo11n.pt")

models = {ratio: f"runs/train/yolo/yolo_{ratio}_train/weights/best.pt"
for ratio in range(10, 110, 10)}

for ratio, model_path in models.items():
    print(f"Valitoidaan malli {ratio}% koulutusdatalla...")

    model = YOLO(model_path)
    model.val(
        data=yaml_path,
```

```
            batch=16,
            imgsz=640,
            project="runs/val/yolo",
            name=f"yolo_{ratio}%_val"
        )

    print("Kaikki mallit validoitu!")
```

6.2.2 RT-DETR

RT-DETR-koneoppimismallin käyttöönotto onnistuu samalla tavoin kuin YOLO11-mallin eli Ultralytics-kirjaston avulla (Ohjelmakoodi 6).

Ohjelmakoodi 6. RT-DETR lataaminen ja määrittely

```
from ultralytics import RTDETR

model = RTDETR("rtdetr-l.pt")
```

Ohjelmakoodi 7 näyttää miten RT-DETR-mallille tulee antaa hyperparametreja, ja miten mallin koulutus saadaan aloitettua. RT-DETR-malli voi käyttää hyväksi myös data.yaml tiedostoa, joten käytetään mallin koulutuksessa samaa yaml-tiedostoa kuin YOLO11-mallilla (Ohjelmakoodi 3). Tiedoston avulla malli saa tiedon datasta, eli mitä dataa on ja missä se sijaitsee. Epochit tarkoittavat montako kierrosta koulutus kestää, tässä tapauksessa 50 kierrosta. Patience-hyperparametri keskeyttää koulutuksen, jos mallin tarkkuus ei kehity 10 epochin aikana. Tämän avulla säästetään aikaa koulutuksen aikana. Cache-hyperparametri määrää, käytetäänkö välimuistia koulutusdatan lataamiseen muistiin nopeuttamaan koulutusta; tässä tapauksessa välimuistia ei käytetä, koska sen arvo on False. Tämä saattaa hidastaa koulutusta, mutta estää kernelin kaatumisen. Batch määrittää montako kuvaa muistissa käsitellään kerralla; tässä tapauksessa käsitellään 4 kuvaa kerrallaan. Tämä asetus hidastaa myös koulutusta, mutta ennaltaehkäisee kernelin kaatumista. Imgsz määrittää käsiteltävien kuvien koon pikseleinä, tässä tapauksessa koko on 640x640. Project-hyperparametri määrittää projektin kansion, johon jokaisen koulutuksen tulokset tallennetaan, tässä tapauksessa "runs/train/re-detr" -kansioon. Name-hyperparametri antaa nimen koulutusajolle, tässä tapauksessa "rt-detr_{ratio}_train". Tämä auttaa jatkotutkimuksissa ja testiajojen suorittamisessa.

Ohjelmakoodi 7. RT-DETR-mallin kouluttaminen

```
model.train(
            data="datasets/data.yaml",
            epochs=50,
```

