Datasetin optimointi ja mallin luominen objektin tunnistusta varten

Calevala Interactive Oy

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus
syksy, 2023

Teija Ouramaa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Tekijä Teija Ouramaa Vuosi 2023

Työn nimi Datasetin optimointi ja mallin luominen objektin tunnistusta varten

Ohjaaja Lasse Seppänen

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli selvittää, millainen on optimoitu datasetti objektin tunnistusohjelmaa varten. Lisäksi selvitettiin, kuinka Yolov5-tietokonenäkömalli luodaan ja tutkittiin erilaisia menetelmiä sen luomiseen. Opinnäytetyön toimeksiantajana toimi Calevala Interactive Oy.

Opinnäytetyön teoriaosuudessa tutustutaan tekoälyn ja koneopin käsitteisiin sekä käydään läpi objektintunnistusprosessin eri vaiheita. Teoriaosuudessa käsitellään myös annotointiprosessia ja erilaisia saatavilla olevia annotointityökaluja. Lisäksi työssä tutustutaan YOLO-tietokonenäkömalliin, mallin kouluttamisen menetelmiin sekä valmiin mallin arviointiin. Opinnäytetyö on toiminnallinen. Käytännön osuudessa luotiin tarkoituksella erilaatuisia datasettejä, joilla koulutettiin eri menetelmiä käyttäen tietokonenäkömalleja.

Käytännön osuuden avulla havaittiin, että korkealaatuisella ja laajalla datasetillä koulutettu tietokonenäkömalli on usein toimivampi sekä tarkempi kuin heikolla datalla koulutettu malli. Näiden tulosten avulla toimeksiantaja pystyy luomaan ohjeet datasetin valmisteluun, optimointiin sekä mallin luomiseen.

Avainsanat annotointi, objektin tunnistus, tietokonenäkö, Yolov5

Sivut 68 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology Abstract

Author Teija Ouramaa Year 2023

Subject Dataset optimization and model creation for object detection

Supervisor Lasse Seppänen

ABSTRACT

The purpose of this thesis was to research what is needed to create an optimized dataset for an object recognition program. In addition, a goal was to create a Yolov5 computer vision model and research different methods to create it. The thesis was commissioned by Calevala Interactive Oy.

The theoretical part of this thesis explores the concepts of artificial intelligence and machine learning as well as goes through the various stages of the object recognition process. The theory section also discusses the annotation process and different annotation tools. In addition, this thesis explores the YOLO computer vision model, the methods of training the model, and the evaluation of the finished model. The thesis is functional. In the practical part, datasets of different quality were deliberately created to train computer vision models. Training was executed using different methods.

The conclusion of the practical part was that a computer vision model trained with a high-quality and extensive dataset is often more functional and accurate than a model trained with weak data. Finally, these results can be used to create instructions for the client company to prepare and optimize an object detection dataset and eventually create a functional model.

Keywords annotation, object detection, computer vision, Yolov5

Pages 68 pages and appendices 1 page

Sanasto

TEKOÄLY	AI, Artificial Intelligence, joustavasti ja käytännöllisesti toimiva järjestelmä, joka oppii monimutkaisessa ja ennalta määrittelemättömässä ympäristössä
DATASETTI	tietoaineisto
KONEOPPIMINEN	oppimisalgoritmeihin perustuva tekoälyn osa-alue
ALGORITMI	prosessikuvaus
TUNNISTE	label, datan luokkamerkintä
LUOKITTELU	objektin luokan ennustaminen ja tunnisteen lisääminen
NEUROVERKKO	koneoppimisen menetelmä, jossa järjestelmä oppii esimerkkejä analysoivan tekniikan avulla
KONVOLUUTIONEUROVERKKO	matemaattista konvoluutio-operaatiota soveltava neuroverkko
KONENÄKÖ	keinotekoisesti ihmisen näkökykyä matkiva menetelmä
TIETOKONENÄKÖ	digitaalisten kuvien tai videoiden sisältöä ymmärtävä menetelmä
PAIKANTAMINEN	objektin sijainnin määrittely kuvassa tai videossa
RAJAUSLAATIKKO	bounding box, paikantamisen visuaalinen määrittely
AUGMENTOINTI	lisäys
ANNOTAATIO	kuvan tunnisteen määrittelyprosessi objektin tunnistuksessa
ANNOTAATTORI	annotoinnin suorittaja
PYTORCH	Pythonilla kirjoitettu avoimen lähdekoodin kirjasto koneoppimiseen
GPU	graphics processing unit, grafiikkaprosessori

Sisällys

1 Johdanto 1
2 Tekoäly 2
2.1 Koneoppiminen 2
2.1.1 Koneoppimisen osa-alueet 3
2.1.2 Neuroverkot 4
2.1.3 Konvoluutioneuroverkot 5
2.1.4 Syväoppiminen 5
2.2 Tietokonenäkö 6
3 Objektintunnistus 7
3.1 Datasetit ja referenssikuvat 7
3.1.1 Aineiston haasteet ja vaatimukset 8
3.1.2 Aineiston siivoaminen 8
3.1.3 Augmentointi 9
3.2 Opetus- ja validointidata 10
4 Annotointi 12
4.1 Luokittelu, paikantaminen ja tunnistaminen 12
4.2 Yleiset ongelmat annotaatiossa 12
4.3 Annotointityökalut 13
4.3.1 Make Sense 13
4.3.2 Label Studio 14
4.3.3 LabelIMG 14
4.3.4 LabelFlow 14
4.3.5 Roboflow 15
5 YOLO 16
5.1 YOLO:n ominaisuudet 16
5.2 YOLO:n historia ja kehitys 16
5.3 YOLOv5 17
6 Mallin kouluttaminen ja arviointi 19
6.1 Ultralytics HUB 19
6.2 Google Colab 20
6.3 Paikallinen tietokone 21
6.4 NVIDIA Cuda Toolkit ja PyTorch 23
6.5 Mallin arviointi 25

7 Kehittämistyön tavoite ja tarkoitus 27
8 Optimoidun datasetin luominen objektintunnistukseen 29
8.1 Referenssikuvat 29
8.2 Augmentointi 30
8.3 Annotointi 31
8.4 Mallin koulutus ja tulokset 37
8.4.1 Ultralytics Hub ja Google Colab 37
8.4.2 Paikallinen tietokone ja CPU 48
8.4.3 Paikallinen tietokone ja GPU 51
9 Tulokset 60
10 Johtopäätökset ja pohdinta 62
11 Yhteenveto 64
Lähteet 65

Kuvat, ohjelmakoodit ja taulukot

Komento 1 YOLOv5-kirjaston kloonaus ja requirements-tiedoston asennus (Ultralytics, ei pvm.-b) 21
Komento 2 Mallin koulutusesimerkki (Ultralytics, ei pvm.-b) 22
Komento 3 PyTorch-asennuskomento (Davies, 2022) 25
Komento 4 Label Studion asennus ja käynnistys 31
Komento 5 YOLOv5-kuvauskannan kloonaus paikalliselle koneelle 48
Komento 6 Requirements-tiedoston asennus paikalliselle koneelle 48
Komento 7 Koulutuksen aloittaminen paikallisella koneella 50
Komento 8 Mallin testauksen suorittaminen paikallisella koneella 50
Komento 9 PyTorchin lataus ja asennus 52
Komento 10 Mallin koulutuskomento Cudan avulla 53

Kuva 1 Neuroverkko (Jyväskylän Yliopisto, ei pvm., s. 8) 5
Kuva 2 Rajauslaatikot (Brownlee, 2019) 7
Kuva 3 YOLO:n kehitys 2015-2022 (Kelta, 2022) 17
Kuva 4 Datasetin rakenne (Krittapholchai, 2022) 19
Kuva 5 .yaml-tiedoston sisältö (Krittapholchai, 2022) 20

Kuva 6 YOLOv5-projektikansion rakenne (Ultralytics, ei pvm.-b) 22
Kuva 7 NVIDIA Cuda Toolkit -lataus (Davies, 2022).....24
Kuva 8 NVIDIA Cuda -asennus (Davies, 2022) 25
Kuva 9 Esimerkki YOLOv5-koulutuksen yhteenvedosta (Seeed Studio, 2023) 26
Kuva 10 Kollaasi referenssikuvista.....30
Kuva 11 Referenssikuvan augmentointi 31
Kuva 12 Label Studion valmiit annotointipohjat 32
Kuva 13 Label Studion asetukset.....33
Kuva 14 Label Studion annotointiaineiston valinta.....34
Kuva 15 Label Studion rajauslaatikoiden lisäys 34
Kuva 16 Virheellisesti annotoidut objektit 35
Kuva 17 Label Studion siirrettävän datan formaatti 36
Kuva 18 Datasetin kansiorakenne ja sisältö 37
Kuva 19 Datasetin lataus 38
Kuva 20 Oma datasetti Ultralytics Hub:ssa 39
Kuva 21 Mallin koulutus, vaihe 1.....39
Kuva 22 Mallin koulutus, vaihe 2.....40
Kuva 23 Mallin koulutus, vaihe 3.....41
Kuva 24 Google Colab -asennukset 41
Kuva 25 API-avaimen lisäys ja mallin URL 42
Kuva 26 Google Colab -tulokset 30 referenssikuvalla 42
Kuva 27 Google Colab -tulokset 30 referenssikuvalla 43
Kuva 28 Google Colab -tulokset 300 referenssikuvalla 44
Kuva 29 300:n referenssikuvan mallin testaus Ultralytics Hub:ssa.....45
Kuva 30 Heikon, 30:n kuvan datasetin koulutuksen yhteenveto.....45
Kuva 31 Heikon, 30:n kuvan datasetillä luodun mallin testaus.....46
Kuva 32 Heikon, 300:n kuvan datasetin koulutuksen yhteenveto.....47
Kuva 33 Heikon, 300:n kuvan datasetillä luodun mallin testaus 47
Kuva 34 Datasets-kansion rakenne ja sisältö paikallisella koneella 49
Kuva 35 30:n kuvan koulutuksen yhteenveto paikallisella koneella.....50
Kuva 36 Mallin testauksen tulokset paikallisella koneella 51
Kuva 37 NVIDIA Cuda Toolkit lataus paikalliselle koneelle 52

Kuva 38 Cuda-asennusten tarkastus 53
Kuva 39 30:n kuvan koulutus Cudan avulla 53
Kuva 40 Cuda-mallin testauksen tulokset paikallisella koneella 54
Kuva 41 300:n kuva koulutus Cudan avulla 55
Kuva 42 300:n kuvan datasetillä luodun mallin testitulokset 55
Kuva 43 Koulutuksen yhteenveto heikolla 30:n kuvan datasetillä 56
Kuva 44 Heikon, 30:n kuvan datasetin mallin testaus paikallisella koneella 57
Kuva 45 Heikon, 300:n kuvan datasetillä luodun mallin koulutuksen yhteenveto 58
Kuva 46 Heikon, 300:n kuvan datasetin mallin testaus paikallisella koneella 59

Ohjelmakoodi 1 Yaml-tiedoston sisältö (Ultralytics, ei pvm.-b) 21
Ohjelmakoodi 2 Coco128.yaml-tiedoston sisältö paikallisella koneella 49

Taulukko 1 Kuvan augmentointimenetelmiä (Dilmegani, 2023) 9
Taulukko 2 Luotujen mallien yhteenveto 61

Liitteet

Liite 1 Aineistonhallintasuunnitelma

1 Johdanto

Tässä opinnäytetyössä tarkastellaan mukautetun datasetin ominaisuuksia kuvan tunnistuspalvelua varten, kun käytetään YOLOv5 tietokonenäkömallinnusta. Tarkoituksena on selvittää, minkälainen on hyvä datasetti ja miten se optimoidaan mallinnusta varten. Työssä käydään läpi manuaalinen referenssikuvien luokittelu ja siihen käytetyt menetelmät ja työkalut sekä datan laadun tarkastelu ja mahdolliset siihen liittyvät sudenkuopat. Lisäksi etsitään keinoja datan hallitsemiseen ja optimoinnin tehostamiseen. Työssä yritetään luoda mahdollisimman optimoitu opetusdataesimerkki, jolla luodaan malli objektin tunnistamista varten käymällä läpi datasetin sekä referenssikuvien valmistelu mallin luomiseen asti.

Tämän työn tavoitteena on luoda yritykselle suositukset kuvan tunnistamisdatan käsittelyä varten. Työn lopputulosten avulla pystytään luomaan ohjeistus kuvantunnistusohjelmaan tarvittavien referenssikuvien ja datasetin hallintaan sekä käsittelyyn.

Toimeksiantajayritys Calevala Interactive Oy käyttää työn tuloksia hyödyksi rakentaessaan uutta palvelua, jossa objektien tunnistus kuvissa on keskeisessä osassa.

Tutkimuskysymykset:

- Millainen on hyvä referenssikuva?
- Millainen on optimoitu datasetti objektin tunnistusta varten?
- Mitkä ovat mahdolliset ongelmat referenssikuvien ja datasetin hallinnassa?
- Mitä eri menetelmiä on luoda Yolov5-malli ja miten ne eroavat toisistaan?

2 Tekoäly

Tekoäly (AI, Artificial Intelligence) on konseptina laaja ja hyvin vaikeasti määriteltävissä. Yksi perustava ongelma määrittelyn taustalla on se, ettei edes luonnollista älykkyyttä voida määritellä yksiselitteisesti. (Ailisto ym., ei pvm., s. 37) Tekoäly voidaan kuitenkin nähdä yhtenä suurena kokonaisuutena erilaisia teknologioita ja sovelluksia. Tällaisia ovat esimerkiksi data-analyysi, koneoppiminen sekä robotiikka ja koneautomaatio. Näitä teknologioita käyttävien järjestelmien älykkyys perustuu niiden käyttämään dataan, algoritmeihin sekä arkkitehtuuriin. Yleisesti tekoäly määritellään järjestelmäksi, joka kykenee toimimaan joustavasti, käytännöllisesti sekä oppimaan monimutkaisessa ja ennalta määrittelemättömässä ympäristössä. (Valtionvarainministeriö, ei pvm., s. 29)

Lisäksi tekoäly voidaan jakaa kahteen luokkaan, heikkoon ja vahvaan tekoälyyn. Heikko tekoäly on algoritmeihin perustuvaa toimintaa, jonka avulla pystytään suorittamaan yksittäisiä tehtäviä. Kaikki nykyiset tekoälyjärjestelmät ovat heikon tekoälyn kategoriaan kuuluvia järjestelmiä. Tällaisia järjestelmiä ovat mm. shakkiohjelma ja hahmontunnistusjärjestelmä. Vahva tekoäly sen sijaan tarkoittaa tekoälyä, joka kykenee älyllisesti ratkaisemaan ongelmia. Se on tietoinen sekä aidosti älyllinen. Tällaista tekoälyä kuvaillaan lähinnä tieteiselokuvissa ja -kirjallisuudessa, eikä todellisuudessa sen luomisessa ole tullut kehitystä moneen vuosikymmeneen. (MinnaLearn & Helsingin Yliopisto, ei pvm.-d)

2.1 Koneoppiminen

Koneoppiminen (Machine Learning) on yksi merkittävä tekoälyn osa-alue, joka liittyy läheisesti tilastotieteisiin. Se perustuu algoritmeihin, jotka oppivat sille syötetyn tiedon avulla. Lopputuloksena syntyy koneoppimismalli. (Model) (MinnaLearn, ei pvm.-a)

Suuri osa nykypäivän tekoälyjärjestelmistä perustuu koneoppimiseen. Tämän osa-alueen järjestelmä pystyy havainnoimaan ympäristöään ja muokkaamaan toimintaansa näiden havaintojen mukaan. Perinteisestä ohjelmoinnista eroten koneoppimisessa järjestelmä ei tarvitse ohjelmoijaa ratkaisemaan jokaista yksityiskohtaa. Koneoppimisjärjestelmä kykenee säätämään ja muuttamaan itseään sille syötetyn datan avulla. (Ailisto ym., ei pvm., s. 37) Tämän syötetyn datan

lisäksi järjestelmän käyttäjän toiminnan tarkoitus on parantaa tekoälyjärjestelmän toimintoja. Tavoitteena on automatisoida datan tulkitsemista ja laajentaa järjestelmän havainnointia algoritmien avulla. (Jyväskylän Yliopisto, ei pvm., s. 7)

2.1.1 Koneoppimisen osa-alueet

Koneoppiminen voidaan jakaa oppimisen perusteella yleisesti kolmeen eri kategoriaan. Nämä kategoriat ovat ohjattu oppiminen, ohjaamaton oppiminen ja vahvistettu oppiminen (Jyväskylän Yliopisto, ei pvm., s. 7).

Ohjatussa oppimisessa koneelle annetaan syöte (Input), jonka avulla järjestelmän tehtävä on ennustaa oikea vaste eli tunniste (Label). Järjestelmään syötetään esimerkiksi kuva eläimistä ja järjestelmä luokittelee kuvasta oikeat, ennalta määritellyt luokat (kissa, koira yms.). (MinnaLearn & Helsingin Yliopisto, ei pvm.-b) Riippuen käytetyn aineiston tyypistä ja tavoitteista, ohjattu oppiminen voidaan edelleen jakaa luokitteluun ja regressioon. Jos data voidaan jakaa eri ryhmiin, puhutaan luokittelusta. Luokittelua on esimerkiksi sähköpostin jakaminen roskapostiin ja tärkeisiin viesteihin. Regressiolla tarkoitetaan dataa, joka on jatkuvaa. Tällaista dataa on esimerkiksi lämpötilan vaihtelun määrittely. (Jyväskylän Yliopisto, ei pvm., s. 15)

Ohjaamattomassa oppimisessa ei käytetä luokkia, vaan järjestelmän tarkoituksena on löytää datasta yhtenäisiä rakenteita, samankaltaisuuksia ja riippuvuuksia (MinnaLearn & Helsingin Yliopisto, ei pvm.-b). Koneelle annetut syötteet yritetään ryhmitellä samankaltaisten ominaisuuksien mukaan ja samalla erotella ne muunlaisista syötteistä. Tämä oppimisen muoto muistuttaa ihmisen oppimista. (Jyväskylän Yliopisto, ei pvm., s. 16)

Vahvistetussa oppimisessa kone, ns. älykäs agentti, oppii ympäristön palautteen kautta. Se saa ympäristöltään positiivista ja negatiivista palautetta toiminnastaan. Tämän palautteen avulla kone oppii toimimaan niin, että positiivinen palaute lisääntyy ja negatiivinen vähenee. Tämän tyyppistä oppimista käytetään esimerkiksi itseohjautuvissa autoissa. (Jyväskylän Yliopisto, ei pvm., s. 16)

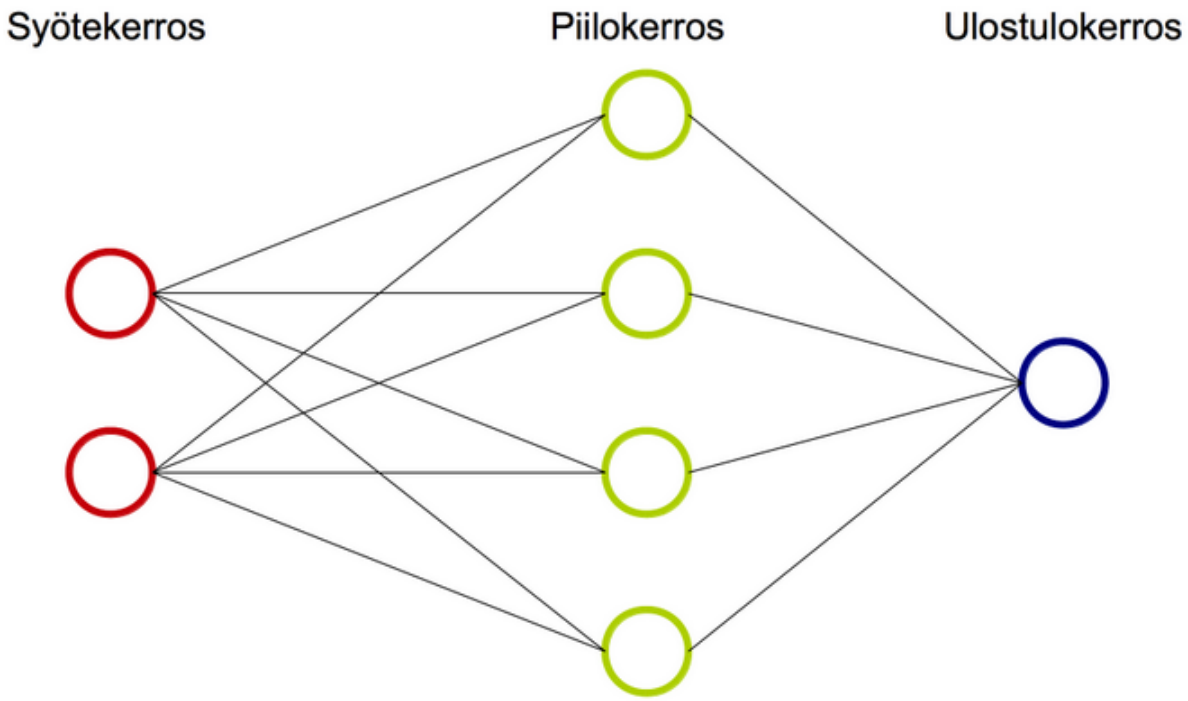
2.1.2 Neuroverkot

Neuroverkot ovat koneoppimisen menetelmä, jossa järjestelmä oppii suorittamaan tietyn tehtävän esimerkkejä analysoivan tekniikan avulla. Esimerkiksi kuvantunnistusjärjestelmässä voidaan syötteenä käyttää tuhansia ennalta luokiteltuja kuvia eri objekteista. Järjestelmä analysoi nämä syötetyt kuvat ja etsii niistä piirteitä sekä kuvioita, jotka korreloivat luokiteltujen objektien kanssa. (Hardesty, 2017)

Keinotekoiset neuroverkot muistuttavat jossain määrin ihmisaivoja, sillä ne koostuvat suuresta joukosta neuroneja, jotka ovat yhteydessä toisiinsa johtojen avulla. Data kulkee neuroneista eteenpäin, eli jokainen neuroni saa edeltävästä kerroksesta syötteenä tietoa ja lähettää tietoa johtojen avulla seuraavaan kerrokseen. (MinnaLearn, ei pvm.-b) Keinotekoinen neuroverkko koostuu syöte-, piilo- ja ulostulokerroksista sekä näiden kerrosten sisällä olevista neuroneista. Piilo- ja ulostulokerrosten neuroneilla on mukautuvia parametreja eli painoja (Weight) ja kynnysarvoja (Bias). Nämä parametrit muuttuvat neuroverkon opetuksen aikana. (Jyväskylän Yliopisto, ei pvm., ss. 26, 29)