```
        cache=False,
        patience=10,
        batch=4,
        imgsz=640,
        project="runs/train/rt-detr",
        name=f"rt-detr_{ratio}_train"
    )
```

Mallin kouluttamisen jälkeen siirrytään seuraavaan vaiheeseen eli validointiin. Validoinnilla varmistetaan mallin suorituskyky. Suorituskykyä mitataan eri mittareilla. Validoinnilla varmistetaan, että malli on valmis lopulliseen testaukseen. Mallin validoinnin jälkeen, jos tulokset ovat hyviä siirrytään seuraavaan ja viimeiseen vaiheeseen eli mallin testaukseen. Testauksen tuloksista kerrotaan seuraavassa luvussa.

7 Tulokset ja mallien vertailu

Koneoppimismalleja voidaan vertailla useilla eri mittarilla. Tässä opinnäytetyössä malleja vertaillaan neljällä eri mittarilla. Mittareista kerrotaan tarkemmin OpenAI:n tekoälysovelluksen ChatGPT:n luomassa taulukossa (Taulukko 4).

Taulukko 4. Koneoppimismallien suorituskykymittarit (OpenAI, 2022)

Mittari	Selite
mAP50-95 (Mean Average Precision @ 50–95% IoU)	Arvioi mallin tarkkuutta eri osumakriteereillä käyttämällä IoU-arvoja **50 % - 95 %** (kasvuväli 5 %). Tämä antaa kattavan kuvan mallin suorituskyvystä eri tarkkuusvaatimuksilla. Korkea arvo osoittaa, että malli tunnistaa ja paikantaa objektit tarkasti sekä **löysemmillä** että **tiukemmilla** osumakriteereillä.
F1-score	Mittaa mallin tasapainoa **precisionin (tarkkuus)** ja **recallin (herkkyys)** välillä. Korkea F1-arvo tarkoittaa, että malli **löytää mahdollisimman paljon kohteita** tekemättä liikaa virheitä. Hyvä F1-score kertoo, että mallilla on sekä korkea tarkkuus että korkea herkkyys.
Inference Time (ms)	Kuvaa, kuinka kauan mallilta kestää käsitellä yksi kuva millisekunneissa. Pienempi arvo tarkoittaa nopeampaa suorituskykyä, mikä on tärkeää **reaaliaikaisissa sovelluksissa**, kuten videonkäsittelyssä ja teollisessa tarkastuksessa.
FPS (Frames Per Second)	Ilmaisee, kuinka monta kuvaa malli pystyy käsittelemään sekunnissa. Suurempi arvo tarkoittaa **nopeampaa ja tehokkaampaa mallia**, mikä on erityisen tärkeää sovelluksissa, kuten **reaaliaikainen kuvantunnistus ja videoanalyysi**.

Seuraavaksi alla on kaksi taulukkoa (Taulukko 5 ja Taulukko 6), joissa ensimmäisessä on RT-DETR-mallin testauksen tulokset jokaisella mitatulla mittarilla, ja jokaisella eri

datamäärän parhaalla mallilla. Jälkimmäisessä taulukossa on YOLO-mallin kaikki tulokset. Tulokset on pyöristetty neljän desimaalin tarkkuuteen. Tuloksia vertaillaan taulukoiden jälkeen visuaalisesti eri graafien avulla, joka helpottaa tuloksien ymmärtämistä ja yleisesti mallien vertailua.

Taulukko 5. RT-DETR-mallin testauksen tulokset

Datasetin koko	mAP50-95	F1-score	Inference Time (ms)	FPS
10% (278 kuvaa)	0.9106	0.994	19.3889	51.576
20% (556 kuvaa)	0.9182	0.9974	19.3224	51.7534
30% (834 kuvaa)	0.908	0.9956	19.3432	51.6977
40% (1113 kuvaa)	0.9113	0.9951	19.2943	51.8287
50% (1391 kuvaa)	0.9055	0.9936	19.2706	51.8926
60% (1669 kuvaa)	0.9202	0.9954	19.3429	51.6985
70% (1948 kuvaa)	0.9114	0.9955	19.3109	51.7841
80% (2226 kuvaa)	0.9153	0.9948	19.3548	51.6667
90% (2504 kuvaa)	0.9172	0.9951	19.3135	51.7772
100% (2783 kuvaa)	0.9169	0.9909	19.4605	51.3862

Taulukko 6. YOLO11-mallin testauksen tulokset

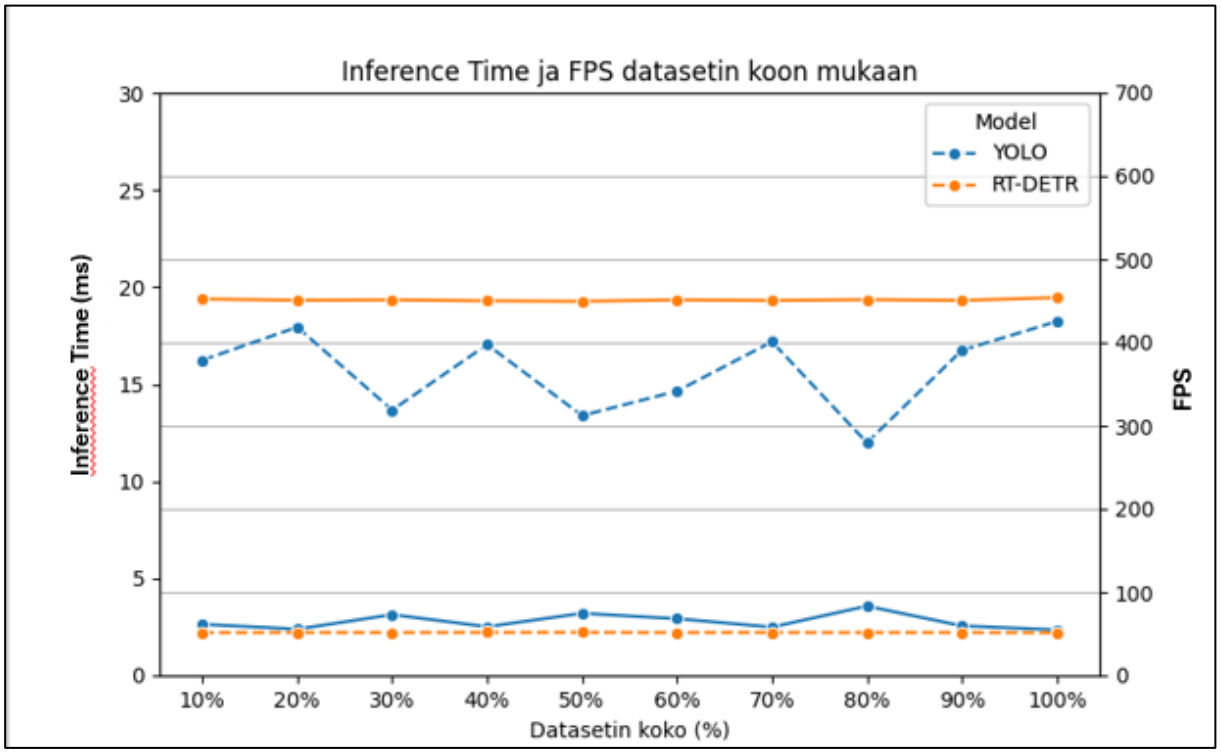
Datasetin koko	mAP50-95	F1-score	Inference Time (ms)	FPS
10% (278 kuvaa)	0.8911	0.9968	2.6393	378.8911
20% (556 kuvaa)	0.9081	0.9975	2.3889	418.6036
30% (834 kuvaa)	0.9008	0.9975	3.1352	318.9604
40% (1113 kuvaa)	0.9034	0.9976	2.5151	397.6036
50% (1391 kuvaa)	0.906	0.9977	3.2025	312.2584
60% (1669 kuvaa)	0.9002	0.9977	2.9264	341.7122
70% (1948 kuvaa)	0.9286	0.9977	2.4939	400.9753
80% (2226 kuvaa)	0.9216	0.9985	3.5744	279.7668
90% (2504 kuvaa)	0.9201	0.9976	2.559	390.7741
100% (2783 kuvaa)	0.9356	0.9976	2.3506	425.4218

Kuva 10 on visuaalinen graafi, jossa verrataan mallien suorituskykyä kahdella eri mittarilla: Inference Time (ms) eli kuinka kauan mallilla kestää tehdä ennuste yhdelle kuvalle ja FPS (Frames Per Second) eli kuinka monta kuvaa malli pystyy käsittelemään sekunnissa. Kuvaajassa katkoviivat visualisoivat FPS:ää ja yhtenäiset viivat Inference Timea. FPS-mittaria tutkiessa, mitä suurempi arvo sitä parempi suorituskyky mallilla on. Inference Time-mittaria tutkiessa, mitä pienempi arvo on, sitä tehokkaampi malli on kyseessä.

Oranssi yhtenäinen viiva kuvaa RT-DETR-mallin Inference Timen kehitystä datasetin kasvaessa. Arvo pysyy tasaisena noin 19 millisekunnissa datasetin koon muutoksista huolimatta. Oranssi katkoviiva kuvaa RT-DETR-mallin FPS-arvojen vaihteluita. FPS-arvo pysyy myös tasaisena datasetin koosta riippumatta noin 51 FPS:ssä.

Kuvaajan sininen yhtenäinen viiva kuvaa YOLO-mallin Inference Timen kehitystä datasetin määrän muuttuessa. Arvo pysyy melko tasaisena kaikilla datasetin määrillä. Vaihteluväli on 2,3506–3,5744. Sininen katkoviiva kuvaa YOLO-mallin FPS-arvoja eri datamäärillä. Arvot vaihtelevat noin 280 ja 425 FPS:n välillä.

Kuva 10. Mallien Inference Timen (ms) ja FPS:n vertailu

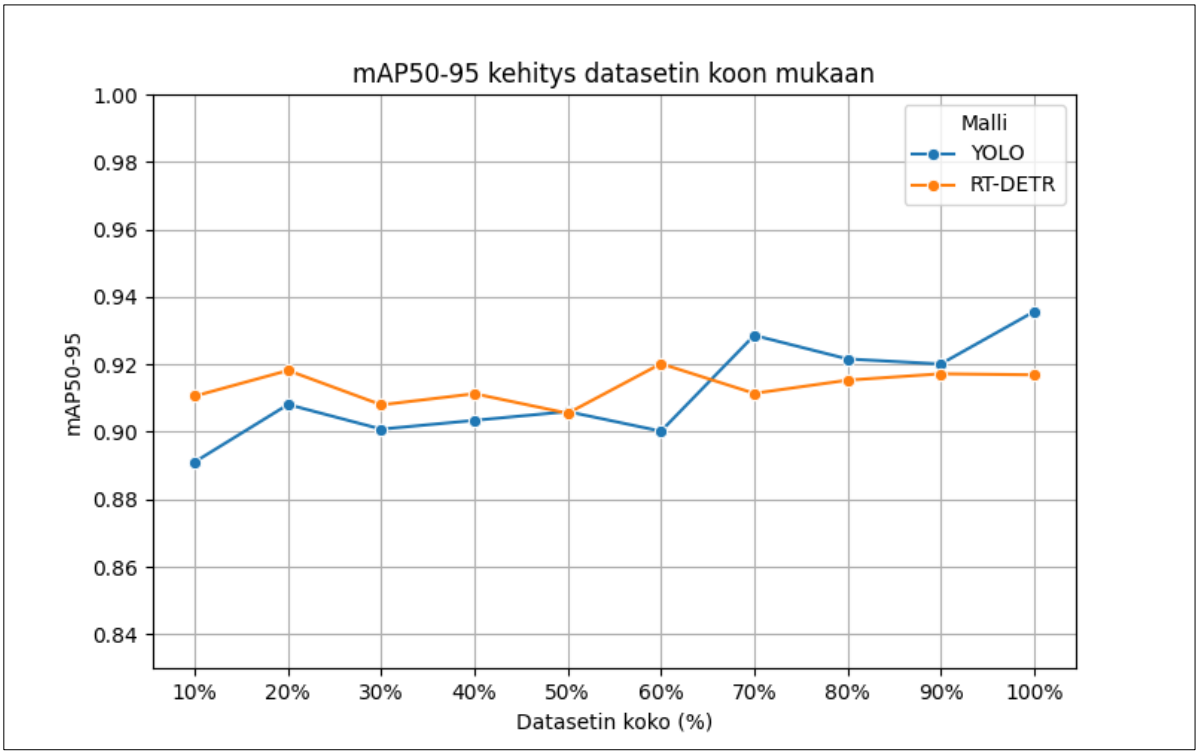


Kuva 11 havainnollistaa mallien mAP50-95 arvojen kehitystä datasetin eri määrillä. mAP-50-95 käyttää useamman eri IoU-arvon keskiarvotarkkuutta 50 % ja 95 % IoU arvojen väliltä viiden prosentin välein. IoU (Intersection over Union) tarkoittaa mittaria, jolla mitataan koneoppimismallin ennustaman alueen ja todellisen alueen päällekkäisyyttä. IoU voi saada arvoja 0 ja 1 väliltä. Mitä suurempi sen arvo sen tarkempi ennustettu alue on.