Kun oppiminen aloitetaan, kaikki painot ja kynnysarvot asetetaan satunnaisiin arvoihin. Opetusdata syötetään syötekerrokseen, joista se kulkee useiden johtojen kautta jatkuvasti muuttuen piilokerroksiin. Oppimisen aikana painot ja kynnysarvot muuttuvat jatkuvasti, kunnes samalla tavalla merkityt ja luokitellut tiedot alkavat tuottaa samankaltaisia tuloksia. Lopulta valmis tuotos saapuu ulostulokerrokseen (Kuva 1). (Hardesty, 2017)

Kuva 1 Neuroverkko (Jyväskylän Yliopisto, ei pvm., s. 8)



2.1.3 Konvoluutioneuroverkot

Konvoluutioneuroverkot (Convolutional Neural Networks, CNN) ovat neuroverkkoja, jotka soveltavat matemaattista konvoluutio-operaatiota ainakin yhdessä sen kerroksista (Goodfellow ym., 2016, s. 326). Tämäntyyppisten neuroverkkojen käyttö on yleistä erityisesti silloin, kun syötetty aineisto on visuaalista. Perinteisiä neuroverkkoja käyttäen datamäärät saattavat kasvaa kuvantunnistuksessa valtaviksi aiheuttaen ongelmia. Konvoluutiokerroksen lisääminen neuroverkkoon ratkaisee tämän ongelman. Konvoluutioneuroverkot auttavat tunnistamaan kuvissa erityisiä kuvapiirteitä (Image Feature) mistä kohtaa kuvaa tahansa. Näiden piirteiden avulla voidaan kuvista tunnistaa yhä monimutkaisempia objekteja. (MinnaLearn & Helsingin Yliopisto, ei pvm.-a)

2.1.4 Syväoppiminen

Syväoppimisella (Deep Learning) tarkoitetaan sellaista koneoppimisen tekniikkaa, jossa käytetään lukuisia piilokerroksia sisältävää neuroverkkoa. Neuroverkon syvyys kasvaa sitä mukaa kun kerrosten lukumäärä kasvaa. Verkon kasvaessa myös oppiminen vaikeutuu, sillä syvä neuroverkko

vaatii suuria datamääriä ja enemmän tehoja. Toisaalta neuroverkon syventyminen mahdollistaa monimutkaisempien asioiden oppimisen. (MinnaLearn & Helsingin Yliopisto, ei pvm.-c)

2.2 Tietokonenäkö

Tietokonenäkö (Computer Vision) on yksi tekoälyn ja koneoppimisen osa-alueista, joka koostuu visuaalisen datan hankkimisesta, käsittelystä, analysoinnista sekä ymmärtämisestä. Sen tarkoitus on kouluttaa tietokoneita syväoppimisen kautta käsittelemään visuaalista dataa, kuten kuvia ja videoita. Tämän lisäksi tarkoituksena on saada kone ymmärtämään visuaalista aineistoa samalla tavalla kuin ihmiset kykenevät. (Simplilearn, 2021)

Termit konenäkö (Machine Vision) ja tietokonenäkö saattavat välillä mennä sekaisin. Konenäöllä viitataan yleisesti menetelmään, jolla yritetään matkia ihmisen näkökykyä. Se koostuu erilaisista välineistä sekä teknisistä laitteista ja ohjelmista, jotka auttavat kuvan tulkinnassa. Tietokonenäkö taas on digitaalisten kuvien tai videoiden sisällön ymmärtämistä. (Jyväskylän Yliopisto, ei pvm.)

3 Objektintunnistus

Objektintunnistus (Object Detection) on osa tietokonenäön osa-aluetta, jonka avulla tunnistetaan ja paikannetaan objekteja visuaalisesta aineistosta (Xiao ym., 2020). Objektintunnistaminen koostuu objektin luokittelusta (Classification) ja paikantamisesta (Localization). Luokittelulla tarkoitetaan aineistosta löytyvän objektin luokan ennustamista ennalta muodostettujen määritelmien mukaan ja selitettävän tunnisteen lisäämistä kuvan kohteeseen. Objektin paikantamisella tarkoitetaan tunnistettavan kohteen sijainnin määrittelyä kuvassa tai videossa. Tähän tarkoitukseen käytetään usein rajauslaatikoita (Bounding Box) (Kuva 2). (Brownlee, 2019)

Kuva 2 Rajauslaatikot (Brownlee, 2019)

Objektintunnistusta voidaan lähestyä monen erilaisen algoritmin kautta. Tällaisia tapoja ovat mm. Fast R-CNN, Retina-Net ja Multibox Detector (SSD). Yksi suosituimpia objektintunnistuksessa käytettäviä algoritmeja on kuitenkin YOLO, joka suoriutuu mallin luomisesta edellä mainittuja lähestymistapoja paremmin. (Karimi, 2021) YOLO-algoritmiin tutustutaan tarkemmin luvussa 5.

3.1 Datasetit ja referenssikuvat

Objektintunnistusta varten tarvitaan laaja aineisto eli datasetti, joka koostuu referenssikuvista. Näiden järjestelmään syötettyjen kuvien avulla objektintunnistusohjelma tai -järjestelmä voidaan opettaa tunnistamaan eri kohteita visuaalisesta aineistosta. Tällaisia valmiita aineistoja löytyy lukuisista eri lähteistä, mutta toisinaan oikeantyyppistä, valmista aineistoa ei ole saatavilla. Tällöin

kuvien kerääminen sekä valmistelu on suoritettava alusta alkaen, mikä saattaa olla hyvin aikaa vievää ja haasteellista. (Kaur & Singh, 2022, s. 6)

3.1.1 Aineiston haasteet ja vaatimukset

Objektin tunnistamiseen aineistosta liittyy useita haasteita. Tarkan objektin paikantamisen ja tunnistamisen lisäksi on selkeästi erotettava eri luokkien kohteet toisistaan (High distinctiveness). Lisäksi on osattava tunnistaa kohteet, jotka ulkonäköeroista huolimatta kuuluvat samaan luokkaan (High robustness). Saman luokan sisäisillä kohteilla voi olla eroavaisuuksia lisäksi väreissä, rakenteissa, koossa tai muodossa. Kuvan luomisen olosuhteet, kuten valaistus tai ympäristö, voivat myös vaikuttaa voimakkaasti kohteen ulkonäköön. Lisähaasteita objektintunnistamiseen luovat esimerkiksi erilaiset suodattimista johtuvat vääristymät kuvissa, kuvien heikko resoluutio tai muunlaiset vääristymät. (Liu ym., 2019)

Datasetin vaatimukset määräytyvät aina tapauskohtaisesti. Koneopissa on kuitenkin yleisesti määritelty suosituksia, jotka selventävät, millaisia piirteitä laadukkaalla aineistolla tulisi olla. Suositusten mukainen, korkeatasoinen datasetti on kohdetta monipuolisesti edustava, jäljitettävissä, tarkka, luotettava, johdonmukainen, eheä sekä puolueeton. (Cappi ym., 2021, ss. 6–12) Koneoppimisen lopputulos määräytyy pitkälti sen mukaan, kuinka paljon aineistoa on käytettävissä, kuinka laadukasta se on ja millainen aineiston rakenne on. Huono opetusdata luo huonoja lopputuloksia, ja päinvastoin. Suuri aineisto ei myöskään aina takaa laadukasta lopputulosta. Tärkeämpää on, että aineisto sisältää kohdetta edustavaa ja oikeanlaista dataa. (MinnaLearn, ei pvm.-a)

3.1.2 Aineiston siivoaminen

Datasetin kokoamisen jälkeen koko aineisto käydään uudestaan läpi ja siitä poistetaan ja korjataan kaikki data, joka on virheellistä tai epäolennaista. Näiden lisäksi kaikki kaksoiskappaleet poistetaan. (Mage, 2021) Objektintunnistamista varten datasetin kuvien kokoa pitää mahdollisesti säätää, rajata tai mitoittaa uudelleen. Aineisto saattaa myös tarvita purkamista tai kuvien väriasteikko muuntamista. (Kanber, 2018, s. 45)

3.1.3 Augmentointi

Yksi keino laajentaa objektintunnistusaineistoa on kuvien augmentointi eli lisääminen. Se on prosessi, jossa käytettyä aineistoa muokataan säilyttäen sen alkuperäinen konteksti. Kuvien kohdalla sillä voidaan tarkoittaa esimerkiksi kuvan kääntämistä, kiertämistä, värilämpötilan muokkaamista tai kohinan lisäämistä (Taulukko 1). Näitä eri keinoja voidaan myös lisätä yhteen kuvaan satunnaisesti yhdistellen. Tämä mahdollistaa luomaan kohteesta erilaisia uusia ilmentymiä, jotka lopulta auttavat vahvistamaan objektien tunnistamista. (Vaith, 2021)

Taulukko 1 Kuvan augmentointimenetelmiä (Dilmegani, 2023)

Noise	Kohina	Rakeisuuden eli kohinan lisääminen kuvaan
Crop	Poisto	Kuvasta rajataan ja poistetaan reuna-alueita
Flip	Kääntö	Kuvan kääntö horisontaalisesti tai vertikaalisesti
Rotation	Kierto	Kuvan kiertäminen ympäri
Scale	Skaalaus	Kuvan skaalaus ulos- tai sisäänpäin tehden kohteesta suuremman tai pienemmän
Translation	Siirto	Kohteen siirto y- tai x- akselilla kuvassa
Brightness	Kirkkaus	Kuvan kirkkauden säätäminen
Contrast	Kontrasti	Kuvan kontrastin säätäminen
Color Augmentation	Värin lisäys	Kuvan pikselien väriarvojen muuntaminen

Saturation	Kylläisyys	Kuvan värien kylläisyyden säätäminen

Tässä opinnäytetyössä aineisto augmentoidaan ennen annotointia. Annotoinnissa datasetin kuviin lisätään kohteille erityiset tunnisteet. Augmentoinnin suorittaminen annotoinnin jälkeen saattaa muokata kuvien lisäksi kuvaan liitettyjä tunnisteita. Osa augmentointimenetelmistä muuttaa kohteen sijaintia kuvassa, mikä saattaa johtaa virheellisiin tunnisteisiin valmiiksi annotoiduissa kuvissa. (Chernytska, 2023) Annotoinnista kerrotaan lisää luvussa 4.

3.2 Opetus- ja validointidata

Koneoppimismallin opettamista varten luotu aineisto jaetaan osiin. Riippuen datan määrästä, aineisto voidaan jakaa kahteen tai kolmeen osaan. Jos dataa on runsaasti, se jaetaan satunnaisesti opetus-, validointi- ja testidataan. Suppeamman aineiston kohdalla käytetään usein jakoa vain opetus- ja validointidataan. Testaus suoritetaan tuolloin erikseen. (Nykänen, 2021)

Opetusdatalla koulutetaan malli tunnistamaan aineistosta ominaisuuksia ja rakenteita. Tätä opetusdataa syötetään neuroverkolle toistuvasti, mikä parantaa mallin oppimista. Mallin koulutuksen optimoimiseksi tarvitaan monipuolinen opetusdata, joka ottaa huomioon erilaiset datan kohteen ilmentymät eri skenaarioissa. (Baheti, 2023)

Validointidataksi kutsutaan aineistoa, jota käytetään mallin suorituksen vahvistamiseen koulutuksen aikana. Tämän prosessin avulla pystytään tarkastelemaan opetuksen kulkua ja seuraamaan, ollaanko oikealla suunnalla. Tarvittaessa mallia voidaan muokata ja säätää haluttuun suuntaan. Lisäksi validointi estää mallin ylioppimista eli tilannetta, jossa malli tunnistaa todella hyvin opetusdatan kohteet, muttei osaa soveltaa oppimaansa uuteen, ennalta näkemättömään aineistoon. (Baheti, 2023)

Testidata on aineisto, jota käytetään mallin testaamiseen koulutuksen jälkeen. Sen avulla voidaan selvittää, miten hyvin luotu malli toimii. (Baheti, 2023)

Data-aineiston jakaminen näihin edellä mainittuihin osioihin ja niiden keskinäiset suhteet riippuvat vahvasti aineiston koosta. Usein aineisto kuitenkin jaetaan suhteessa 70–80 % opetusdataa ja 20–30 % validointidataa. (Baheti, 2023)

4 Annotointi

Kuvien annotoinnilla tarkoitetaan prosessia, jossa objektintunnistuksessa käytettävälle kuvasetille määritellään tunnisteita. Annotoinnin suorittaa yleensä annotaattori eli henkilö, jolla on tarvittavat tiedot projektin vaatimuksista. Annotaattori pystyy lisäksi tunnistamaan kuva-aineistosta relevantit objektit ja lisäämään niihin oikeat tunnisteet. Tunnisteiden tarkkuutta voidaan lisätä käyttämällä useampia annotaattoreita kuvien annotointiin. Varsinainen annotointiprosessi sisältää yleensä useita vaiheita. Näitä ovat esimerkiksi objektien ja tunnisteiden määrittely, rajauslaatikoiden merkitseminen, annotaatioiden formaatin valinta sekä epätarkkojen tunnisteiden korjaaminen. (Datagen, ei pvm.)

Annotoinnin avulla luodaan koulutusaineisto tietokonenäkömallin opettamista varten. Tämä malli oppii siten itsenäisesti havaitsemaan objekteja ja tunnistamaan kuvia. Riippuen objektintunnistusprojektin laajuudesta ja tyypistä, kuva-aineiston kuville voidaan määritellä yksi tai useampi tunniste. (Datagen, ei pvm.)

4.1 Luokittelu, paikantaminen ja tunnistaminen

Kuvan luokittelu tarkoittaa menetelmää, jossa kohteen tunniste lisätään koko kuvaan. Kun kohteelle löydetään ja merkitään tunnisteen lisäksi sijainti kuvassa, puhutaan paikantamisesta. Usein paikantaminen muodostetaan suorakulmion muodossa koordinaattien avulla. Jos kuvasta löydetään ja merkitään useampia kohteita, puhutaan tunnistamisesta (Detection). Ero paikantamisen ja tunnistamisen välillä on siis kohteiden lukumäärä. Tällä pieneltä vaikuttavalla erolla on suuri merkitys syväoppimismallia suunniteltaessa. (Bonaccorso ym., 2018, ss. 640–641)

4.2 Yleiset ongelmat annotaatiossa

Kehittyneistä työkaluista ja algoritmeista huolimatta aineiston annotointi on yhä haasteellista. Neuroverkon kouluttamista varten tarvittavan suuren datamäärän huolellinen läpikäyminen ja annotointi vie paljon aikaa. Tähän tarkoitukseen löytyy paljon erilaisia työkaluja ja alustoja, joilla voidaan luoda laadukasta aineistoa syväoppimista varten. Ongelmana on usein sopivimman työkalun löytäminen. Tämän lisäksi suuren datasetin annotointi saattaa vaatia useampia annotaattoreita prosessin nopeuttamiseksi, mikä nostaa projektin kustannuksia. (Chutani, 2022)

Vaikka annotointiin on olemassa automatisoituja työkaluja, työ tarvitsee silti äärimmäistä tarkkuutta ja vaatii aina ammattimaista valvontaa. Laadun takaamiseksi manuaalisen ja mahdollisen automaattisen annotoinnin välillä on vallittava tasapaino. Kaiken lisäksi verkossa käytettävien työkalujen kohdalla pitää vielä ottaa huomioon kyberturvallisuus, sillä useimmat alustat vaativat koko aineiston lataamista verkkoon ennen annotoinnin aloittamista. Tämä saattaa olla mahdollinen tietoturvauhka. (Chutani, 2022)

4.3 Annotointityökalut

Kuvien annotointia varten on olemassa useita ilmaisia ja avoimen lähdekoodin työkaluja. Näillä työkaluilla voidaan esimerkiksi mahdollistaa rajauslaatikoiden merkitseminen kuviin ja tarjota järjestelmiä, joilla objekteille voidaan lisätä oikeat tunnisteet. (Datagen, ei pvm.)

Hyvällä annotointityökalulla on monia ominaisuuksia. Tällaisia ovat mm. selkeä ja helposti opittava käyttöliittymä. Käyttäjän on myös hyvä pystyä muokkaamaan työkalua omiin tarpeisiinsa. Laadukas annotointijärjestelmä on lisäksi tarkka sekä tukee useita tekniikoita mallin laadun mittaamiseen. (Datagen, ei pvm.)

Tähän opinnäytetyöhön kerätyt annotointityökalut on valittu edellä mainittuja ominaisuuksia silmällä pitäen. Erityisesti lähdetään tutkimaan työkaluja, jotka ovat maksuttomia tai tarjoavat maksullisen version ohella ilmaisen kokeiluversion. On myös tärkeää, että valitut työkalut tukevat YOLO-formaattia, sillä tässä opinnäytetyössä mallin koulutus tapahtuu YOLOv5-algoritmin avulla.

4.3.1 Make Sense

Make Sense on ilmainen avoimen lähdekoodin annotointityökalu, jota käytetään selaimella. Siinä on yksinkertainen käyttöliittymä, johon voidaan ladata datasetit ja annotoida kuvat. Lopuksi annotoimalla luodut tunnisteet voidaan ladata halutussa formaatissa. Make Sense on toimiva työkalu pieniin projekteihin, joissa aineiston kuvien lukumäärä voidaan laskea sadoissa ja annotointia suorittaa vain yksi henkilö. Ladattua aineistoa ei tallenneta alustalle eikä lähetetä eteenpäin, joten aineiston tietoturva on kunnossa. (Pokhrel, 2020)

Make Sense -työkalun käytössä on huomioitava seuraavat asiat. Alusta ei tarjoa mahdollisuutta tiimin sisäiseen työskentelyyn, joten annotointi tapahtuu vain yhden henkilön toimesta. Aineistoa ei voi myöskään ladata alustalle pakatussa muodossa, joten jokainen kuva pitää ladata yksitellen. Lisäksi projekteja ei voi tallentaa, joten selaimen sivun päivitys pyyhkii pois tehdyn työn ja koko projekti on aloitettava alusta. (Pokhrel, 2020)

4.3.2 Label Studio

Label Studio on avoimen lähdekoodin työkalu datan annotointiin ja eri tietotyyppien tutkimiseen. Sen käyttöä varten on asennettava erillinen sovellus, josta löytyy omat versionsa useimmille käyttöliittymille. Label Studio ei ole kuitenkaan käyttövalmis alusta, vaan jokainen käyttäjä itse luo sen tarpeidensa mukaisesti. (Liubimov, 2020)

Label Studio mahdollistaa usean projektin luomisen ja muokkaamisen sekä tiimiläisten lisäämisen omiin projekteihin. Myös aineiston lataaminen alustalle voidaan suorittaa suoraan paikalliselta koneelta tai pilvestä. Aineiston kuvat voidaan ladata joko yksitellen tai pakatussa kansiossa. (Liubimov, 2020)

4.3.3 LabelIMG

LabelIMG on suosittu graafinen ja avoimen lähdekoodin annotointityökalu, joka tarjoaa ainoastaan rajauslaatikoiden käytön kohteiden paikantamiseen ja havainnointiin kuvissa. Siinä on yksinkertainen käyttöliittymä ja se mahdollistaa valmiiden annotaatioiden tallentamisen useassa eri formaatissa. (Iakushechkin, 2023)

4.3.4 LabelFlow

LabelFlow on annotointityökalu, jonka lähdekoodi on avoimesti saatavilla Githubissa. Siinä on yksinkertainen ja käyttäjäystävällinen käyttöliittymä, jolla tunnistetaan kohteita aineiston kuvista. Tähän tarkoitukseen voidaan käyttää rajauslaatikoita ja monikulmioita (Polygons). (LabelFlow, ei pvm.-b) LabelFlow-työkalulla voi työskennellä paikallisesti ja offline-tilassa, jolloin käytössä on yksi työtila. Kirjautumalla online-tilaan käyttäjällä on mahdollisuus luoda rajattomasti erilaisia työtiloja. Online-tila mahdollistaa lisäksi työskentelyn yhdessä tiimiläisten kanssa. (LabelFlow, ei

pvm.-a) LabelFlow tarjoaa käyttäjilleen ilmaisen käytön aina 50 000 ladattuun kuvaan asti, minkä jälkeen alustan käyttö muuttuu maksulliseksi (LabelFlow, ei pvm.-c).

4.3.5 Roboflow

Roboflow on tietokonenäköjärjestelmä, jonka avulla voidaan luoda objektintunnistusmalleja. Mallien luomisen lisäksi sillä voidaan suorittaa datasettien esikäsittely, muokkaus sekä annotointi (Theophilus, 2022). Annotointia voidaan suorittaa rajauslaatikoiden, polygonien tai segmenttien avulla. Manuaalisen annotoinnin lisäksi Roboflow:ssa on mahdollista käyttää tekoälyä automaattiseen aineiston annotointiin. Se sisältää myös monipuoliset työkalut tiimin sisäiseen työskentelyyn. Saatavilla olevat työkalut ja alustan tarkemmat ominaisuudet määräytyvät käyttäjän tilauksen mukaan. Roboflow tarjoaa ilmaisen käytön lisäksi maksullisia vaihtoehtoja. (Roboflow, ei pvm.)

5 YOLO

YOLO (You Only Look Once) on suosittu objektintunnistusalgoritmi ja tietokonenäkömalli. Se käyttää neuroverkkoarkkitehtuuria nopeaan ja tarkkaan kohteen tunnistamiseen. YOLO-algoritmin tarkoitus on tunnistaa kohteen luokka sekä rajauslaatikko, jonka avulla objekti voidaan paikantaa kuvassa. YOLO ennustaa seuraavat arvot rajauslaatikosta suorittaessaan objektin tunnistusta: laatikon keskipisteen koordinaatit, laatikon leveyden sekä korkeuden. Lisäksi YOLO arvioi kohteen luokan arvon sekä ennusteen todennäköisyyden. (Zvornicanin, 2022)

Tässä opinnäytetyössä objektintunnistus suoritetaan käyttämällä YOLOv5-versiota. Siitä kerrotaan tarkemmin luvussa 5.3.