RT-DETR-malli suoriutuu YOLO-mallia paremmin pienemmillä koulutusdatamäärillä. Toisaalta koulutusdatamäärien kasvaessa RT-DETR mAP50-95 pisteytys ei kasva samoissa määrin kuin YOLO-mallilla. YOLO-malli onkin suuremmilla koulutusdatamäärillä tarkempi kuin RT-DETR.

RT-DETR-malli suoriutuu kiitettävästi jo pienillä koulutusdatamäärillä. Mallin sorituskyky ei myöskään merkittävästi parane datasetin koon kasvaessa. YOLO-malli sen sijaan hyötyy suuremmasta datasetistä ja saavuttaakin lopulta paremman mAP50-95-arvon. Tämä viittaa siihen, että YOLO voi olla parempi vaihtoehto suuremmalla datamäärällä, kun taas RT-DETR on kilpailukykyinen erityisesti silloin, kun dataa on vähemmän käytettävissä.

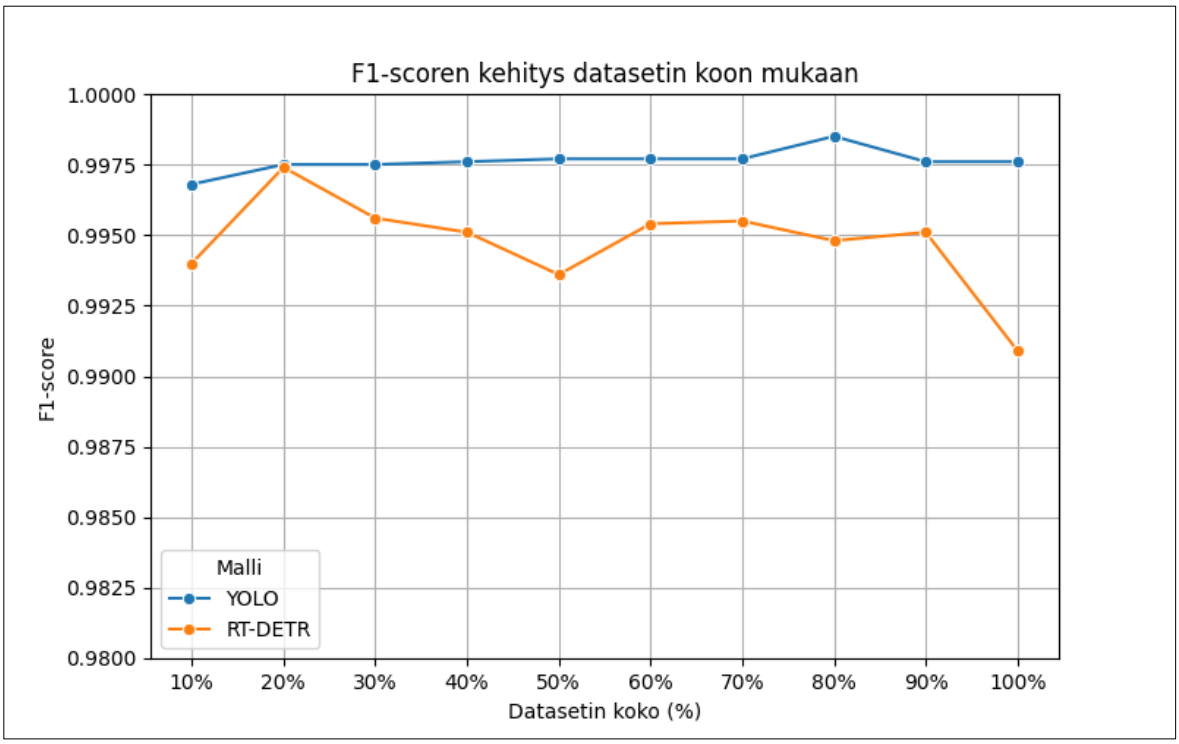
Kuva 11. Mallien havaitsemistarkkuuden vertailu (mAP50-95)



Kuva 12 näyttää koneoppimismallien F1-score-mittarin vertailun eri datamäärillä. F1-score-mittari yhdistää recall-arvon sekä precision-arvon yhdeksi arvoksi. Arvoja voi saada 0 ja 1 väliltä. Mitä suurempi luku, sen parempi malli kyseessä.

Molempien mallin F1-score on erittäin hyvä. YOLO-malli säilyttää kuitenkin korkean F1-scoren jokaisella datasetin eri koolla. RT-DETR-mallin F1-score taas vaihtelee ja laskee datasetin koon kasvaessa.

Kuva 12. Mallien F1-score-mittarin vertailu



8 Johtopäätökset ja pohdinta

Tässä opinnäytetyössä tutkitut ja vertaillut koneoppimismallit YOLO11 ja RT-DETR osoittautuivat molemmat hyviksi vaihtoehdoiksi tuotantolinjojen objektintunnistuksen kontekstissa. Tulosten perusteella voidaan todeta, että mallien suorituskyvyt ovat tämän tutkimuksen otannalla hyvin lähellä toisiaan. RT-DETR suoriutui kuitenkin paremmin pienemmillä koulutusdatamäärillä, kun taas YOLO saavutti paremman suorituskyvyn suuremmilla koulutusdatamäärillä. Nämä tulokset kertovat siitä, että RT-DETR:n transformer-pohjainen arkkitehtuuri oppii tehokkaasti vähäiselläkin määrällä koulutusdataa, kun taas YOLO ja sen konvoluutioneuroverkkoihin perustuva arkkitehtuuri hyötyy suuremmista koulutusdatamääristä.

Opinnäytetyön tavoitteena oli selvittää, kumpi malleista suoriutuu pienemmällä koulutusdatamäärällä ja kumpi malleista suoriutuu objektintunnistuksesta paremmin, ja tuloksien perusteella voidaan todeta, että RT-DETR on pienemmillä koulutusdatamäärillä parempi vaihtoehto.

Jatkokehitystä ajatellen projektia voisi viedä eteenpäin esimerkiksi hyperparametrien optimoinnilla. Tähän voisi käyttää apuna Grid Search tai Bayesian Optimization-menetelmiä. Lisäksi mallien testaaminen aidossa tuotantoympäristössä voisi antaa tarvittavaa lisätietoa jatkokehitystä ajatellen. Objektintunnistukseen soveltua koneoppimismalleja on olemassa todella paljon. Tätä opinnäytetyötä tehdessä vuoden 2025 keväällä myös YOLO-mallista julkaistiin uusi versio YOLO12, joka voisi olla potentiaalinen malli jatkokehitystä ajatellen.

Yksi huomionarvoinen esiin nostettava asia on, että tässä tutkimuksessa ei tutkittu mallien suoriutumista usean eri objektin samanaikaisesta tunnistamisesta. Mallit koulutettiin, validoitiin ja testattiin yksittäisillä kuvilla jokaisesta eri luokasta erikseen. Jatkotutkimuksen aiheena voisi olla vertailututkimus, miten mallit suoriutuvat usean eri objektiluokan samanaikaisesta tunnistamisesta.

9 Yhteenveto

Tämän opinnäytetyön tavoitteena oli vertailla kahta eri koneoppimismallia, YOLO11 ja RT-DETR eri mittareilla. Tavoitteeseen päästiin, eli molemmat mallit saatiin koulutettua ja testattua, ja vertailukelpoisia tuloksia saatiin taulukkomuodossa sekä graafisesti eri kaavioilla toteutettuna ja tutkimuskysymyksiin vastaamisessa onnistuttiin.

Tutkimuksesta saatujen tulosten perusteella RT-DETR osoittautui paremmaksi vaihtoehdoksi tutkittavista koneoppimismalleista. RT-DETR:n suorituskyky oli parempi pienemmillä datamäärillä, kun taas YOLO oli vahvempi suuremmilla datamäärillä. Tämä oli työn kannalta oleellisin kysymys, ja tämä auttaa toimeksiantajaa lopulliseen tuotteeseen päätyvän koneoppimismalliarkkitehtuurin valinnassa.

Opinnäytetyön aikana pääsin syventymään lisää koneoppimisen maailmaan. Pääsin tutkimaan ja oppimaan konenäöstä, ja mitä mahdollisuuksia se tuo mukanaan. Erittäin

antoisaa oli myös saada opinnäytetyöhön toimeksianto. Pääsin tutkimaan ja työskentelemään merkityksellisen työn parissa. Lisäksi tämänkaltainen pidempi projekti opetti minulle projektinhallinnan sekä kommunikoinnin taitoja, jotka vien takuuvarmasti mukanani työelämään.

Lähteet

Alpaydin, E. (2020). *Introduction to Machine Learning, Fourth Edition*. MIT Press. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=6676810

Artasanchez, A., & Joshi, P. (2020). *Artificial intelligence with python: Your complete guide to building intelligent apps using python 3.x, second edition* (2nd ed). Packt Publishing.

Coeckelbergh, M. (2024). *Miksi tekoäly nakertaa demokratiaa ja mitä sille voidaan tehdä* (K. Pietiläinen, Käänt.). Terra Cognita.

Cui, J., & Yang, A. (2025). *Why DeepSeek is different, in three charts*. NBC News. https://www.nbcnews.com/data-graphics/deepseek-ai-comparison-openai-chatgpt-google-gemini-meta-llama-rcna189568

Duffy, C. (2025). *Trump announces a $500 billion AI infrastructure investment in the US | CNN Business*. CNN. https://www.cnn.com/2025/01/21/tech/openai-oracle-softbank-trump-ai-investment/index.html