5.1 YOLO:n ominaisuudet

YOLO on hyvin nopea objektintunnistusalgoritmi. Sen sijaan, että tunnistettavia kuvia käytäisiin läpi kuvasta pikseli pikseliltä, YOLO katsoo koko kuvaa yhtenä kokonaisuutena. Kuva jaetaan osioihin ruudukon avulla, minkä jälkeen jokainen osa ruudukosta luokitellaan ja paikannetaan. Lopulta järjestelmä ennustaa rajauslaatikoiden sijainnin kuvassa regressioalgoritmia käyttäen. (Davies, 2022)

YOLO on yli kaksi kertaa nopeampi kuin muut vastaavat järjestelmät tunnistaen jopa 45 kuvaa sekunnissa. Kuvien ja videoiden lisäksi se on hyvin tehokas tunnistamaan kohteita reaaliaikaisesta datasta. Toinen huomionarvoinen ominaisuus YOLO-algoritmissa on sen kyky tunnistaa kohteet aineistossa täsmällisesti. Lisäksi se pystyy tunnistamaan aineistosta yli 9000 eri luokkaa samanaikaisesti. Avoimen lähdekoodin ja aktiivisen yhteisön avulla YOLO on kehittynyt vuosien aikana nopeasti ja jatkaa yhä tätä vahvaa kehityssuuntaa. (Kelta, 2022)

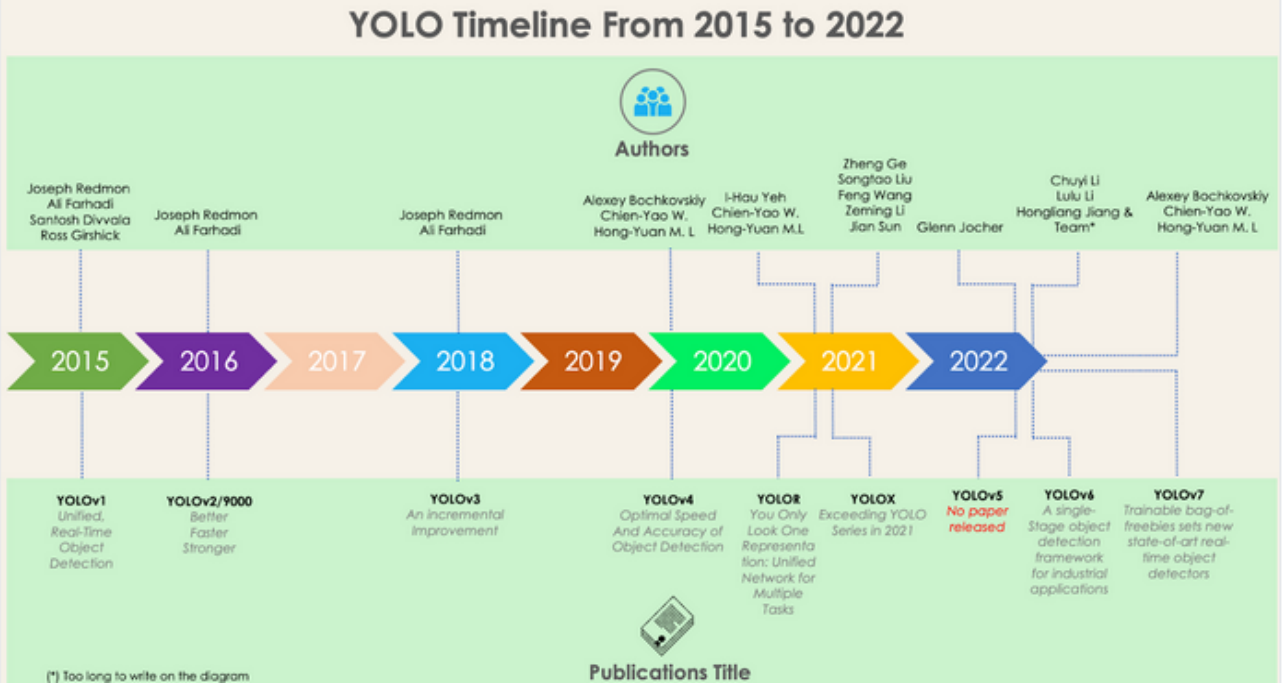
5.2 YOLO:n historia ja kehitys

Josep Redmon, Santosh Divvala, Ross Girshick ja Ali Farhadi julkaisivat ensimmäisen version YOLO-algoritmista vuonna 2015 tutkielmassaan ”You Only Look Once: Unified, Real-Time Object Detection”. Luokittelun sijaan he käyttivät regressiomallia objektintunnistusongelman

ratkaisemiseksi. Tämän järjestelmän avulla kuvaa katsotaan nimensä mukaisesti vain kerran ja silti pystytään ennustamaan mikä sekä missä objekti kuvassa on. (Redmon ym., 2016)

YOLO on kehittynyt vuosien mittaan ja siitä on luotu useita versioita. Ensimmäiset YOLO-versiot kehitettiin Darknet-neuraaliverkkojen avulla, YOLOv5 lähtien käyttöön otettiin Pytorch. (Kelta, 2022) PyTorch on Torch-kirjastoon perustuva avoimen lähdekoodin Python-kirjasto, jota käytetään paljon tietokonenäköprosesseissa (Exxact, 2022). Jokaista YOLO-versiota on kehitetty edeltäjäänsä paremmaksi lisäämällä ja muokkaamalla sen monia ominaisuuksia (Kuva 3) (Kelta, 2022). Kuva 3 näkyvien YOLO-versioiden lisäksi YOLO:sta on julkaistu viimeisin versio, YOLOv8 (Ultralytics, ei pvm.-c).

Kuva 3 YOLO:n kehitys 2015-2022 (Kelta, 2022)



5.3 YOLOv5

Glenn Jocher julkaisi YOLOv5:n vuonna 2020. Se oli ensimmäinen YOLO-versio, josta ei tehty julkista tutkielmaa ja joka toteutettiin käyttäen Pytorchia Darknetin sijaan. Toinen YOLOv5-version muutoksista oli Focus-kerroksen (Focus layer) lisääminen, mikä käytännössä korvasi neuroverkoston kolme ensimmäistä kerrosta. Tämä muutos vähensi kerrosten ja parametrien määrää lisäten merkittävästi koko järjestelmän nopeutta. (Kelta, 2022)

YOLOv5:sta on saatavilla neljä eri versiota: YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large) ja YOLOv5x (extra large). Mallien tarkkuus paranee näissä versioissa asteittain siirryttäessä pienemmästä versiosta suurempaan. Lisäksi mallien kouluttaminen vaatii joka versiossa eri määrän aikaa. Näistä YOLOv5-versioista tehokkain on YOLOv5x. (Solawetz, 2020b)

YOLOv5 augmentoi syötettyä aineistoa koulutuksen aikana. Skaalauksen ja värin lisäämisen lisäksi se käyttää augmentointitekniikkana mosaiikkimuokkausta (Mosaic Augmentation). Tässä tekniikassa neljä aineiston kuvaa yhdistetään sattumanvaraisessa suhteessa toisiinsa yhdeksi kuvaksi. Sen avulla on kuvista helpompi tunnistaa myös pienempiä kohteita. Manuaalisen aineiston augmentoinnin sekä YOLOv5:n sisäisen augmentoinnin yhdistelmällä voidaan varmistaa koulutetun mallin maksimaalinen suorituskyky. (Solawetz, 2020b)

6 Mallin kouluttaminen ja arviointi

Aineiston kokoamisen ja annotoinnin jälkeen mallin koulutus voidaan aloittaa. Koulutus voidaan toteuttaa paikallisella koneella tai käyttäen ulkoisia alustoja esim. pilvessä. (Seeed Studio, 2023)

Seuraavissa alaluvuissa käydään läpi eri vaihtoehtoja mallin kouluttamista varten. Ne ovat ilmaisia käyttää tai ne tarjoavat maksullisen käytön lisäksi ilmaisen kokeilujakson. Lisäksi ne kaikki käyttävät YOLOv5-algoritmia objektintunnistuksessa. Mallin kouluttamisen vaihtoehtoja tutkitaan erityisesti helppokäyttöisyyttä silmällä pitäen.

6.1 Ultralytics HUB

Ultralytics HUB on alusta, jonka avulla koulutetaan malleja. Alustalle voi ladata oman datasetin tai käyttää jo valmiita, olemassa olevia datasettejä. Selaimella toimiva Ultralytics HUB löytyy osoitteesta https://hub.ultralytics.com. Profiilin luominen ja kirjautuminen tapahtuu joko Googlen, GitHubin tai Applen tunnusten avulla. Tilin luominen onnistuu myös sähköpostin kautta. Ilmaiseen käyttäjätiliin saa 100 Gb tallenustilaa käyttöönsä. (Seeed Studio, 2023)

Oma datasetti ladataan alustalle pakatussa .zip-muodossa. Riippumatta miten ja millä alustalla datasetti on luotu ja annotoitu, datasetin kansiorakenne tulee olla Kuva 4 mukainen. (Krittapholchai, 2022)

Kuva 4 Datasetin rakenne (Krittapholchai, 2022)

```
-your project name (My project name is POM)
|-images
 |-train
 |-val
 |-test (optional)
|-labels
 |-train
 |-val
 |-test (optional)
```

Pakatussa kansiossa tulee lisäksi olla .yaml-tiedosto, jolla tulee olla sama nimi kuin projektilla. Tässä tiedostossa määritellään polut opetus-, validointi- sekä mahdollisesti testausdatan kuville ja tunnisteille. Lisäksi määritellään luokkien lukumäärä sekä nimet (Kuva 5). (Krittapholchai, 2022)

Kuva 5 .yaml-tiedoston sisältö (Krittapholchai, 2022)

```
path : #
train: images/train
val: images/val
test: images/test
nc: n
names : ['label 1' , 'label 2' , ... , 'label n']
```

Datasetin latauksen jälkeen valitaan millä YOLO-versiolla koulutus suoritetaan. Lisäksi valitaan koulutusalusta. Alustana voidaan käyttää Google Colabia tai paikallista ohjelmaa. Tarvittavien asennusten lisäksi kopioidaan tarvittavat koodit suoraan Ultralytics HUB:sta käytettävälle alustalle ja koulutus voidaan aloittaa. Sen kulkua voidaan seurata reaaliaikaisesti ja nämä tiedot löytyvät alustalta myös jälkikäteen tarkasteltaviksi. Luotua mallia on myös mahdollista testata. Lopulta valmis malli voidaan ladata halutussa tiedostomuodossa. (Krittapholchai, 2022)

6.2 Google Colab

Google Colab On Googlen luoma alusta python-ohjelmointikielen kirjoittamiseen ja suorittamiseen. Se sopii hyvin opiskeluun, koneoppimiseen ja data-analytiikkaan. Google Colab on täysin selainpohjainen, eikä vaadi asennusta. Käyttäjä saa ilmaiseksi käyttöönsä alustan tarjoamat tietojenkäsittelyresurssit, kuten GPU:n (Graphics Processing Unit, grafiikkaprosessori). Luodut koodit ajetaan käyttäjätilikohtaisilla virtuaalikoneilla ja tallennetaan Google Drive:in. (Google, ei pvm.)

Ultralytics tarjoaa mahdollisuuden YOLOv5-mallin kouluttamiseen selaimella Google Colab -alustaa käyttäen. Sivuston valmiissa muistiossa on selkeät ohjeet vaihe vaiheelta objektintunnistusmallin luomiseen. (Ultralytics, ei pvm.-a) Sivusto löytyy osoitteesta https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb.

6.3 Paikallinen tietokone

YOLOv5-objektintunnistus voidaan ajaa suoraan paikalliselta tietokoneelta. YOLOv5-kirjasto kloonataan ja requirements.txt asennetaan Komento 1 mukaisesti. Tätä varten koneelle tulee olla asennettu Python 3.7.0. tai sitä uudempi versio. (Ultralytics, ei pvm.-b)

Komento 1 YOLOv5-kirjaston kloonaus ja requirements-tiedoston asennus (Ultralytics, ei pvm.-b)

```
git clone https://github.com/ultralytics/yolov5 # clone
cd yolov5
pip install -r requirements.txt # install
```

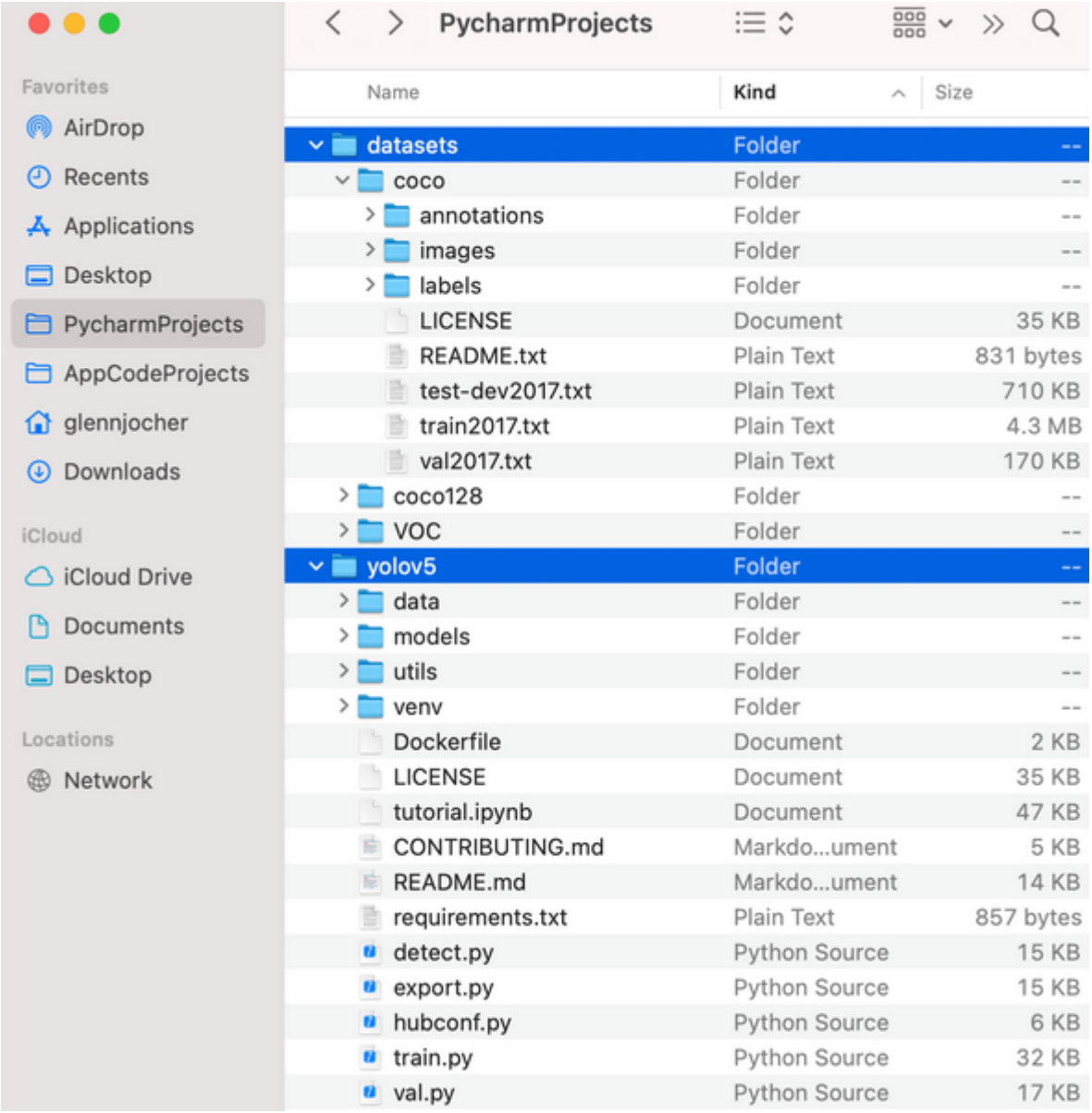
Seuraavaksi luodaan .yaml-tiedosto datasettiä varten. Sen avulla määritellään datasetin sekä opetus-, validointi- ja testitiedostohakemistojen polut. Lisäksi määritellään datasetin luokkien lukumäärä sekä jokaisen luokan nimi (Ohjelmakoodi 1). (Ultralytics, ei pvm.-b)

Ohjelmakoodi 1 Yaml-tiedoston sisältö (Ultralytics, ei pvm.-b)

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt,
or 3) list: [path/to/imgs1, path/to/imgs2, ..]

path: ../datasets/coco128  # dataset root dir
train: images/train2017  # train images (relative to 'path') 128 images
val: images/train2017  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes (80 COCO classes)
names:
  0: person
  1: bicycle
  2: car
  ...
  77: teddy bear
  78: hair drier
  79: toothbrush
```

Tämän jälkeen tuodaan valmiiksi annotoitu datasetti YOLO-formaatissa projektikansion juureen. Kansiorakenteen tulee olla Kuva 6 mukainen. (Ultralytics, ei pvm.-b)

Kuva 6 YOLOv5-projektikansion rakenne (Ultralytics, ei pvm.-b)



Koulutuksen aloittamiseksi valitaan käytettävä YOLOv5-versio. Lisäksi koulutuksen aloittavassa Komento 2 määritellään mm. käytettävä datasetti, sen erän koko sekä kuvien koko. (Ultralytics, ei pvm.-b)

Komento 2 Mallin koulutusesimerkki (Ultralytics, ei pvm.-b)

```
# Train YOLOv5s on COCO128 for 3 epochs
$ python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --
weights yolov5s.pt
```

Mallin koulutuksen tulokset löytyvät runs/train/ -kansiosta. Mahdolliset lisäkoulutusten tulokset löytyvät edellisen kansion alakansioista runs/train/exp2, runs/train/exp3 jne. (Ultralytics, ei pvm.-b)

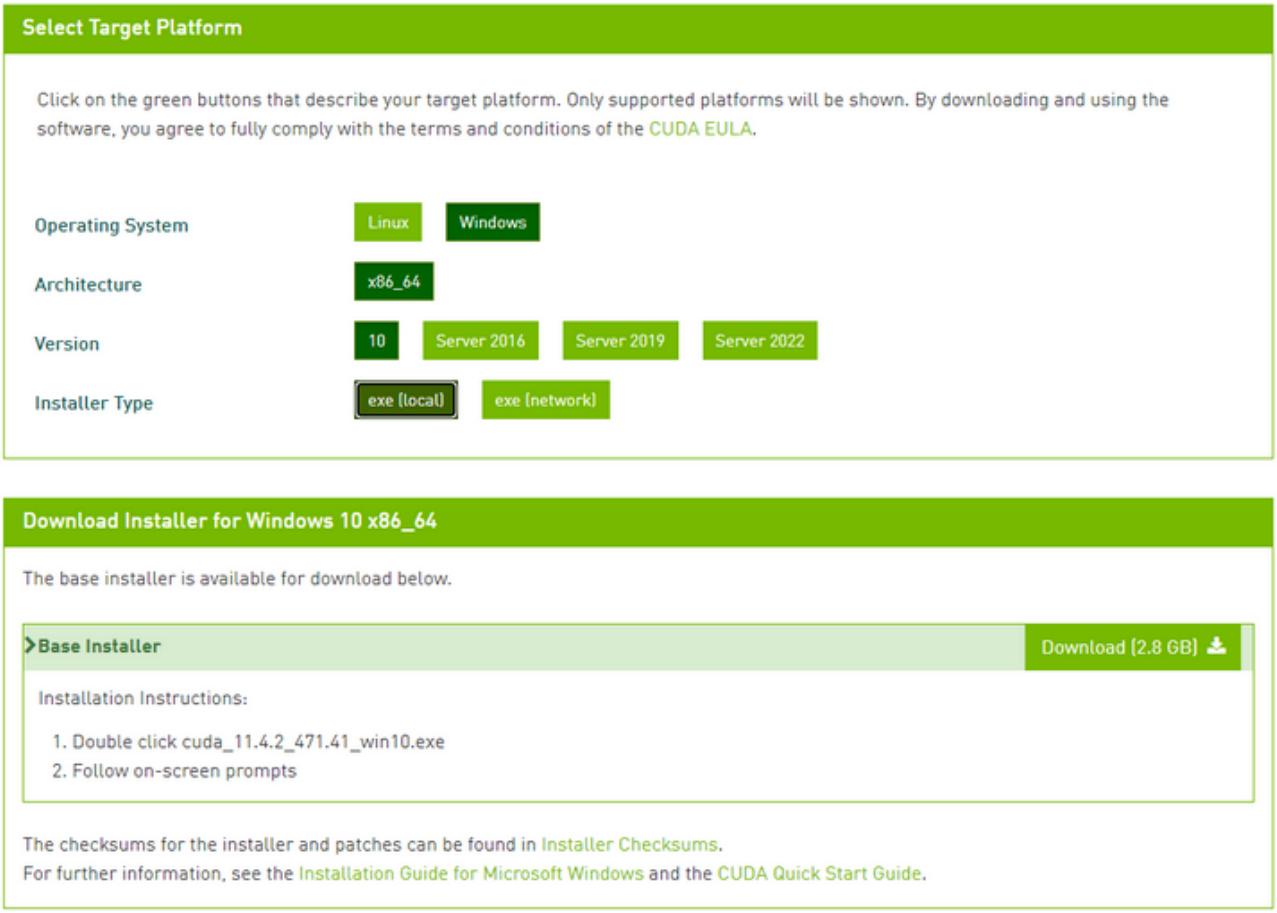
6.4 NVIDIA Cuda Toolkit ja PyTorch

Ilman grafiikkaprosessoria, CPU:n varassa toimiva YOLO, kykenee prosessoimaan vain muutaman kuvan sekunnissa. Grafiikkaprosessorin käyttöönotolla voidaan nostaa kuvienprosessointi jopa yli 45:n kuvaan sekunnissa. Tämä tarkoittaa siis yli 20-kertaista nopeutta CPU:n varassa toimivaan YOLO:on verrattuna. (Emmytheo 24/7, 2019)

NVIDIA Cuda Toolkit tarjoaa kehitysympäristön sovelluksille, jotka vaativat korkeatasoisen grafiikkaprosessorin toimiakseen. Tämän ympäristön avulla voidaan kehittää, optimoida ja ottaa käyttöön sovelluksia hyödyntäen tietokoneen omaa grafiikkaprosessoria. (NVIDIA, 2013)

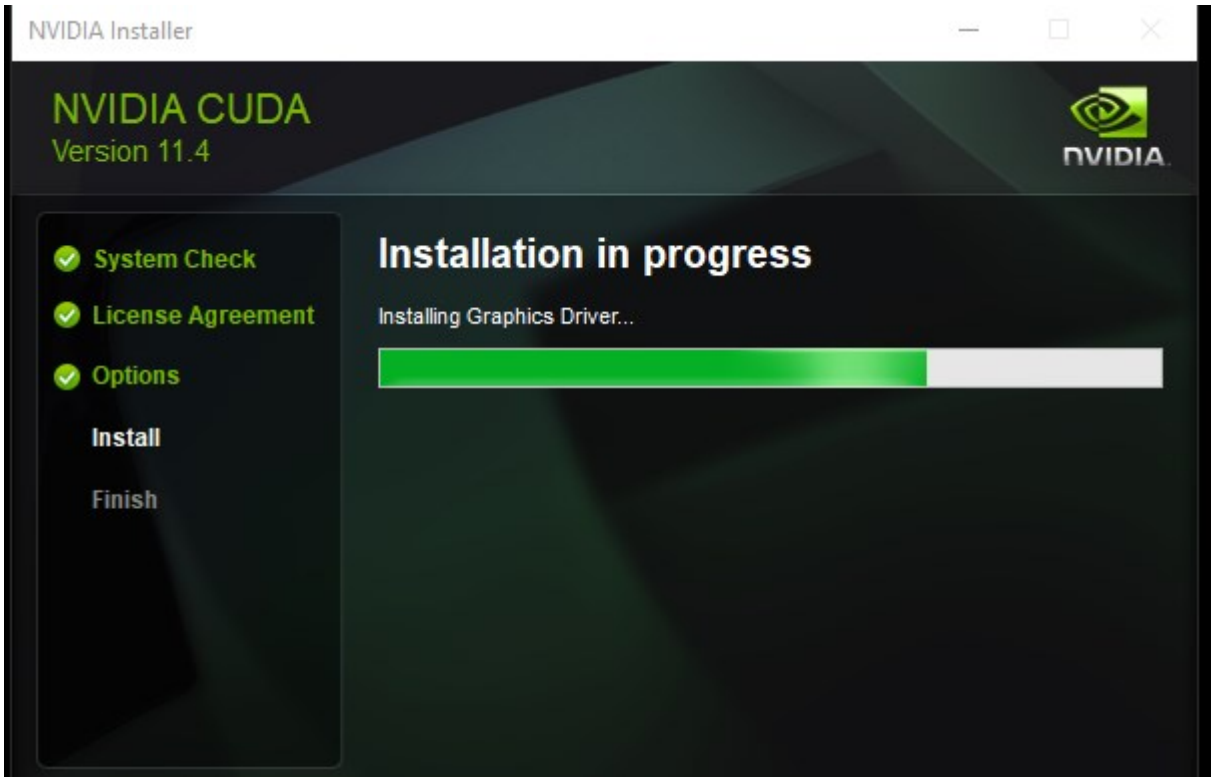
Tämän työkalun avulla saadaan lisää nopeutta ja tehokkuutta objektintunnistusmallin luomiseen paikallisella koneella. NVIDIA Cuda Toolkit voidaan ladata sivustolla https://developer.nvidia.com/cuda-downloads. Ennen latausta valitaan oikeat asetukset käytetyn alustan mukaisesti (Kuva 7). (Davies, 2022)

Kuva 7 NVIDIA Cuda Toolkit -lataus (Davies, 2022)



Latauksen jälkeen Cuda voidaan asentaa. Tietokone täytyy mahdollisesti käynnistää uudelleen asennuksen valmistuttua (Kuva 8). (Davies, 2022)

Kuva 8 NVIDIA Cuda -asennus (Davies, 2022)



Seuraavaksi asennetaan PyTorch käytettävän alustan mukaisesti. Sivustolla https://pytorch.org/get-started/locally/ opastetaan asennuskomennon luomisessa. Luotu komento suoritetaan paikallisen koneen Python Scripts -kansiossa (Komento 3). (Davies, 2022)

Komento 3 PyTorch-asennuskomento (Davies, 2022)

Run this Command:

```
pip3 install torch==1.9.0+cu111 torchvision==0.10.0+cu111 torchaudio===0.9.0 -f
https://download.pytorch.org/whl/torch_stable.html
```

6.5 Mallin arviointi

Koulutuksen valmistuttua YOLO tulostaa yhteenvedon tapahtumista. Siinä näkyy vaiheiden (Epoch) määrä sekä opetukseen kulunut kokonaisaika. (Seeed Studio, 2023) Sarake P (Precision) kuvaa koulutuksen tarkkuutta. Tarkkuus mittaa sitä, kuinka moni ennustetuista tilanteista osoittautui oikeiksi. Sarakkeessa R (Recall) esitetään koulutuksen herkkyys. Herkkyydellä tarkoitetaan niiden tilanteiden osuutta, jossa malli ennusti kohteen oikein. (Dang, 2022) mAP-arvoa käytetään usein yleisesti kuvaamaan objektintunnistusmallin tarkkuutta ja vakautta. Se ottaa huomioon mallin

tarkkuuden ja herkkyyden lisäksi niiden eron. (Shah, 2023) Tämä arvo perustuu IoU-mittariin (Intersection over Union). Sen avulla tarkastellaan kunkin luokan kohteelle luodun rajauslaatikon ja kohteen todellisen rajauslaatikon päällekkäin osuvaa pinta-alaa. mAP@0.5 antaa prosentuaalisen arvon niistä tilanteista, joissa näiden rajauslaatikoiden yhteinen pinta-ala ylittää 50 %. mAP@0.5:0.95 käy läpi edellä kuvatun menetelmän viiden prosenttiyksikön välein 50 %:n ja 95 %:n ylittävien päällekkäisyyksien välillä ja luo niistä keskiarvon. Usein YOLO-objektintunnistusmallin toimivuutta arvioidaan lähinnä mAP@0.5:0.95-arvon avulla. Mitä korkeampi tämä arvo on, sitä tarkempi ja herkempi luotu malli on (Kuva 9). (Solawetz, 2020a)

Kuva 9 Esimerkki YOLOv5-koulutuksen yhteenvedosta (Seeed Studio, 2023)

```
100 epochs completed in 2.201 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 6.6MB
Optimizer stripped from runs/train/exp/weights/best.pt, 6.6MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model Summary: 280 layers, 3089188 parameters, 0 gradients, 4.2 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100%|          | 2/2 [00:05<00:00,  2.52s/it]
                 all        105        893      0.916      0.909       0.95      0.713
                leaf        105        717       0.86       0.83      0.906      0.608
          pink flower       105        176      0.972      0.989      0.994      0.817
```

Valmis malli tallentuu projektikansioon runs/train/exp/weights nimellä best.pt. (Seeed Studio, 2023).

7 Kehittämistyön tavoite ja tarkoitus

Tämän työn tarkoituksena on tutkia toimeksiantajan uutta palvelua varten, mitä tulee ottaa huomioon luodessa datasettiä objektintunnistusta varten. Lisäksi tutkitaan mahdollisia menetelmiä mallin luomista varten sekä tarkastellaan niiden eroja. Tavoitteena on löytää selkeä, edullinen ja tehokas tapa käsitellä kuva-aineistoa sekä luoda objektintunnistusmalli. Tulosten perustella toimeksiantaja pystyy luomaan toimintaohjeistuksen objektintunnistuspalvelun eri vaiheisiin varmistaen projektityöskentelyn johdonmukaisuuden tiimiläisten välillä.

Jotta objektintunnistuksen aineistoa voidaan alkaa luomaan ja malleja kouluttamaan, täytyy ymmärtää tekoälyn ja koneopin perusteita. Mahdollisten käytettävien alustojen ja työkalujen määrä on suuri, joten niiden toimintoja ja eroja on tärkeää tarkastella. Aihealueen laaja-alaisuuden takia on myös tärkeää, että osataan rajata ja määritellä juuri ne osa-alueet, jotka ovat välttämättömiä tässä opinnäytetyössä luotavalle objektintunnistusprosessille. Lopuksi on varmistettava, että osataan analysoida työstä saatuja tuloksia.

Objektintunnistusmallia varten tarvitaan ensin referenssikuvat, joilla lähdetään rakentamaan suurempaa datasettiä. Tässä opinnäytetyössä kuvat kerätään Canva-ohjelman avulla, sillä sieltä löytyy suuri kirjasto valokuvia tähän tarpeeseen. Valokuvista valitaan 30 kappaletta mahdollisimman monipuolisesti kohdetta sisältäviä kuvia eri ympäristöissä ja kuvakulmissa. Toimeksiantajan projektia varten tutkitaan, onko tämä referenssikuvien määrä tarpeeksi suuri vai tarvitaanko laajempi kuva-aineisto.

Kuva-aineiston käsittelyä halutaan automatisoida mahdollisimman paljon. Tähän tarpeeseen löytyy erilaisia työkaluja ja alustoja, jotka ovat pääasiassa maksullisia. Useat työkalut tarjoavat myös ilmaisia kokeiluversioita, mutta niiden käyttö on selkeästi rajoitetumpaa. Jotta saadaan selkeä kuva datasetin vaatimuksista ja samalla pidetään kustannukset vielä kurissa, aineistoa muokataan tämän työn osalta manuaalisesti. Tämä lisää kuva-aineiston ja sen vaatimusten kontrollia sekä auttaa ymmärtämään syvemmin, mitä asioita on otettava aineiston käsittelyssä huomioon. Tämä aineiston vaatimusten selkeyttäminen auttaa jatkossa toimeksiantajayrityksen päätöksentekoa mahdollisen automatisointityökalun valinnassa.

Referenssikuvien augmentointi on mahdollista suorittaa automaattisesti. Tähän tarkoitukseen löytyy valmiita työkaluja. Lisäksi on mahdollista suoraan ohjelmoida koodi, joka suorittaa augmentoinnin halutulla tavalla. Tässä opinnäytetyössä augmentointi kuitenkin suoritetaan manuaalisesti Krita-kuvankäsittelyohjelmalla. Tällä tavalla voidaan selvittää, minkälaisia augmentointeja voidaan luoda sekä niiden toimivuus objektintunnistusmallin luomisen kannalta.

Datasetin annotointiin valittiin Label Studio sen monipuolisuuden ja helppokäyttöisyyden takia. Lisäksi se mahdollistaa tiimityöskentelyn annotoinnissa, mikä on toimeksiantajan projektia ajatellen tärkeä ominaisuus annotointityökalussa. Label Studio on mahdollista myös yhdistää omaan web-sovellukseen Label Studio Frontendia käyttäen, mikä on erinomainen ominaisuus toimeksiantajan tulevaa projektia silmällä pitäen.

Käytännön osiossa käydään läpi usea eri vaihtoehto objektintunnistusmallin luomiseen. Käyttämällä eri tekniikoita voidaan vertailla valmiiden mallien toimivuutta ja tarkkuutta. Lisäksi voidaan arvioida, mikä tekniikka olisi toimeksiantajan projektia ajatellen tehokkain vaihtoehto mallin luomiseen. Vaihtoehtoja läpikäydessä pidetään mielessä myös, minkälaista laitteistoa kukin tekniikka tarvitsee ja voidaanko käytännön toteutus suorittaa mahdollisimman mutkattomasti.

8 Optimoidun datasetin luominen objektintunnistukseen

Tässä opinnäytetyössä luodaan datasetti, jonka avulla halutaan tunnistaa Suomen lippuja kuvista. Datasettiä laajennetaan augmentoimalla ja kohteet annotoidaan aineistosta. Tarkoituksena on testata erilaisia datasettejä, jotta saadaan selville, millä keinoin saadaan mahdollisimman tarkka ja toimiva objektintunnistusmalli Suomen lipun tunnistamiseen kuvista. Lisäksi testataan erilaisia tapoja luoda objektintunnistusmalli sekä tarkastellaan valmiiden mallien eroja.

8.1 Referenssikuvat

Referenssikuvien kokoaminen tehdään käyttämällä Canva-työkalua. Luotuun datasettiin kerätään 30 erityyppistä ja -laatuista kuvaa, joissa kaikissa on yksi tai useampi Suomen lippu (Kuva 10). Kaikkien kuvien kooksi määritetään 640 x 640 pikseliä, sillä tämä on YOLOv5-objektintunnistusjärjestelmän käyttämä oletuskuvakoko. Käyttämällä tätä kuvakokoa pystytään myös kontrolloimaan datasetin vaatimaa tilaa alustalla, joka suorittaa lopullisen objektintunnistuksenmallin kouluttamisen. Kuvat nimetään numeroilla 1.png, 2.png jne. Tämä selkeyttää tunnisteiden lisäämistä kuviin annotointivaiheessa.

Referenssikuvien kokoamisen jälkeen koko datasetti yleensä siivotaan ja samalla varmistetaan sen laadukkuus. Koska tätä aineistoa ei ole koonnut ulkopuolinen taho, voidaan olla varmoja sen laadukkuudesta, eikä siivoamiselle ole tarvetta.

Kuva 10 Kollaasi referenssikuvista

8.2 Augmentointi

Jokainen datasettiin lisätty kuva augmentoidaan manuaalisesti käyttämällä Krita-kuvankäsittelyohjelmaa. Menetelminä käytetään Taulukko 1 lueteltuja augmentointimenetelmiä sattumanvaraisesti ja yhdistellen (Kuva 11). Jotta myöhemmin voidaan tarkastella alkuperäisen Flags30-datasetin ja augmentoidun aineiston eroja, luodaan uusi datasetti, Flags300. Alkuperäisen Flags30-datasetin 30:n kuvan augmentoinnin jälkeen uudessa Flags300-datasetissä on 300 erilaista kuvaa Suomen lipusta. Augmentoinnin suorittaminen manuaalisesti kesti yhdeltä henkilöltä noin kaksi tuntia.

Kuva 11 Referenssikuvan augmentointi

8.3 Annotointi

Datasetin kuvien annotointiin käytetään Label Studio -työkalua. Sen asennus toteutetaan komentokehotteessa Komento 4 mukaisesti.

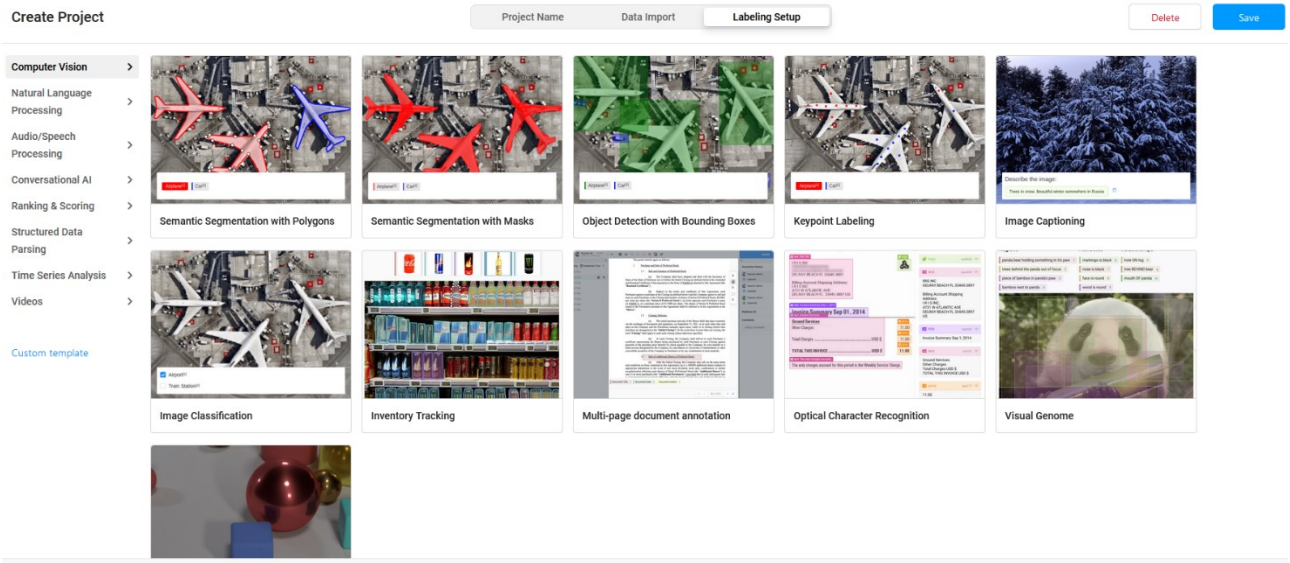
Komento 4 Label Studion asennus ja käynnistys