Géron, A. (2023). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition*. O'Reilly Media, Inc. https://learning.oreilly.com/library/view/hands-on-machine-learning/9781098125967/

IBM. (2024). *What Is Artificial Intelligence (AI)? | IBM*. https://www.ibm.com/think/topics/artificial-intelligence

Järvinen, P. (2023). *Tekoäly ja minä: Ihmisenä tekoälyn aikakaudella*. Tammi.

Kejriwal, K. (2023). *YOLOv7: Edistynein objektintunnistusalgoritmi?* Unite.AI. https://www.unite.ai/fi/yolov7/

Khanam, R., & Hussain, M. (2024). *YOLOv11: An Overview of the Key Architectural Enhancements* (arXiv:2410.17725). arXiv. https://doi.org/10.48550/arXiv.2410.17725

Klusaitė, L. (2023). *Mitä on koneoppiminen? | NordVPN*. https://nordvpn.com/fi/blog/koneoppiminen/

Lehto, M., Neittaanmäki, P., Niinimäki, E., Nyrhinen, R., Ojalainen, A., Pölönen, I., Rautiainen, I., Ruohonen, T., Tuominen, H., Vähäkainu, P., Äyrämö, S., & Äyrämö, S.-M. (2019).

tekoälyn_perusteita_ja_sovelluksia_kirja—TIM. https://tim.jyu.fi/view/kurssit/tie/tiep1000/tekoalyn-sovellukset/kirja

Microsoft. (2024). *Microsoft Copilot*. Microsoft Copilot: tekoälykumppanisi. https://copilot.microsoft.com

Mukherjee, S. (2024). *What's New in YOLOv11 Transforming Object Detection Once Again Part 1 | DigitalOcean*. https://www.digitalocean.com/community/tutorials/what-is-new-with-yolo

OpenAI. (2022). *ChatGPT*. https://chatgpt.com

Provendor Oy. (n.d.). *Siemens – Konenäkö – Provendor Oy*. https://provendor.fi/siemens-konenako/

SAP. (n.d.). *Machine Learning | Definition, types, and examples*. SAP. https://www.sap.com/products/artificial-intelligence/what-is-machine-learning.html

Suomen Koodikoulu. (2019). *Johdatus tekoälyyn*. https://aoe.fi/api/v1/download/file/Johdatustekolyyn-1576067361216.pdf

Tekoälyaika. (2023). Mikä on tekoäly—Tekoalyaika.fi. *Mikä on tekoäly*. https://tekoalyaika.fi/mista-on-kyse/

Theobald, O. (2017). *Machine learning for absolute beginners: A plain English introduction* (Second edition). Scatterplot Press.

Ultralytics. (n.d.a). *RT-DETR (Realtime Detection Transformer)*. https://docs.ultralytics.com/models/rtdetr

Ultralytics. (n.d.b). *YOLO11*. https://docs.ultralytics.com/models/yolo11

Veerman, N. (2024). *PythonYoloV5—V7 Foto download*. Roboflow. https://universe.roboflow.com/niels-veerman/pythonyolov5/dataset/7

Viitaila, M. (2018). Tekoälyn perusteet: Koneoppiminen, työn tulevaisuus ja hyvä vai paha tekoäly. *Microsoft Pulse*. https://pulse.microsoft.com/fi-fi/transform-fi-fi/na/fa2-tekoalyn-perusteet-koneoppiminen-tyon-tulevaisuus-ja-hyva-vai-paha-tekoaly/