```
pip install label-studio # asennus
label-studio # käynnistys
```

Label Studio käynnistyy selaimessa paikallisella koneella osoitteessa localhost:8080. Uusi projekti aloitetaan valitsemalla ”Create Project”. Projektille lisätään nimi sekä tarvittaessa kuvaus Description-kenttään. Data Import -välilehdellä ladataan valmis datasetti valitsemalla Upload Files. Labeling Setup -välilehdellä määritellään asetukset kuvien tunnistamiselle. Label Studiossa on

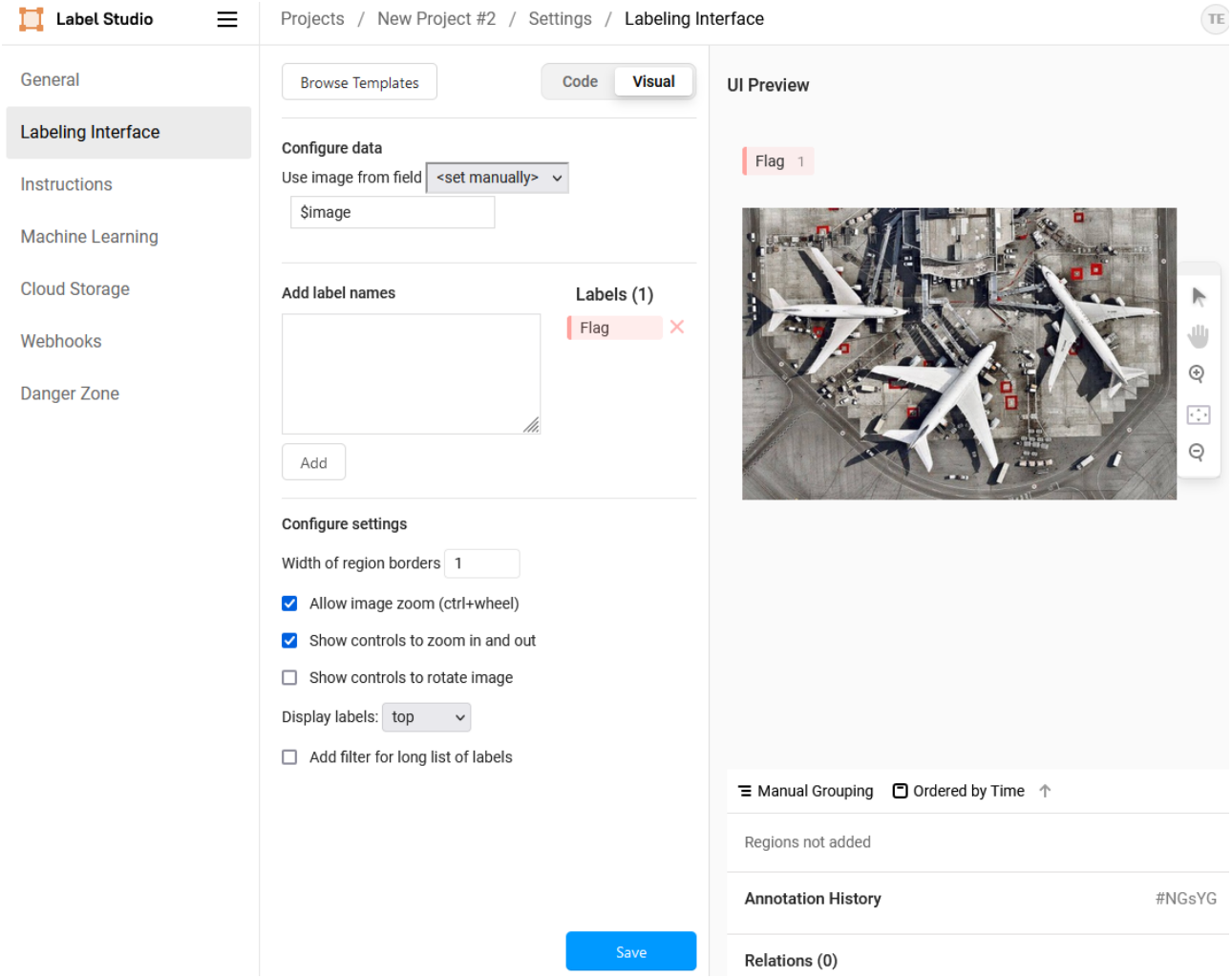
monta valmista pohjaa erityyppisille annotointiprojekteille. Tässä työssä käytetään Label Studion valmista tietokonenäköpohjaa ”Object Detection with Bounding Boxes” (Kuva 12).

Kuva 12 Label Studion valmiit annotointipohjat



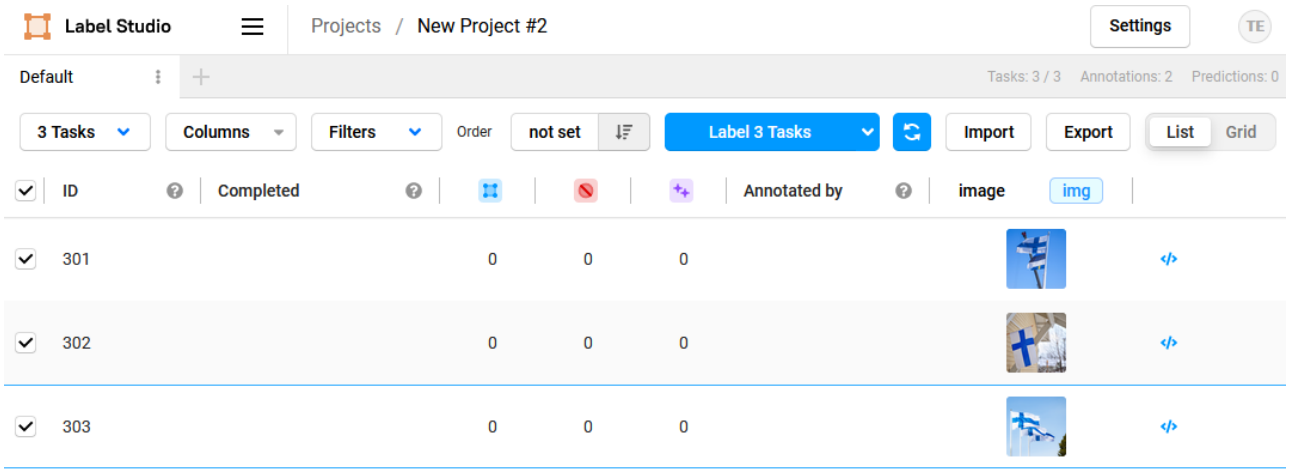
Labeling Setup -välilehdellä lisätään tunnistettavien kohteiden luokkien nimet Add label names -kenttään. Jokainen uusi luokka lisätään omalle riville. Configure data -kohtaan jätetään oletusasetukset referenssikuvien hakemiselle manuaalisesti. Lisäksi kohdassa Configure settings voidaan muokata lisäasetuksia annotointityökalun käyttöliittymään (Kuva 13).

Kuva 13 Label Studion asetukset



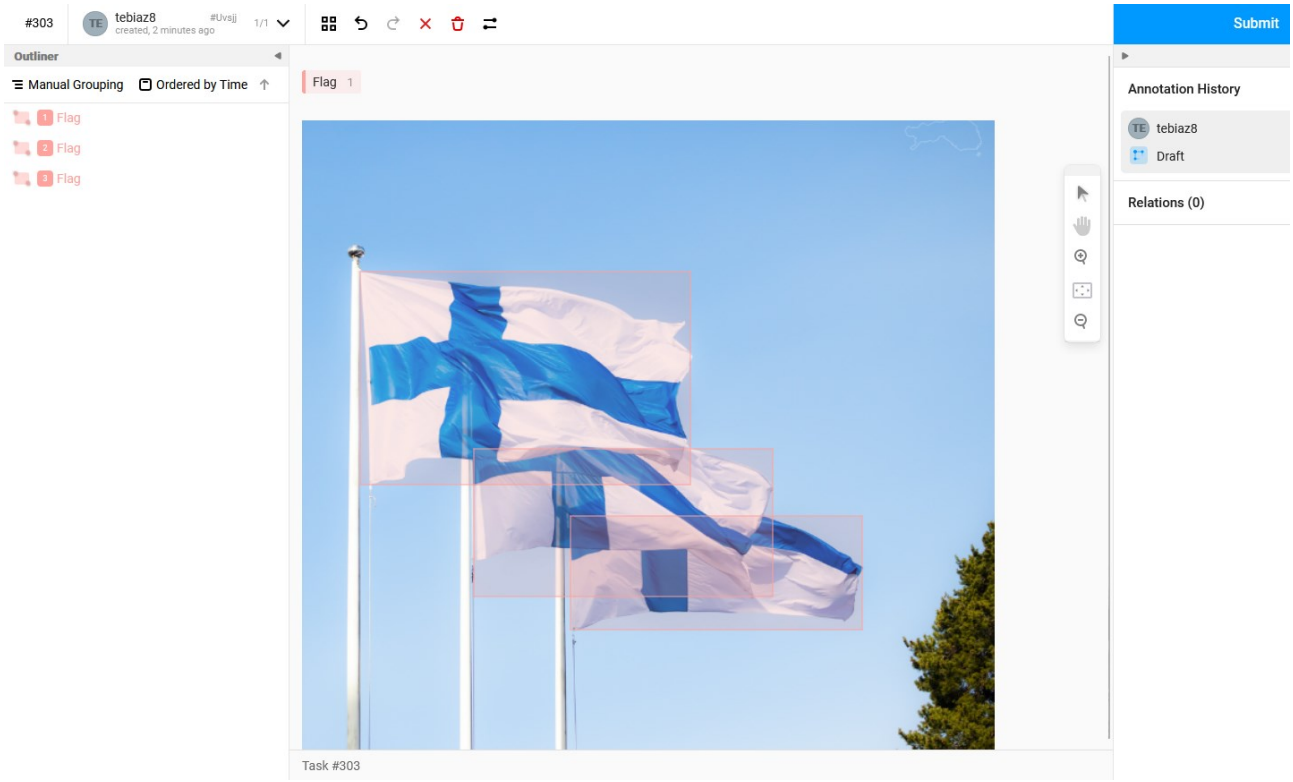
Asetusten tallentamisen jälkeen päästään projektin etusivulle, jossa näkyvät aikaisemmin ladatut referenssikuvat. Valikosta valitaan annotoitavat referenssikuvat ja aloitetaan annotointi valitsemalla Label Tasks (Kuva 14).

Kuva 14 Label Studion annotointiaineiston valinta



Tunnisteen lisääminen kuvaan aloitetaan valitsemalla oikea luokka tunnisteelle. Tämän jälkeen tunnistettava objekti merkitään rajauslaatikolla. Rajauslaatikon on oltava mahdollisimman tiukasti objektin ympärillä. Tarvittaessa kuvaa voidaan zoomata lähemmäs tarkkuuden lisäämiseksi. Kaikki kuvassa näkyvät objektit, joista on näkyvillä yli 50 %, tulee merkitä. Jos kohde on toisen kohteen takana tai muuten osittain näkyvissä, rajauslaatikko merkitään kohteen näkyvissä olevien linjojen mukaisesti (Kuva 15).

Kuva 15 Label Studion rajauslaatikoiden lisäys



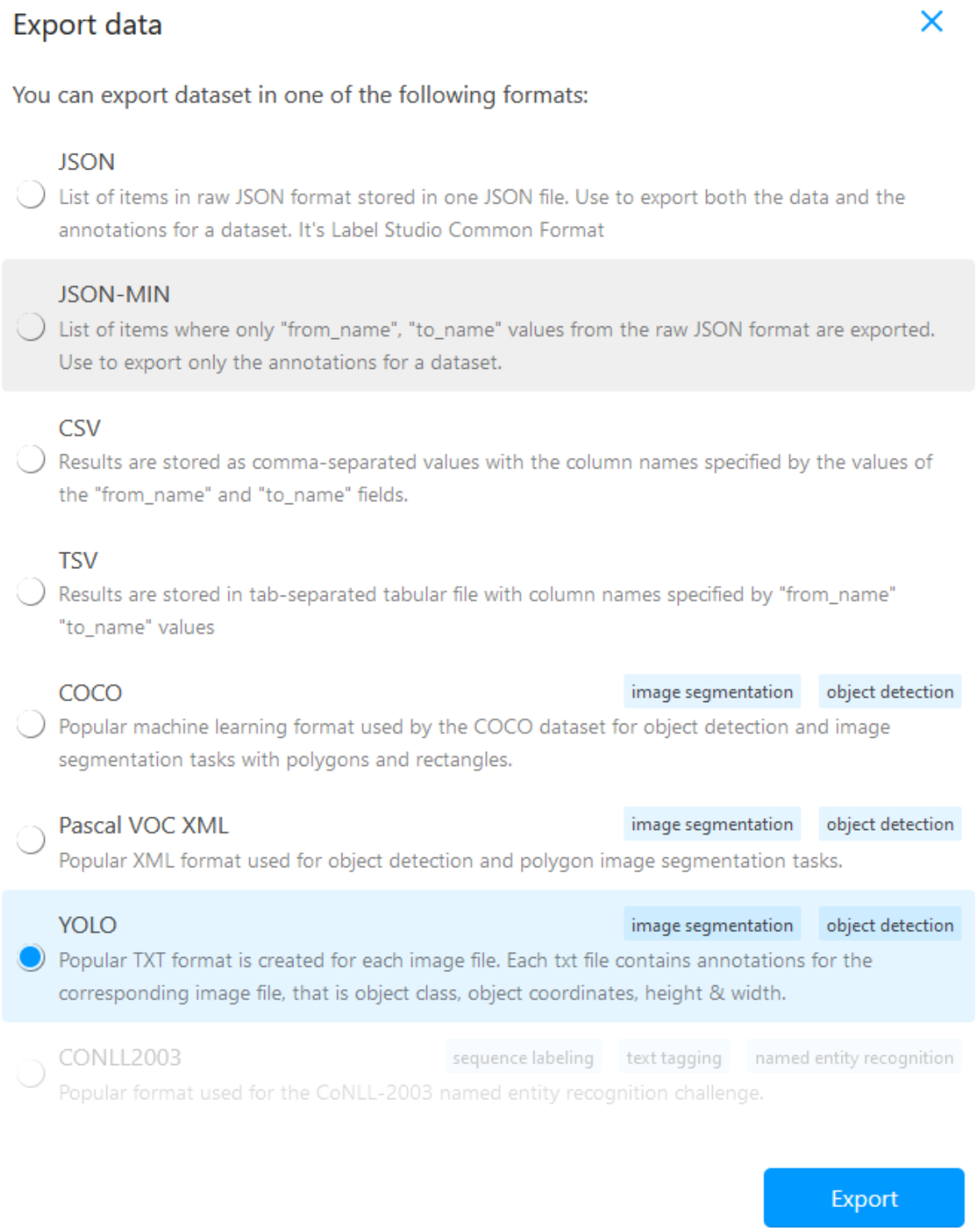
Optimoidun datasetin lisäksi annotoidaan sama aineisto uudestaan. Tämä datasetti on nimeltään BadFlags. Tällä kertaa annotoidaan kuvat huolimattomasti, jotta voidaan vertailla heikommin annotoidun ja optimoidun datasetin eroja. Tämän aineiston kohteita ei rajata tiukasti ja osa kohteista jätetään merkitsemättä. Osaan kuvista luodaan lisäksi rajauslaatikot virheellisesti väärään kohteeseen. Lisäksi kohteissa, jotka ovat vain osittain näkyvissä, merkintä suoritetaan sattumanvaraisesti (Kuva 16).

Kuva 16 Virheellisesti annotoidut objektit

Submit-painikkeella tallennetaan tehdyt muutokset ja ohjelma siirtyy seuraavaan kuvaan. Kun kaikki kuvat ovat annotoitu, kaikki tiedostot siirretään paikalliselle koneelle pakatussa muodossa

painamalla Export-nappia projektin etusivulla. Koska objektintunnistusmalli luodaan YOLOv5-algoritmia käyttäen, formaatiksi valitaan YOLO (Kuva 17).

Kuva 17 Label Studion siirrettävän datan formaatti



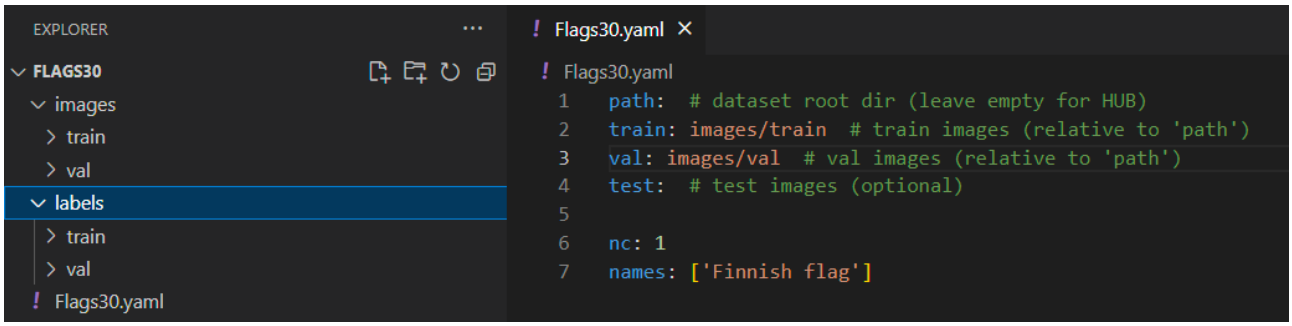
Luodussa kansiossa on kaksi alikansiota, images ja labels. Jokaisella images-kansion kuvatiedostolla on samanniminen txt-tiedosto labels-kansiossa. Txt-tiedosto sisältää tiedot

annotoidun kohteen luokasta, kohteen keskipisteen koordinaatit sekä kohteen leveyden ja korkeuden kuvassa.

8.4 Mallin koulutus ja tulokset

Mallin kouluttamiseen tehdään neljä kansiota neljälle erikokoiselle ja erilaatuiselle datasetille. Ensimmäisessä datasetissä on alkuperäiset 30 kuvaa muokkaamattomina sekä niiden tunnisteet. Toisessa datasetissä on augmentoidut 300 kuvaa tunnisteineen. Lisäksi luodaan vastaavat datasetit heikosti annotoiduille kuville. Kuvat jaetaan suhteessa 80 % koulutusdata ja 20 % validointidata. Lisäksi molempiin kansioihin luodaan .yaml-tiedosto (Kuva 18).

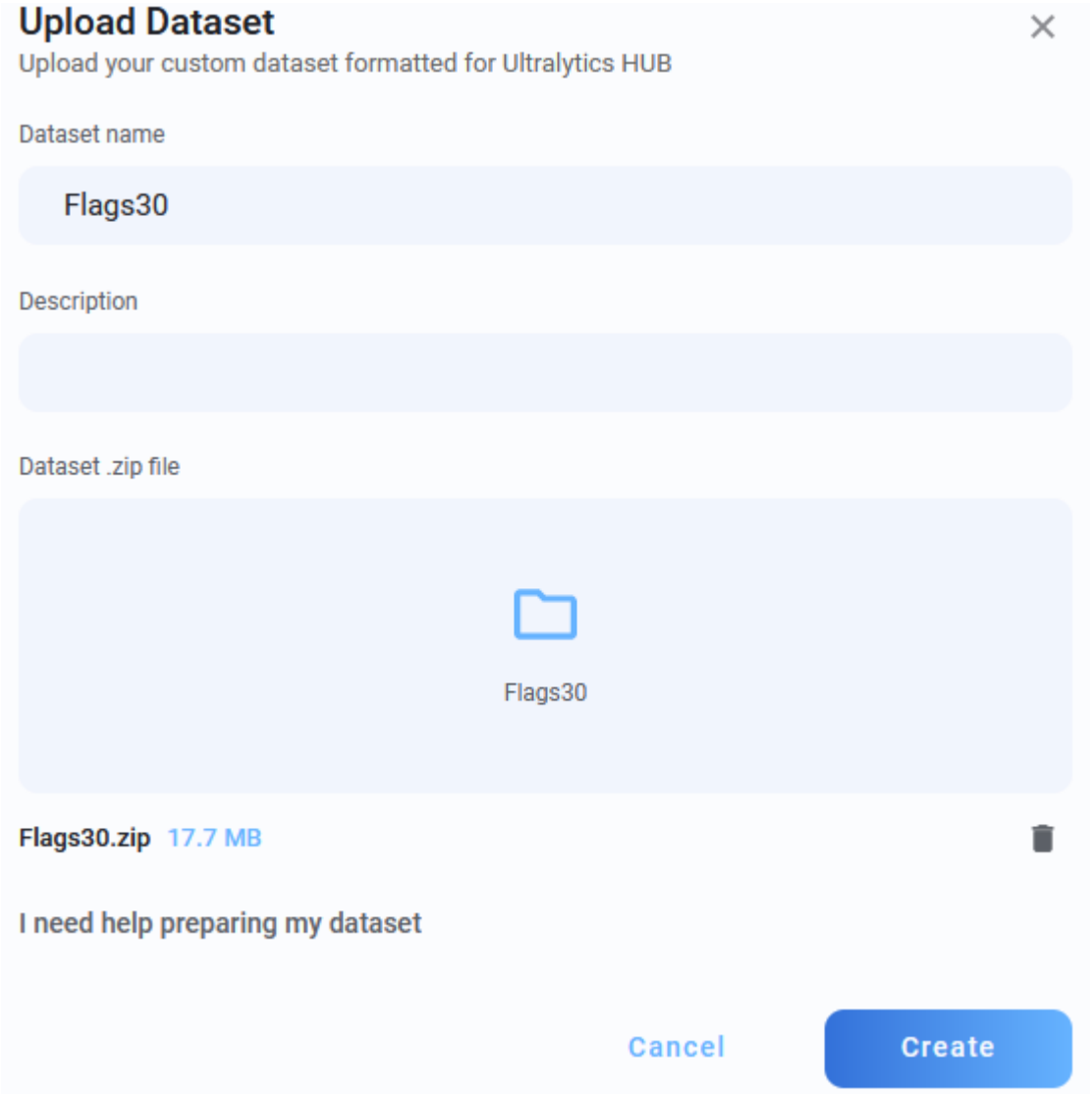
Kuva 18 Datasetin kansiorakenne ja sisältö



8.4.1 Ultralytics Hub ja Google Colab

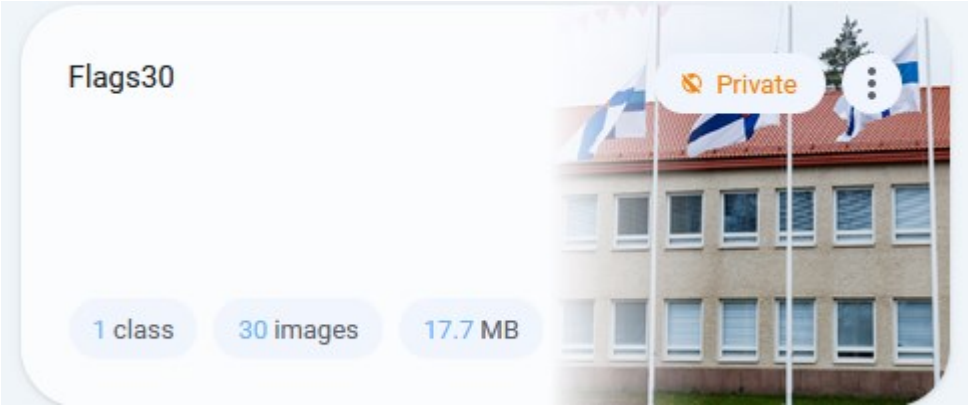
Ultralytics Hub:n käyttöä varten luodaan ilmainen käyttäjätili alustalle. Käyttöliittymä on selkeä ja sisältää selkeän ohjeistuksen mallin kouluttamiseen. Oman datasetin lataamista varten datasettikansio pakataan zip-formaattiin (Kuva 19).

Kuva 19 Datasetin lataus



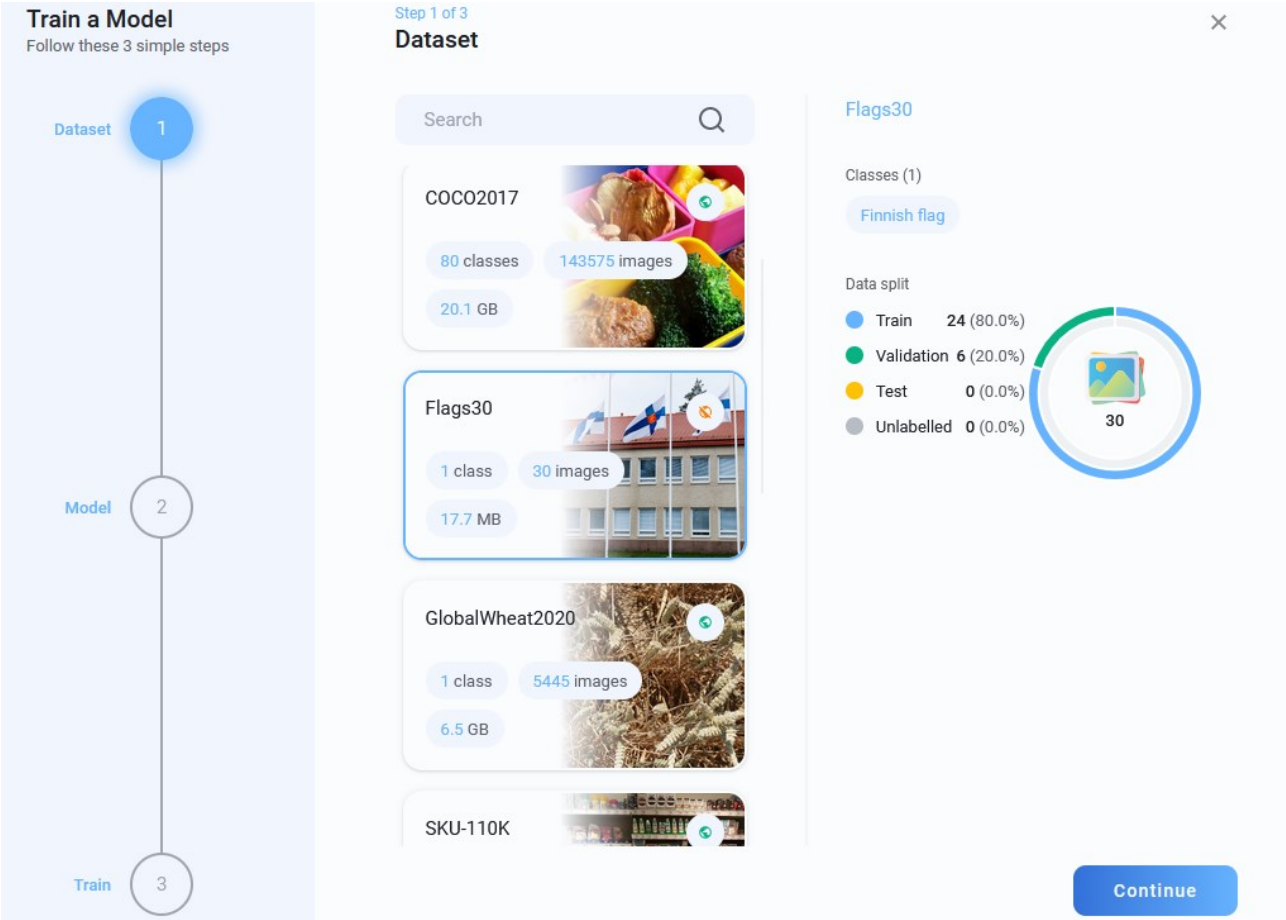
Create-nappulasta painamisen jälkeen lataus alkaa. Usean minuutin jälkeen lataus vieläkin jatkuu, eikä valmistu. Selaimen sivun päivityksen jälkeen palataan Ultralytics Hubin etusivulle ja tarkistetaan Datasets-sivulta, onnistuiko lataus. Ladattu Flags30-datasetti löytyy hakemistosta sisältäen 30 kuvaa ja yhden luokan (Kuva 20).

Kuva 20 Oma datasetti Ultralytics Hub:ssa



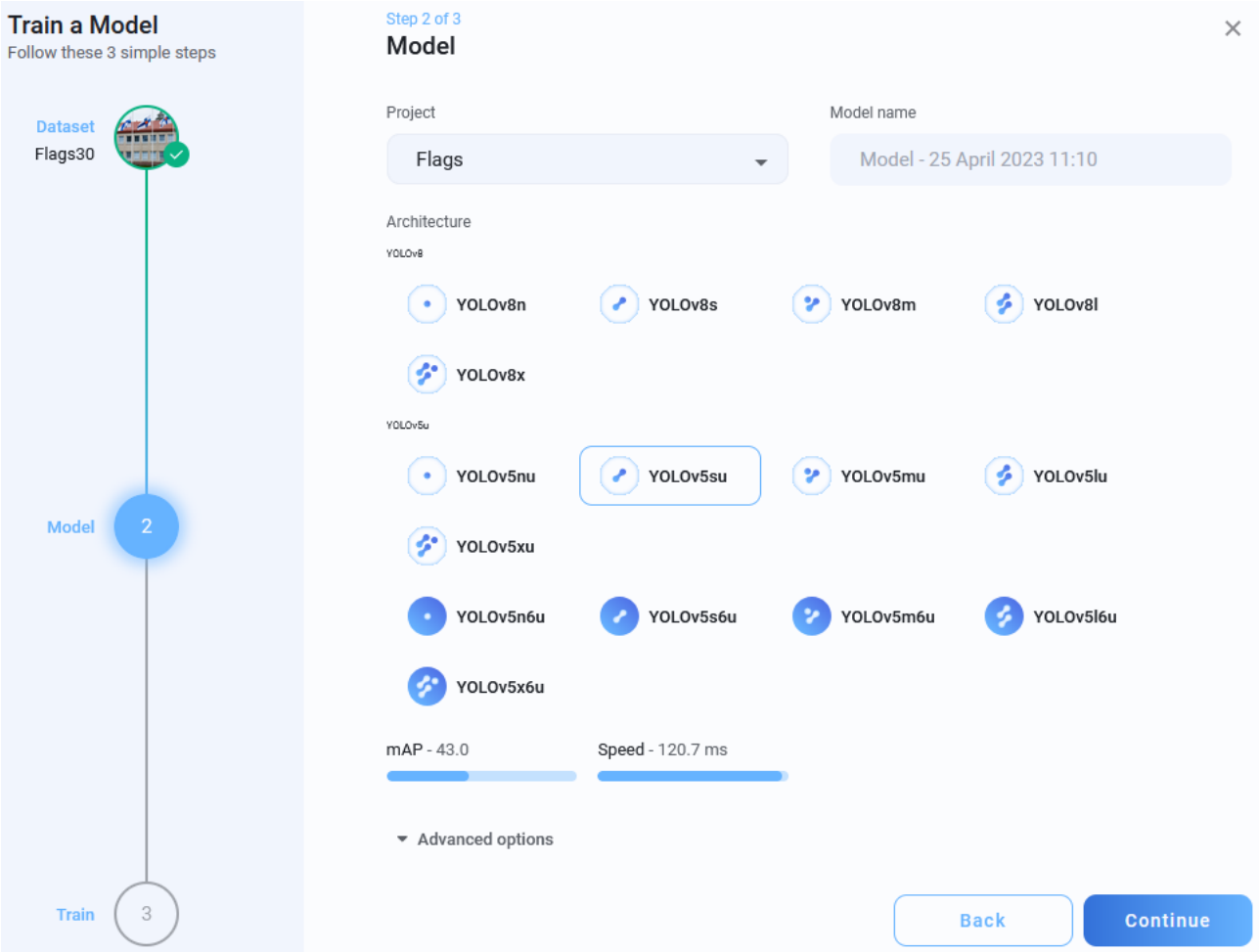
Mallin koulutus aloitetaan Model-sivulta ja valitaan Create Model. Avautuvasta ikkunasta valitaan juuri ladattu datasetti. Alusta näyttää oikein datasetin luokat sekä tunnistaa oikein koulutusdatan ja validointidatan jaon (Kuva 21).

Kuva 21 Mallin koulutus, vaihe 1



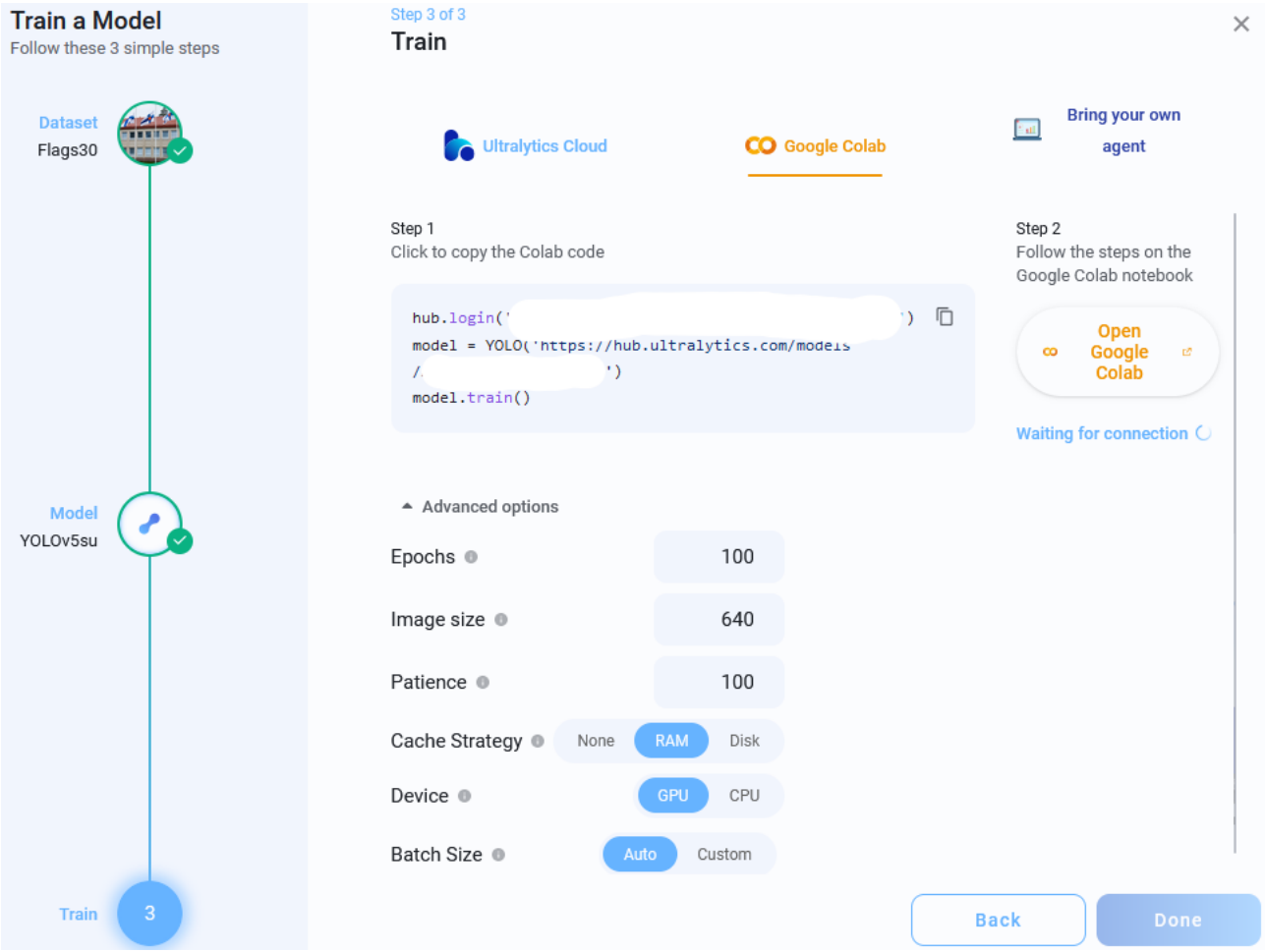
Toisessa vaiheessa voidaan lisätä malli tiettyyn projektiin ja antaa mallille uusi nimi. Lisäksi valitaan YOLO-versio, jota käytetään koulutuksessa (Kuva 22).

Kuva 22 Mallin koulutus, vaihe 2



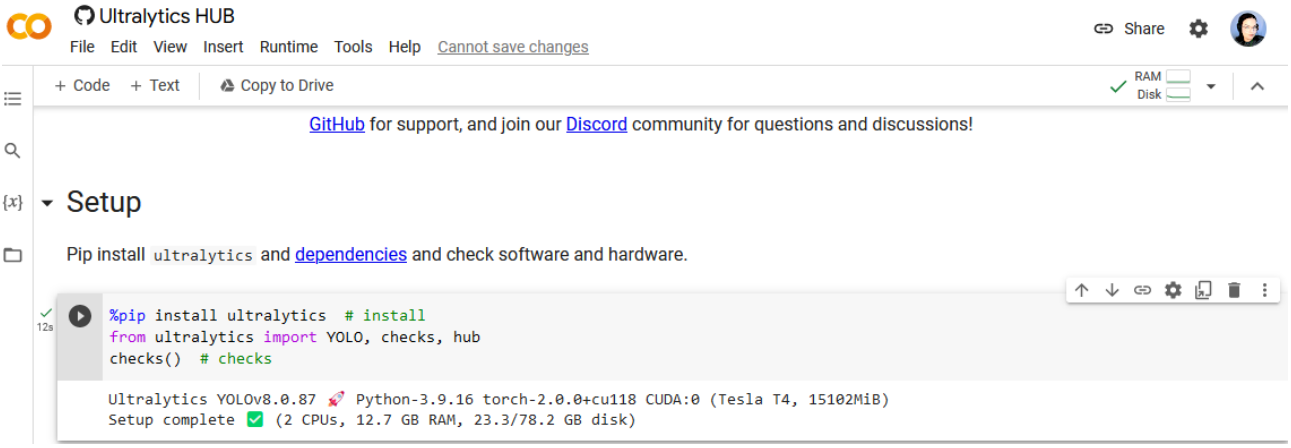
Seuraavassa vaiheessa kopioidaan ensin Ultralytics Hubin koodi Google Colabia varten. Sen jälkeen avataan Google Colab, jossa jatketaan koulutuksen seuraaviin vaiheisiin (Kuva 23).

Kuva 23 Mallin koulutus, vaihe 3



Google Colab -sivulla luodaan yhteys Googlen servereihin. Aluksi tehdään tarvittavat asennukset ohjeistuksen mukaisesti (Kuva 24).

Kuva 24 Google Colab -asennukset



Seuraavaan kenttään liitetään aikaisemmin Ultralytics Hub:in luoma API-avain ja mallin URL-osoite (Kuva 25). Tämän jälkeen koulutus voidaan aloittaa.

Kuva 25 API-avaimen lisäys ja mallin URL

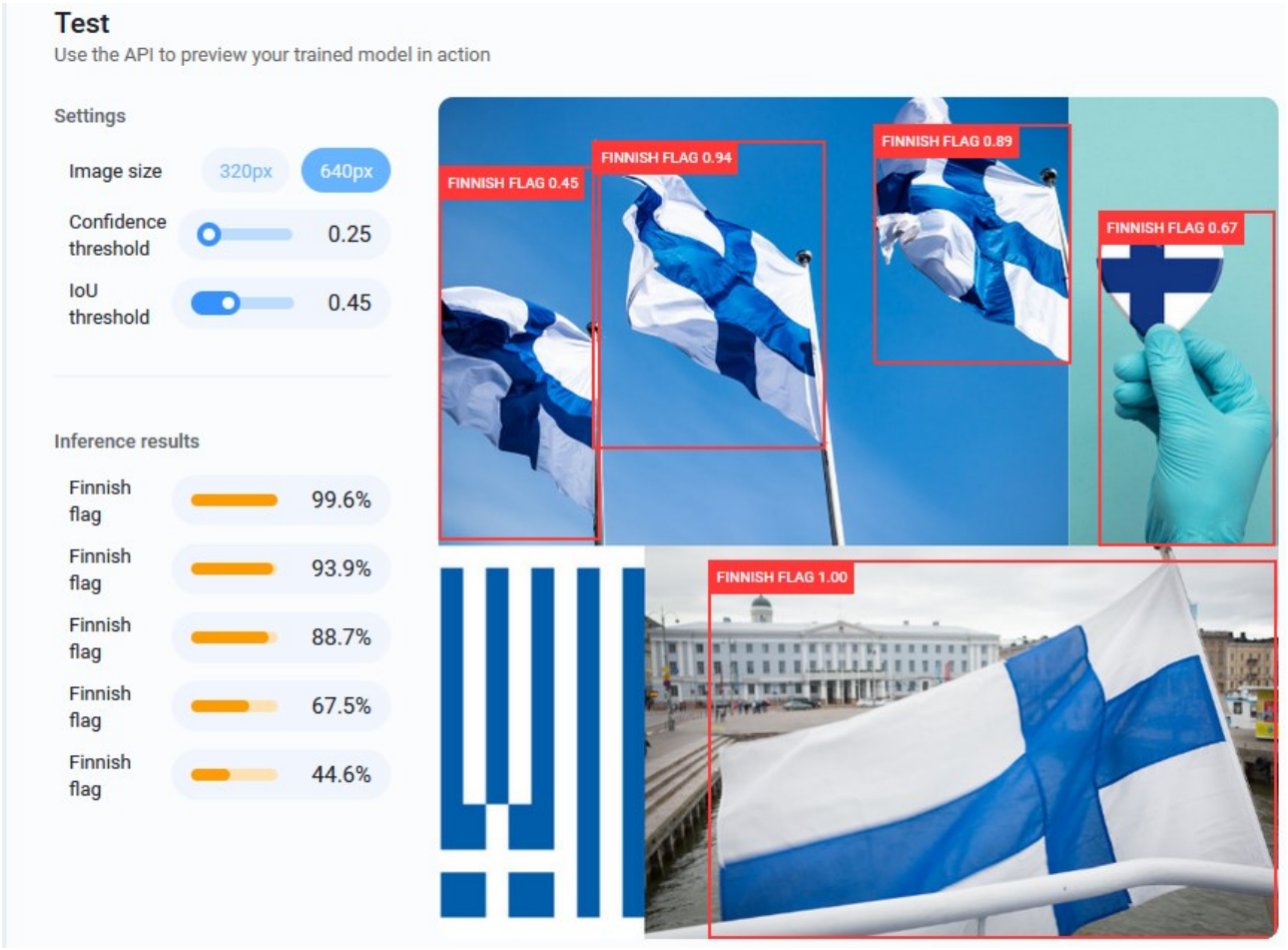
Koulutuksen kulkua voi seurata reaaliaikaisesti niin Google Colab -sivulla sekä Ultralytics Hub:ssa. Molemmat luovat mallin valmistumisen jälkeen yhteenvedon koulutuksesta ja mallista. 30:n kuvan datasetillä koulutus Google Colab:ia käyttäen kesti 2,1 minuuttia. Tarkasteltaessa yhteenvedon mAP50-95-saraketta, malli näyttää suoriutuvan hyvin. Valmiin mallin tiedostokoko on 18,5 MB (Kuva 26).

Kuva 26 Google Colab -tulokset 30 referenssikuvalla