VTT. (n.d.). *Konenäkö teollisuudessa*. https://www.vttresearch.com/fi/palvelut/konenako

Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., & Chen, J. (2024). *DETRs Beat YOLOs on Real-time Object Detection* (arXiv:2304.08069). arXiv. https://doi.org/10.48550/arXiv.2304.08069

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön aineiston kuvaus

Opinnäytetyö on tutkimuksellinen kehitysprojekti, jonka aikana kerätään erilaista teknistä tietoa projektin edistymisestä ja lopputuloksesta. Teknisellä tiedolla tarkoitetaan kuvankaappauksia koneoppismallien kehitysprosesseista, suorituskykyjen arvioinneista sekä esimerkkejä käsiteltävästä datasta. Kehitysprojektin aikana pidetään päiväkirjaa, johon kirjataan yksityiskohtaisesti projektin eteneminen, tehtävät päätökset, ratkaisut, ongelmat ja niiden ratkaisut sekä muut merkittävät tapahtumat. Lisäksi kerätään tietoa projektin onnistumisesta, esimerkiksi testiraportteja ja muita relevantteja dokumentteja. Tämä aineistonhallintasuunnitelma varmistaa, että kaikki aineiston kerääminen, käsittely, säilyttäminen ja mahdollinen tuhoaminen tapahtuvat asianmukaisesti.

Aineiston tallennus ja säilytys

Aineisto säilytetään, ja sitä käsitellään opinnäytetyön tekemisen aikana kahdessa eri paikassa. Ensimmäinen paikka on opinnäytetyön tekijän salasanalla suojatun tietokoneen C-asemalla, kansiossa nimeltä "Opinnäytetyö". Tästä kansiosta tehdään säännöllisesti varmuuskopiot ulkoiselle muistitikulle sekä Microsoft OneDriveen varmistaen, että aineisto säilyy turvallisesti ja se voidaan tarvittaessa palauttaa. Toinen paikka on opinnäytetyön toimeksiantajan Konekatse Oy:n tarjoama pilvipalvelin. Pilvessä pääasiassa suoritetaan laskentaa, ja säilytetään työssä käytettävää dataa. Aineistoa säilytetään C-asemalla sekä ulkoisella muistitikulla vähintään yhden vuoden ajan opinnäytetyön valmistumisesta. Tämä varmistaa, että opinnäytetyön tulokset ovat käytettävissä ja niitä voidaan tarvittaessa tarkastella. Opinnäytetyön tekijän lisäksi aineistoa käsittelee mahdollisesti myös opinnäytetyön ohjaaja sekä toimeksiantajayrityksen yhteyshenkilö.

Henkilötietojen ja arkaluonteisten tietojen käsittely

Opinnäytetyössä ei käsitellä arkaluonteisia tietoja, kuten henkilötietoja.

Aineiston omistajuus

Opinnäytetyön aineiston sekä tulokset omistaa Konekatse Oy.

Opinnäytetyöaineiston jatkokäyttö työn valmistumisen jälkeen

Opinnäytetyön valmistuttua aineisto siirtyy Konekatse Oy:n omistukseen mahdollista jatkotutkimusta ja kaupallisen tuotteen kehittämistä varten.

Liite 2. Datan jakaminen koulutus-, validointi ja testausjoukkioihin Python-ohjelmointikielellä

```
import os
import random
import shutil

# määritä alkuperäisen datan ja kohdekansioiden polut
all_images_dir = "datasets/all/images"
all_labels_dir = "datasets/all/labels"

train_images_dir = "datasets/train/images"
train_labels_dir = "datasets/train/labels"

valid_images_dir = "datasets/valid/images"
valid_labels_dir = "datasets/valid/labels"

test_images_dir = "datasets/test/images"
test_labels_dir = "datasets/test/labels"

# luo kohdekansiot, jos niitä ei vielä ole
for dir_path in [train_images_dir, train_labels_dir, valid_images_dir,
valid_labels_dir, test_images_dir, test_labels_dir]:
    os.makedirs(dir_path, exist_ok=True)

# listaa kaikki kuvatiedostot alkuperäisessä hakemistossa
image_files = [f for f in os.listdir(all_images_dir) if
f.endswith(('.jpg', '.png'))]

# sekoita tiedostot satunnaisesti
random.shuffle(image_files)

# laske jakojen määrät
num_images = len(image_files)
num_train = int(num_images * 0.7)
num_valid = int(num_images * 0.15)

# jaa tiedostot joukkoihin
train_files = image_files[:num_train]
valid_files = image_files[num_train:num_train + num_valid]
test_files = image_files[num_train + num_valid:]

# funktio tiedostojen siirtämiseen
def move_files(file_list, src_images_dir, src_labels_dir,
dest_images_dir, dest_labels_dir):
    for file_name in file_list:
        # siirrä kuvatiedosto
        shutil.move(os.path.join(src_images_dir, file_name),
os.path.join(dest_images_dir, file_name))

        # siirrä vastaava label-tiedosto
        label_file = file_name.replace('.jpg', '.txt').replace('.png',
'.txt')
        shutil.move(os.path.join(src_labels_dir, label_file),
os.path.join(dest_labels_dir, label_file))

# siirrä tiedostot omiin kansioihinsa
move_files(train_files, all_images_dir, all_labels_dir,
train_images_dir, train_labels_dir)
```

```
move_files(valid_files, all_images_dir, all_labels_dir,
valid_images_dir, valid_labels_dir)
move_files(test_files, all_images_dir, all_labels_dir,
test_images_dir, test_labels_dir)

# tulosta tiedostojen määrät
print(f"Train: {len(train_files)} images")
print(f"Validation: {len(valid_files)} images")
print(f"Test: {len(test_files)} images")
```