```
100 epochs completed in 0.035 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 18.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 18.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.87 🚀 Python-3.9.16 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9111923 parameters, 0 gradients, 23.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00, 11.83it/s]
                   all          6         11          1      0.908      0.932      0.862
Speed: 0.2ms preprocess, 5.4ms inference, 0.0ms loss, 1.3ms postprocess per image
Results saved to runs/detect/train
Ultralytics HUB: Syncing final model...
100%|██████████| 17.7M/17.7M [00:01<00:00, 12.4MB/s]
Ultralytics HUB: Done ✅
```

Mallin toimivuutta voidaan testata Ultralytics Hub:ssa. Testaukseen käytetään erikseen luotua kollaasia kuvista, joissa on täysin uusia kuvia Suomen lipusta, sekä Kreikan lippu hämäystä luomaan. Malli tunnistaa kuvasta Suomen liput korkealla varmuudella, muttei tunnista virheellisesti Kreikan lippua Suomen lipuksi (Kuva 27).

Kuva 27 Google Colab -tulokset 30 referenssikuvalla



Seuraavaksi käydään mallin koulutus uudestaan läpi käyttämällä suurempaa 300 referenssikuvan datasettiä, jossa on alkuperäisen 30 kuvan lisäksi augmentoidut kuvat. Kaikki työvaiheet ovat samat kuin ensimmäisessä koulutuksessa. Datasetti jaetaan myös suhteessa 80/20 eli koulutusdataan kuuluu 240 referenssikuvaa ja validointidataan 60 kuvaa. Koulutukseen käytetään myös samaa YOLOv5-versiota.

300:n kuvan datasetillä koulutus Google Colabia käyttäen kesti 9,24 minuuttia. mAP50-95-arvon avulla voidaan päätellä, että koulutus loi selkeästi tarkemman mallin verrattuna pienemmän datasetin malliin (Kuva 28).

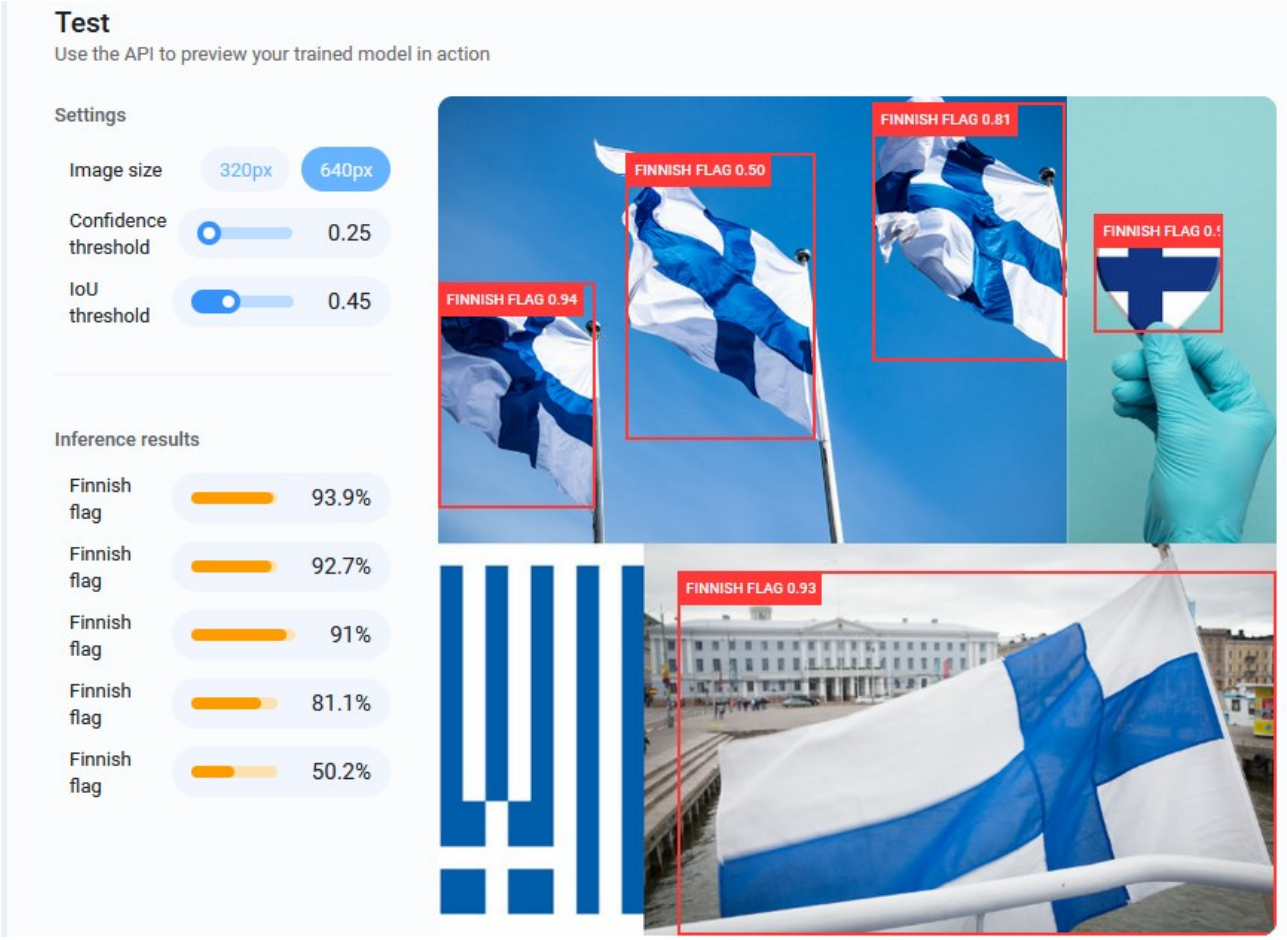
Kuva 28 Google Colab -tulokset 300 referenssikuvalla

```
100 epochs completed in 0.154 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 18.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 18.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.87 🚀 Python-3.9.16 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9111923 parameters, 0 gradients, 23.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  1.06it/s]
                   all         60         93          1      0.968      0.987      0.934
Speed: 0.2ms preprocess, 4.8ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to runs/detect/train
Ultralytics HUB: Syncing final model...
100%|██████████| 17.6M/17.7M [00:01<00:00, 16.0MB/s]
Ultralytics HUB: Done ✅
```

Mallia testataan samalla kuvakollaasilla, jota käytettiin pienemmän datasetin testauksessa. Malli näyttää suoriutuvan hyvin kaikkien tunnistettavien kohteiden kohdalla. Aiemmin tarkasti tunnistettujen arvot ovat kuitenkin hieman pudonneet, yhden lipun kohdalla pudonneet lähes puoleen aikaisemmasta. Aiemmin heikoimmin tunnistetut kohteet ovat toisaalta saaneet paremmat arvot suuremmalla datasetillä. Lisäksi malli osasi paikantaa sydämen muotoisen lipun paremmin kuin pienellä datasetillä luodun mallin kanssa. Tälläkään kertaa malli ei tunnistanut virheellisesti Kreikan lippua Suomen lipuksi (

Kuva 29).

Kuva 29 300:n referenssikuvan mallin testaus Ultralytics Hub:ssa



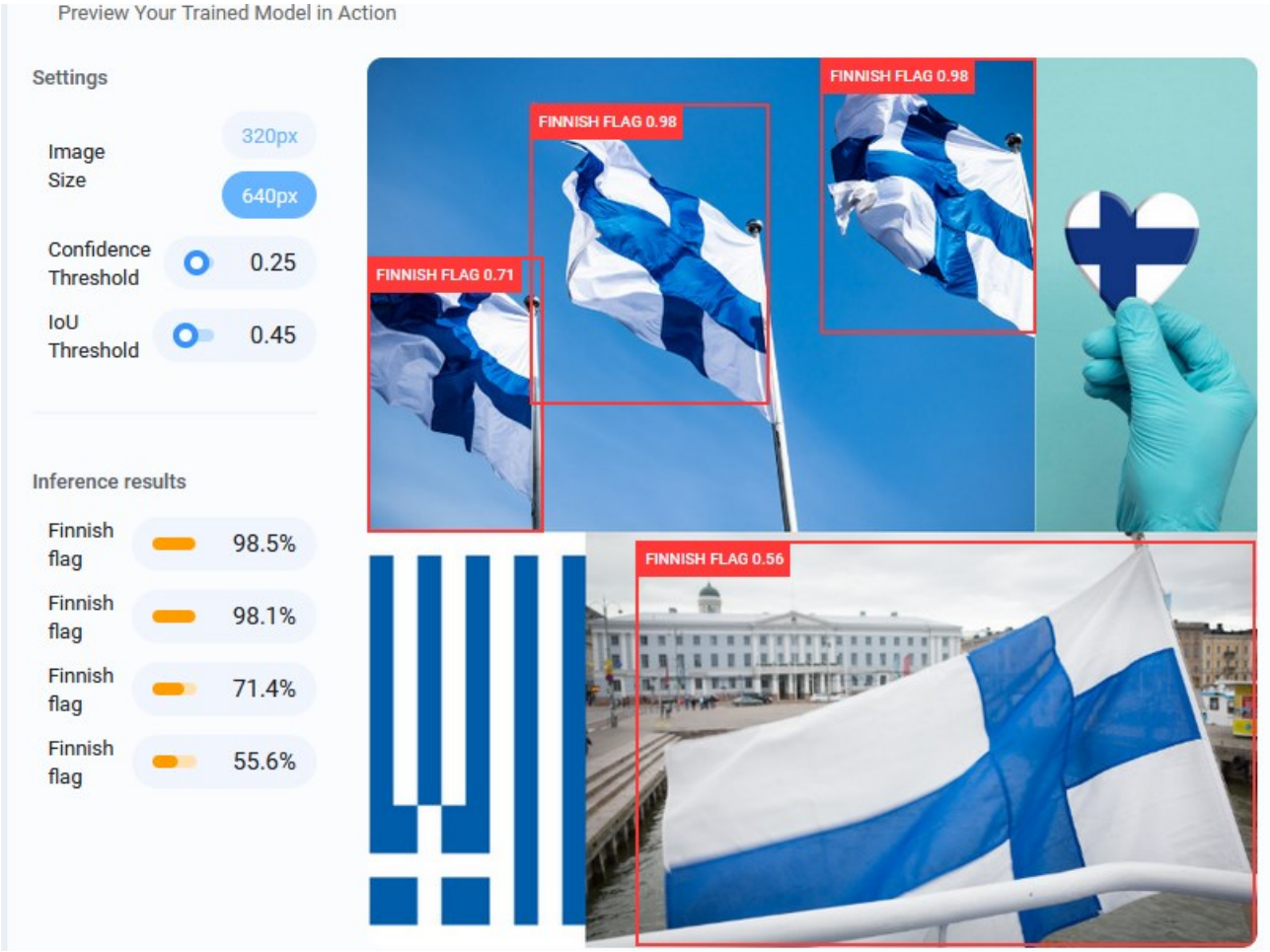
Vertailun vuoksi ajetaan Ultralytics Hubin ja Google Colabin kautta myös BadFlags-datasetit, jotka ovat tarkoituksella annotoitu huolimattomasti ja ohjeistuksista välittämättä. Koulutusaika on lähes sama kuin 30:n kuvan optimoidun datasetin kohdalla, mutta tarkkuus on tällä kertaa huomattavasti heikompi. mAP50-95-arvo jäi alle 0,5 (Kuva 30).

Kuva 30 Heikon, 30:n kuvan datasetin koulutuksen yhteenveto

```
100 epochs completed in 0.033 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 18.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 18.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.94 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9111923 parameters, 0 gradients, 23.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00, 11.52it/s]
                   all          6         12      0.904      0.786      0.943      0.457
Speed: 0.2ms preprocess, 5.1ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/train
Ultralytics HUB: Syncing final model...
100%|██████████| 17.6M/17.7M [00:00<00:00, 19.0MB/s]
Ultralytics HUB: Done ✅
```

Heikko mallin matala tarkkuus ilmenee, kun mallia testataan. Ensimmäistä kertaa malli ei tunnista sydämen muotoista lippua. Muiden arvojen kohdalla on myös eroja, joista on kuitenkin vaikea vetää minkäänlaisia johtopäätöksiä. Tarkkuudet vaihtelevat huomattavasti optimoidun datasetin mallin arvoihin verrattuna. Kuva 31 osoittaa, että tulokset ovat osin parempia ja osin huonompia.

Kuva 31 Heikon, 30:n kuvan datasetillä luodun mallin testaus



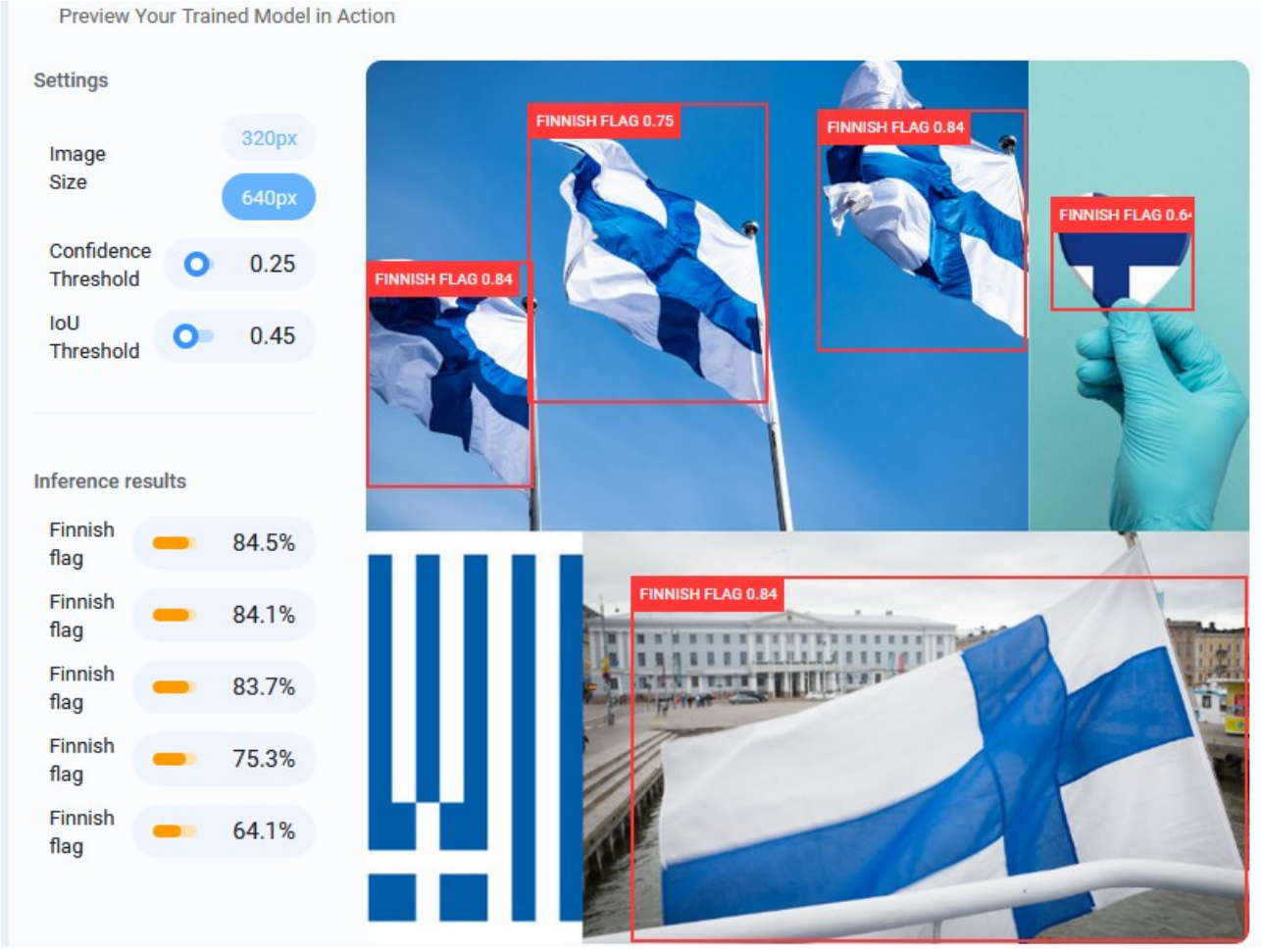
Koulutetaan lopuksi vielä uusi malli käyttäen suurempaa, heikkoa datasettiä. Tässä BadFlags-aineistossa on 300 kuvaa, jotka ovat huolimattomasti ja osin väärin annotoituja. Mallin tarkkuus ei juurikaan parantunut verrattuna pienemmän datasetin koulutettuun malliin. mAP50-95-arvo oli hieman yli 0,5 (Kuva 32).

Kuva 32 Heikon, 300:n kuvan datasetin koulutuksen yhteenveto

```
100 epochs completed in 0.149 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 18.5MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 18.5MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.94 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv5s summary (fused): 193 layers, 9111923 parameters, 0 gradients, 23.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  1.16it/s]
                   all         60         87      0.929      0.907      0.929      0.541
Speed: 0.2ms preprocess, 5.1ms inference, 0.0ms loss, 1.9ms postprocess per image
Results saved to runs/detect/train2
Ultralytics HUB: Syncing final model...
100%|██████████| 17.6M/17.7M [00:00<00:00, 34.1MB/s]
Ultralytics HUB: Done ✅
```

Tätä mallia testatessa näyttää, ettei eroa optimoituun malliin juurikaan ole. Keskimääräinen tarkkuus on kuitenkin kohteiden tunnistamisessa alempi. Lisäksi tarkasteltaessa rajauslaatikoita voidaan todeta niiden olevan selkeästi epätarkempia. Kohteiden rajauslaatikon ulkopuolelle jää osia tunnistettavasta objektista. Lisäksi rajauslaatikkoon on otettu mukaan paljon ympäröivää kuvaa, joka ei ole osa tunnistettavaa kohdetta (Kuva 33).

Kuva 33 Heikon, 300:n kuvan datasetillä luodun mallin testaus



8.4.2 Paikallinen tietokone ja CPU

YOLOv5-mallin kouluttaminen voidaan suorittaa paikallisella tietokoneella. Varsinainen koulutus suoritetaan Visual Studio Code -ohjelman avulla. Tässä työssä käytettävällä koneella on asennettuna Python-versio 3.9.13. Koulutuksen aloittamista varten kloonataan YOLOv5-kuvauskanta Ultralyticsin github-sivuilta (Komento 5).

Komento 5 YOLOv5-kuvauskannan kloonaus paikalliselle koneelle

```
C:\Users\tebia>git clone https://github.com/ultralytics/yolov5
Cloning into 'yolov5'...
remote: Enumerating objects: 15598, done.
remote: Counting objects: 100% (205/205), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 15598 (delta 97), reused 119 (delta 56), pack-reused 15393
Receiving objects: 100% (15598/15598), 14.58 MiB | 20.09 MiB/s, done.
Resolving deltas: 100% (10626/10626), done.
```

Lisäksi asennetaan requirements.txt-tiedosto python-ympäristössä. Tätä varten siirrytään ensin YOLOv5-kuvauskannan juureen. (Komento 6).

Komento 6 Requirements-tiedoston asennus paikalliselle koneelle

```
C:\Users\tebia> cd yolov5

C:\Users\tebia\yolov5>pip install -r requirements.txt
```

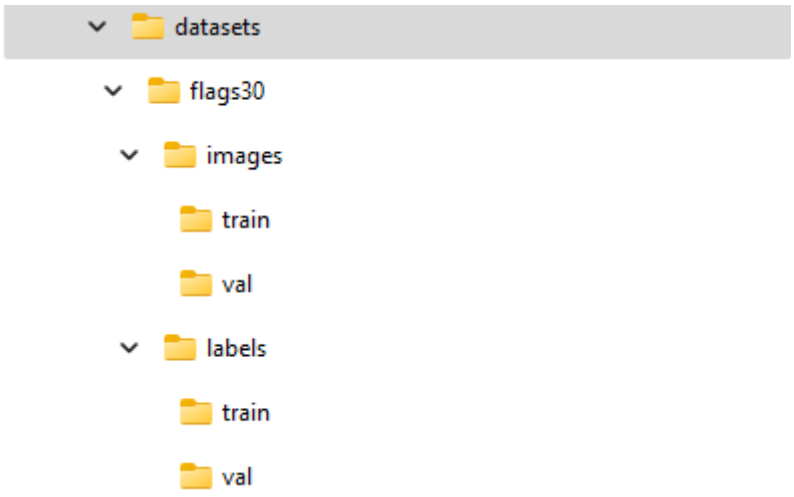
Koulutukseen käytetään samoja datasettejä kuin alaluvussa 8.4.1. Kloonatussa YOLOv5-kuvauskannassa on coco128.yaml-tiedosto, jota muokataan datasetteihin sopivaksi (Ohjelmakoodi 2).

Ohjelmakoodi 2 Coco128.yaml-tiedoston sisältö paikallisella koneella

```yaml
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
# COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017) by Ultralytics
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128  ← downloads here (7 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/flags30  # dataset root dir
train: images/train  # train images (relative to 'path') 24 images
val: images/val  # val images (relative to 'path') 6 images
test:  # test images (optional)

nc: 1
names: ['Finnish flag']


# Download script/URL (optional)
download: https://ultralytics.com/assets/coco128.zip
```

Lisäksi luodaan datasets-kansio, jonne siirretään flags30-datasetti images- ja labels-alakansioineen. Datasets-kansio sijoitetaan sama juurikansion alle kuin yolov5-hakemisto. Flags30-kansiosta poistetaan aiemmin luotu yaml-tiedosto, silla vastaavat tiedot ovat nyt coco128.yaml-tiedostossa (Kuva 34).

Kuva 34 Datasets-kansion rakenne ja sisältö paikallisella koneella



Koulutus voidaan nyt aloittaa. Se suoritetaan samoilla arvoilla kuin alaluvussa 8.4.1 kuvatussa koulutuksessa (Komento 7).

Komento 7 Koulutuksen aloittaminen paikallisella koneella

```
PS C:\Users\tebia\yolov5> python train.py --img 640 --epochs 100 --data coco128.yaml --weights yolov5s.pt
```

30:n kuvan datasetillä koulutus kesti paikallisella koneella 25,44 minuuttia. Yhteenvedon mukaan malli näyttää suoriutuvan kohtuullisesti. Valmiin mallin tiedostokoko on 14,4 MB (Kuva 35).

Kuva 35 30:n kuvan koulutuksen yhteenveto paikallisella koneella

```
100 epochs completed in 0.424 hours.
Optimizer stripped from runs\train\exp\weights\last.pt, 14.4MB
Optimizer stripped from runs\train\exp\weights\best.pt, 14.4MB

Validating runs\train\exp\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 1/1 [00:01<00:00,  1.14s/it]
                 all          6         11      0.886      0.818      0.891      0.602
Results saved to runs\train\exp
```

Mallia testataan samalla kuvakollaasilla kuin aikaisemmissa testauksissa. Se lisätään yolov5-hakemiston data/images-kansioon. Tässä testikuvassa on useampi kuva Suomen lipusta sekä yksi Kreikan lippu. Testaus suoritetaan suoraan komennolla Visual Studio Coden terminaalissa (Komento 8).

Komento 8 Mallin testauksen suorittaminen paikallisella koneella

```
PS C:\Users\tebia\yolov5> python detect.py --weights runs\train\exp\weights\best.pt --img 640 --conf 0.25 --source data/images
```

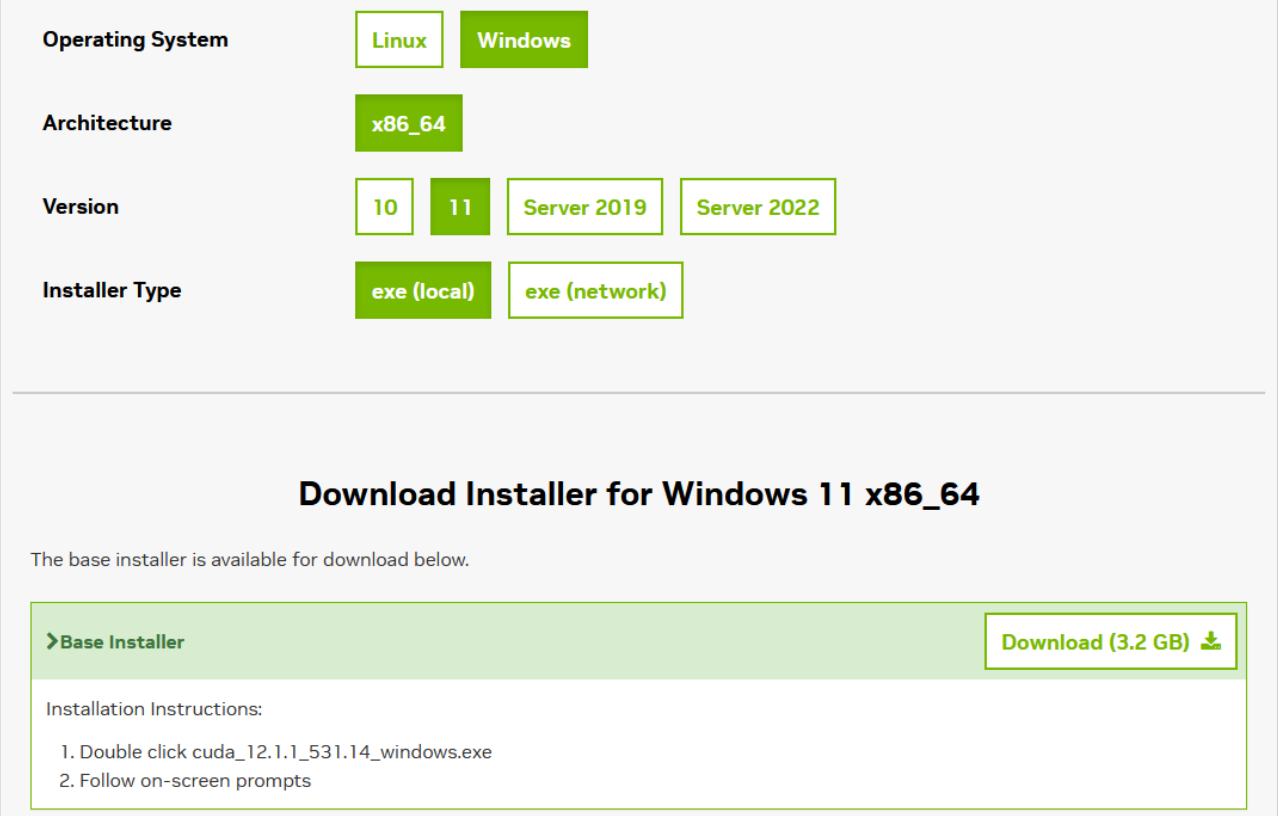
Malli suoriutuu selkeästi heikommin kohteiden tunnistamisessa sekä paikantamisessa. Malli osasi paikantaa sydämen muotoisen kuvasta, mutta virheellisesti tunnisti erikseen myös sydäntä pitelevän käden Suomen lipuksi. Lisäksi malli paikansi ja tunnisti lipun yhdestä varsinaisesta kohteesta kolme kertaa. Tälläkään kertaa malli ei mennyt halpaan Kreikan lipun kanssa (Kuva 36).

Kuva 36 Mallin testauksen tulokset paikallisella koneella

8.4.3 Paikallinen tietokone ja GPU

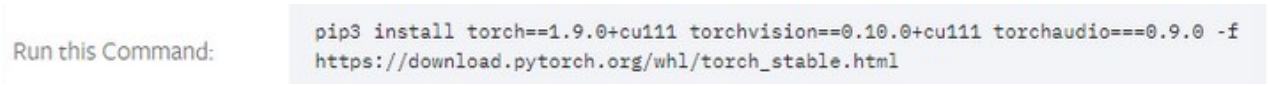
Ensimmäisen mallin luominen paikallisella koneella oli hidasta, eikä tulos ollut kovin hyvä. Suuremman datasetin käyttäminen kouluttamisessa veisi tunteja. Edellistä koulutusta tarkasteltaessa huomataan, ettei siinä käytetty koneen grafiikkaprosessoria apuna, vaan YOLO toimii ainoastaan CPU:n varassa. Tätä varten lähdetään asentamaan Kuva 37 mukaisesti NVIDIA Cuda Toolkittiä paikalliselle koneelle.

Kuva 37 NVIDIA Cuda Toolkit lataus paikalliselle koneelle



Ladattu tiedosto ajetaan ja asennetaan paikalliselle koneelle. Tämän jälkeen ladataan ja asennetaan PyTorch. Komento 9 ajetaan paikallisella koneella samassa sijainnissa, jossa Python-komennot ovat tallennettuina.

Komento 9 PyTorchin lataus ja asennus



Komentokehotteen avulla tarkastetaan vielä, että edelliset asennukset ovat onnistuneet. Syötteestä nähdään, että asennukset ovat onnistuneet ja NVIDIA Geforce RTX 3060 -näytönohjain toimii indeksillä 0 (Kuva 38).

Kuva 38 Cuda-asennusten tarkastus

```
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
>>> torch.cuda.get_device_properties(0).name
'NVIDIA GeForce RTX 3060'
>>>
```

Uuden mallin luomisen Komento 10 poikkeaa aikaisemmista koulutuksien komennoista. Aikaisemmin käytetyn komennon perään lisätään –device ja määritellään käytetyn laitteen indeksi.

Komento 10 Mallin koulutuskomento Cudan avulla

```
PS C:\Users\tebia\yolov5> python train.py --img 640 --epochs 100 --data coco128.yaml --weights yolov5s.pt --device 0
```

Mallin koulutus on huomattavasti nopeampi, ja suoriutuu jopa Google Colab -alustaa nopeammin kestäen vain 1,32 minuuttia. Mallin koulutus näyttää suoriutuvan edellistä mallia paremmin, joskin hieman Google Colabia heikommin (Kuva 39).

Kuva 39 30:n kuvan koulutus Cudan avulla

```
100 epochs completed in 0.022 hours.
Optimizer stripped from runs\train\exp2\weights\last.pt, 14.4MB
Optimizer stripped from runs\train\exp2\weights\best.pt, 14.4MB

Validating runs\train\exp2\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 1/1 [00:00<00:00, 11.66it/s]
                 all          6         11      0.972      0.909      0.906      0.584
Results saved to runs\train\exp2
```

Malli suoriutuu melko hyvin kohteiden tunnistamisessa sekä paikantamisessa. Malli paikansi sydämen muotoisen lipun kuvasta, mutta tunnisti jälleen väärin myös sydäntä pitelevän käden Suomen lipuksi. Mallin tarkkuus paikallisella koneella luotuna on kaiken kaikkiaan parempi, kun käytössä on ollut NVIDIA Cuda Toolkit. Tällä kertaa malli kuitenkin meni ensimmäistä kertaa halpaan Kreikan lipun kanssa (Kuva 40).

Kuva 40 Cuda-mallin testauksen tulokset paikallisella koneella

Seuraavaksi mallin koulutus käydään läpi käyttäen suurempaa 300:n kuvan datasettiä. Kaikki aikaisemmat työvaiheet ja käytetyt arvot pysyvät samoina datasetin kokoa lukuun ottamatta. Koulutus valmistuu nopeammin kuin mikään edellisistä koulutuksista kestäen vain 5,76 minuuttia. Malli on myös kaikkia aiemmin luotuja malleja tarkempi (Kuva 41).

Kuva 41 300:n kuva koulutus Cudan avulla

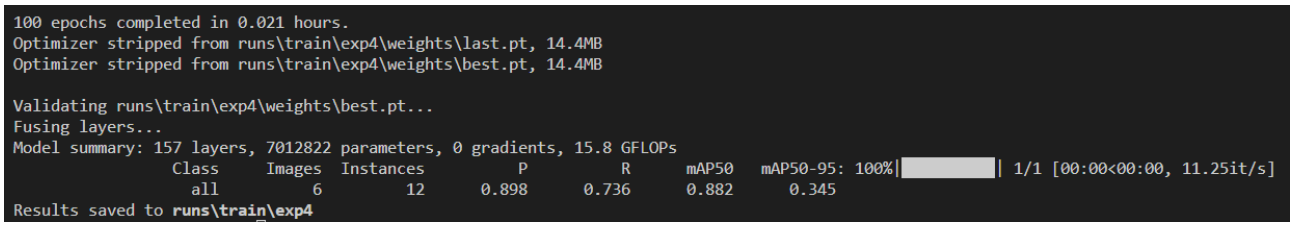
```
100 epochs completed in 0.096 hours.
Optimizer stripped from runs\train\exp3\weights\last.pt, 14.4MB
Optimizer stripped from runs\train\exp3\weights\best.pt, 14.4MB

Validating runs\train\exp3\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 2/2 [00:00<00:00,  2.15it/s]
                   all         60         93          1      0.975      0.987       0.91
Results saved to runs\train\exp3
```

Malli tunnistaa objektit kuvasta oikein. Lisäksi tunnistamisen tarkkuus on korkea (Kuva 42).

Kuva 42 300:n kuvan datasetillä luodun mallin testitulokset

Suoritetaan koulutukset vielä BadFlags-dataseteillä käyttäen paikallista konetta ja Cuda toolkittiä. Ensin koulutetaan malli pienemmällä 30:n kuva datasetillä. Koulutuksen yhteenvedossa Kuva 43 nähdään, että luodun mallin tarkkuus on hyvin heikko.

Kuva 43 Koulutuksen yhteenveto heikolla 30:n kuvan datasetillä

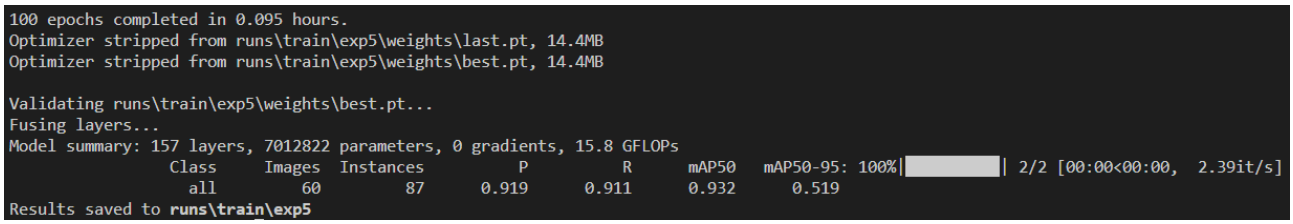


Mallia testatessa voidaan vahvistaa edelliset päätelmät. Malli on epätarkka, tunnistaa objekteja väärin ja heikolla varmuudella (Kuva 44).

Kuva 44 Heikon, 30:n kuvan datasetin mallin testaus paikallisella koneella

Suuremmalla, mutta heikosti annotoidulla datasetillä koulutuksen tulokset eivät ole juurikaan edellistä koulutusta paremmat. Mallin tarkkuus on yhteenvedon mukaan lähes puolet heikompi kuin optimoidulla datasetillä luodussa mallissa (Kuva 45).

Kuva 45 Heikon, 300:n kuvan datasetillä luodun mallin koulutuksen yhteenveto



Mallia testatessa nähdään, että se on vain hieman pienemmän datasetin mallia tarkempi. Lisäksi luotu malli tunnistaa kohteita yhä virheellisesti (Kuva 46).

Kuva 46 Heikon, 300:n kuvan datasetin mallin testaus paikallisella koneella

9 Tulokset

Käytännön osiossa tutkittiin Suomen lippujen tunnistamista ja paikantamista valokuvista. Tätä varten luotiin yksi yhteinen 30:n kuvan referenssikuvakirjasto, jota augmentoimalla luotiin myös suurempi, 300:n kuvan kuva-aineisto. Kaikki kuvat annotointiin kahdella tavalla. Ensimmäisessä annotoinnissa kuvat annotoitiin tarkasti, johdonmukaisesti ja määritellyn ohjeistuksen mukaisesti. Toisessa annotointimenetelmässä annotointi suoritettiin epätarkasti, osa kohteista jätettiin merkitsemättä ja osin myös virheellisesti. Yhteensä luotiin 9 erilaista kuvantunnistusmallia näiden aineistojen avulla.

Kuvien määrällä oli tärkeä merkitys monessa vaiheessa. Ensimmäiset 30 referenssikuvaa loivat hyvän pohjan suuremman datasetin luomiselle. Niitä augmentoimalla saatiin helposti suurempi aineisto mallin koulutusta varten. 30 kuvalla luodut mallit olivat selkeästi epätarkempia suurempien datasettien avulla luotuihin malleihin verrattuna. 300:n kuvan avulla luodut mallit tunnistivat kohteet hyvin testikuvista ja tarkkuus oli korkea. Tarkkuutta on todennäköisesti mahdollista vielä parantaa, sillä korkein mAP50-95-arvo oli 0,934. Datasetin koon kasvattaminen 500:n kuvaan saattaisi tuoda malliin hieman lisätarkkuutta. Lisäksi tässä työssä tunnistettava kohde oli vain yksi selkeä ilmentymä. Jos kohteena olisi ollut yleisempi tunnistettava luokka, esimerkiksi eurooppalainen lippu, referenssikuvia tarvittaisiin huomattavasti enemmän.

Alkuperäisen referenssikuva-aineiston kuvat olivat laadukkaita. Tässä sillä viitataan aineiston monipuolisuuteen ja kohteiden esittäminen vaihtelevissa ympäristöissä. Aineistossa ei kuitenkaan otettu ollenkaan huomioon ollenkaan negatiivisia kohteiden ilmentymiä. Mallia ei koulutettu missään datasetissä oppimaan, mitkä kohteet ovat vääriä. Suomen lipun näköisillä kohteilla, jotka eivät kuitenkaan ole Suomen lippuja, olisi voitu lisätä mallien tarkkuutta ja siten välttää väärien kohteiden tunnistaminen.

Augmentoinnin avulla lisättiin alkuperäiseen referenssikuvakirjastoon määrää ja laatua. Se helpotti työn suorittamista siinä suhteessa, ettei referenssikuvia pitänyt olla suuria määriä. Aineiston laadukkuutta voisi mahdollisesti kuitenkin parantaa lisäämällä referenssikuvien määrä ja laskemalla augmentointien määrää. Autenttisuus ja monipuolisuus datasetissä on selkeästi sen tärkein ominaisuus. Tästä kertoo erityisesti se, ettei pienellä 30:n kuvan optimoidulla datasetillä mallin suorituskyky ollut täysin epäkelpoinen.

Kuvien määrän lisäksi annotoinnilla oli suurin vaikutus mallin toimivuuteen. Tarkkaan ja johdonmukaisesti annotoiduilla dataseteillä koulutetut mallit toimivat huomattavasti paremmin kuin heikkojen BadFlags-datasettien avulla luodut mallit.

Taulukko 2 kerättiin kaikki tiedot luotujen mallien yhteenvedoista. Mallien kokonaisvaltainen toimivuus näkyy sarakkeessa mAP50-95. Mitä korkeampi tämä arvon on, sitä paremmin luotu malli suoriutui. 300:n kuvan optimoidut datasetit suoriutuivat parhaiten. Niiden jälkeen tulevat pienemmät 30:n kuvan optimoidut datasetit. Taulukon pohjalla näkyvät heikosti suoriutuvat BadFlags-datasetit.

Taulukko 2 Luotujen mallien yhteenveto

Alusta	**Datasetti**	**Kuvien määrä**	**P (tarkkuus)**	**R (herkkyys)**	**mAP50**	**mAP50 -95**	**Koulutuksen kesto (h)**
Google Colab	Flags300	300	1	0.968	0.987	**0,934**	0.154
Paikallinen kone (GPU)	Flags300	300	1	0.975	0.987	**0,91**	0.096
Google Colab	Flags30	30	1	0.908	0.932	**0,862**	0.035
Paikallinen kone (CPU)	Flags30	30	0.886	0.818	0.891	**0,602**	0.424
Paikallinen kone (GPU)	Flags30	30	0.972	0.909	0.906	**0,584**	0.022
Google Colab	BadFlags300	300	0.929	0.907	0.929	**0,541**	0.149
Paikallinen kone (GPU)	BadFlags300	300	0.919	0.911	0.932	**0,519**	0.095
Google Colab	BadFlags30	30	0.904	0.786	0.943	**0,457**	0.033
Paikallinen kone (GPU)	BadFlags30	30	0.898	0.736	0.882	**0,345**	0.021

10 Johtopäätökset ja pohdinta

Toimivan tietokonenäkömallin kouluttamisessa pitää ottaa monta asiaa huomioon. Aineiston käsittelyllä on suuri merkitys lopullisen mallin toimivuuteen. Työskentelyn tarkkuus, suunnitelmallisuus ja johdonmukaisuus ovat erityisen tärkeitä.

Laadukkaat referenssikuvat luovat hyvän pohjan mallin koulutusta varten. Augmentoinnin avulla voidaan laajentaa datasettiä ja luotavan mallin tarkkuutta. Tarvittavan datasetin laajuus riippuu vahvasti kunkin objektintunnistusprojektin vaatimuksista. Tunnistettavan objektin ollessa selkeä ja visuaalisesti yksiselitteinen, datasetiksi saattaa riittää suppeampi, muutaman sadan kuvan datasetti. Toisaalta tunnistettavan objektin ollessa vaikeasti rajattavissa tai sen sisältäessä useita erilaisia alaluokkia tai määritelmiä, datasetin on otettava kaikki nämä eroavuudet hyvin huomioon. Tällöin tarvittavan aineiston koko saattaa kasvaa sadoista kuvista tuhansiin kuviin.

Rakennettaessa datasettiä objektintunnistusmallia varten on tärkeää ymmärtää tunnistettava kohde ja mitä piirteitä siltä erityisesti vaaditaan. Kun tunnistettavan objektin vaatimukset ymmärretään, osataan referenssikuville valita oikeat augmentointimenetelmät. Ilman tätä ymmärrystä ja käyttämällä virheellisiä augmentointeja luotu objektintunnistusmalli saattaa antaa virheellisiä tai epätarkkoja tuloksia.

Datasetin annotointi on hyvin tärkeä vaihe objektintunnistusmallia luodessa. Juuri oikeanlaisen työkalun löytäminen kullekin projektille on tärkeää, jotta työskentely on yhdenmukaista ja tehokasta. Annotaattoreilla tulee olla selkeä ohjeistus siitä, miten rajauslaatikot luodaan. Epämääräiset tai muuten puutteelliset ohjeistukset saattavat heikentää annotoinnin laatua ja siten myös luotavan mallin laatua.

YOLOv5-mallin luomiseen on olemassa useita vaihtoehtoja. Se, mikä menetelmä sopii kullekin projektille, riippuu jokaisen projektin saatavilla olevista resursseista. Nämä resurssit on tärkeää olla selvillä ennen objektintunnistusmallin koulutuksen aloittamista. Kuten monessa muusakin tilanteessa, hyvin suunniteltu on puoliksi tehty.

Opinnäytetyön avulla luodaan ohjeistukset datasettien käsittelyyn Calevala Interactive Oy:n objektin tunnistusta käyttävää palvelua varten. Lisäksi opinnäytetyö mahdollistaa ulkopuolisten palvelujen ja työkalujen kriittisen arvioinnin sovellusta suunnitellessa. Saadut tulokset

nopeuttavat, yhdenmukaistavat sekä selkeyttävät luotavan palvelun ohjelmointiprosessia. Objektintunnistusmallin luomiseen tarvittavien vaiheiden ja prosessien ymmärtäminen auttaa minimoimaan ongelmat palvelun suunnittelussa sekä ohjelmoinnissa.

Varsinainen teoriapohja sekä toiminnallinen osuus tästä opinnäytetyöstä syntyivät nopeasti. Tämä mahdollisti aihealueen syvällisemmän tutkiskelun tulosten pohjalta. Calevala Interactive Oy:n palvelun luomisessa on täten päädytty monelta osin luomaan työkalut alusta alkaen itse, minimoiden ulkopuolisten palvelujen tarve ja samalla vähentäen projektin kustannuksia.

11 Yhteenveto

Tämä opinnäytetyön tarkoituksena oli datasetin optimointi sekä mallin luominen objektin tunnistusmallia varten. Vaikka aihealue oli laaja, sen pilkkominen pienempiin osiin helpotti aiheen tutkimista ja käsittelyä. Tutkimuskysymyksiin vastaaminen onnistui ja toimeksiantajayritys sai tarvitsemansa tiedot työn pohjalta.

Koko opinnäytetyön aihealue oli minulle kokonaisuudessa aivan uusi, joten olen oppinut kaikki tässä työssä käsitellyt asiat aivan alusta. Teoriapohjan lisäksi opin ymmärtämään objektintunnistusprosessin vaiheita sekä käsittämään, miten tärkeitä pienetkin yksityiskohdat tämänlaisessa projektissa ovat.

Työn tulokset ovat tärkeät toimeksiantajan objektintunnistamisprojektin kannalta. Saatujen tietojen avulla tiedetään, minkälaisia vaiheita objektin tunnistamisessa on, paljonko niihin tarvitaan aikaa ja lisäksi, miten ne tulee käytännössä toteuttaa. Työn tulosten avulla osataan optimoida luodut mallit sekä tunnistaa ongelmia tai puutteita kouluttamisessa. Opinnäytetyötä tullaan käyttämään apuna luodessa materiaaleja toimeksiantajayrityksen objektintunnistuspalveluprojektin eri tiimeille. Työn tulokset mahdollistavat lisäksi luotavan palvelun optimoimisen juuri sen tarpeita ja resursseja silmällä pitäen.

Lähteet

Ailisto, H., Myllymäki, P., Tarkoma, S., Kämäräinen, J.-K., Röning, J., Salakoski, T., Solin, A., Saariluoma, P., Mikkonen, T., van Gils, M., Väänänen, K., Puolamäki, K., Ylén, P., Roos, T., Leikas, J., Honkela, A., Kutila, M., Ruotsalainen, L., Ylikoski, P., & Linturi, R. (ei pvm.). *Tekoälyratkaisut tänään ja tulevaisuudessa*.

Baheti, P. (2023). *Train Test Validation Split: How To & Best Practices [2023]*. https://www.v7labs.com/blog/train-validation-test-set

Bonaccorso, G., Fandango, A., & Shanmugamani, R. (2018). *Python: Advanced Guide to Artificial Intelligence: Expert Machine Learning Systems and Intelligent Agents Using Python*. Packt Publishing, Limited. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=5626921

Brownlee, J. (2019, toukokuuta 21). A Gentle Introduction to Object Recognition With Deep Learning. *MachineLearningMastery.Com*. https://machinelearningmastery.com/object-recognition-with-deep-learning/

Cappi, C., Chapdelaine, C., Gardes, L., Jenn, E., Lefevre, B., Picard, S., & Soumarmon, T. (2021). *Dataset Definition Standard (DDS)*.

Chernytska, O. (2023, helmikuuta 11). *Complete Guide to Data Augmentation for Computer Vision*. Medium. https://towardsdatascience.com/complete-guide-to-data-augmentation-for-computer-vision-1abe4063ad07

Chutani, P. (2022, marraskuuta 29). *Challenges in Image Annotation*. Pratham Software™. https://www.thepsi.com/challenges-in-image-annotation/

Dang, T. (2022). *Guide to accuracy, precision, and recall*. Mage. https://www.mage.ai/blog/definitive-guide-to-accuracy-precision-recall-for-product-developers

Datagen. (ei pvm.). Image Annotation for Computer Vision: A Practical Guide. *Datagen*. Noudettu 30. maaliskuuta 2023, osoitteesta https://datagen.tech/guides/image-annotation/image-annotation/

Davies, D. (2022, marraskuuta 15). *YOLOv5 Object Detection on Windows (Step-By-Step Tutorial)*. W&B. https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4

Dilmegani, C. (2023, maaliskuuta 6). *Top Data Augmentation Techniques: Ultimate Guide for 2023*. https://research.aimultiple.com/data-augmentation-techniques/

Emmytheo 24/7. (2019, syyskuuta 23). Implementing Real-Time Object detection on a CPU using one of the smartest Convolutional Neural…. *Medium*. https://medium.com/@chidi.mgbara/implementing-real-time-object-detection-on-a-cpu-using-one-of-the-smartest-convolutional-neural-d31297804931

Exxact. (2022). *YOLOv5 PyTorch Tutorial | How to Use YOLOv5 in Pytorch*. https://www.exxactcorp.com/blog/Deep-Learning/YOLOv5-PyTorch-Tutorial

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Deep Learning. https://www.deeplearningbook.org/

Google. (ei pvm.). *Google Colab*. Noudettu 13. huhtikuuta 2023, osoitteesta https://research.google.com/colaboratory/faq.html

Hardesty, L. (2017, huhtikuuta 14). *Explained: Neural networks*. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Iakushechkin, D. (2023). *The best image labeling tools for Computer Vision*. Dida Machine Learning. https://dida.do/blog/the-best-labeling-tools-for-computer-vision

Jyväskylän Yliopisto. (ei pvm.). *tekoälyn_perusteita_ja_sovelluksia_kirja—TIM*. Noudettu 22. maaliskuuta 2023, osoitteesta https://tim.jyu.fi/view/kurssit/tie/tiep1000/tekoalyn-sovellukset/kirja#DKUvbnUuGytQ

Kanber, B. (2018). *Hands-On Machine Learning with JavaScript: Solve Complex Computational Web Problems Using Machine Learning*. Packt Publishing, Limited. http://ebookcentral.proquest.com/lib/hamk-ebooks/detail.action?docID=5405677

Karimi, G. (2021). *Introduction to YOLO Algorithm for Object Detection*. Engineering Education (EngEd) Program | Section. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

Kaur, J., & Singh, W. (2022). Tools, techniques, datasets and application areas for object detection in an image: A review. *Multimedia Tools and Applications*, *81*(27), 38297–38351. https://doi.org/10.1007/s11042-022-13153-y

Kelta, Z. (2022). *YOLO Object Detection Explained: A Beginner's Guide*. https://www.datacamp.com/blog/yolo-object-detection-explained

Krittapholchai, C. (2022, kesäkuuta 11). NO code YOLOv5 model with Ultralytics HUB. *Medium*. https://medium.com/@chanon.krittapholchai/no-code-yolov5-model-with-ultralytics-hub-c8f1323af2d9

LabelFlow. (ei pvm.-a). *Introduction to Workspaces*. Noudettu 5. huhtikuuta 2023, osoitteesta https://labelflow.gitbook.io/labelflow/workspaces/introduction-to-workspaces

LabelFlow. (ei pvm.-b). *Label Types*. Noudettu 5. huhtikuuta 2023, osoitteesta https://labelflow.gitbook.io/labelflow/labelling-interface/quick-start

LabelFlow. (ei pvm.-c). *Workspace Settings & Billing*. Noudettu 5. huhtikuuta 2023, osoitteesta https://labelflow.gitbook.io/labelflow/workspaces/introduction-to-workspaces

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2019). *Deep Learning for Generic Object Detection: A Survey* (arXiv:1809.02165). arXiv. http://arxiv.org/abs/1809.02165

Liubimov, N. (2020, tammikuuta 31). *Introducing Label Studio, a swiss army knife of data labeling*. Medium. https://towardsdatascience.com/introducing-label-studio-a-swiss-army-knife-of-data-labeling-140c1be92881

Mage. (2021, joulukuuta 10). *10 steps to build and optimize a ML model*. DEV Community. https://dev.to/mage_ai/10-steps-to-build-and-optimize-a-ml-model-4a3h

MinnaLearn. (ei pvm.-a). *Mitä on tekoäly ja koneoppiminen*. Mitä on tekoäly ja koneoppiminen. Noudettu 28. maaliskuuta 2023, osoitteesta https://www.minnalearn.com/fi/courses/ai-for-built-environment/tekoalyn-hyodyntamisen-edellytykset/mita-on-tekoaly-ja-koneoppiminen/

MinnaLearn. (ei pvm.-b). *Neuroverkkojen periaatteet—Elements of AI*. Noudettu 23. maaliskuuta 2023, osoitteesta https://course.elementsofai.com/fi/5/1

MinnaLearn, & Helsingin Yliopisto. (ei pvm.-a). *Edistyneet neuroverkkomenetelmät—Elements of AI*. Noudettu 23. maaliskuuta 2023, osoitteesta https://course.elementsofai.com/fi/5/3

MinnaLearn, & Helsingin Yliopisto. (ei pvm.-b). *Koneoppimisen lajit—Elements of AI*. Noudettu 23. maaliskuuta 2023, osoitteesta https://course.elementsofai.com/fi/4/1

MinnaLearn, & Helsingin Yliopisto. (ei pvm.-c). *Miten neuroverkkoja rakennetaan? - Elements of AI*. Noudettu 23. maaliskuuta 2023, osoitteesta https://course.elementsofai.com/fi/5/2

MinnaLearn, & Helsingin Yliopisto. (ei pvm.-d). *Tekoälyn filosofiaa—Elements of AI*. Noudettu 23. maaliskuuta 2023, osoitteesta https://course.elementsofai.com/fi/1/3

NVIDIA. (2013, heinäkuuta 2). *CUDA Toolkit—Free Tools and Training*. NVIDIA Developer. https://developer.nvidia.com/cuda-toolkit

Nykänen, O. (2021, toukokuuta 24). *Weka-aloitusohje koneoppimisen kokeiluiden tekemiseen ilman ohjelmointia | AI-lähettiläs | Tampereen korkeakouluyhteisö*. AI-lähettiläs. https://blogs.tuni.fi/ai-lahettilas/tekninen-aloitusohje/weka-aloitusohje-koneoppimisen-kokeiluiden-tekemiseen-ilman-ohjelmointia/

Pokhrel, S. (2020). *How to Annotate Your Images Using the MakeSense Tool*. https://xailient.com/blog/how-to-annotate-your-images-using-the-makesense-tool/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. http://arxiv.org/abs/1506.02640

Roboflow. (ei pvm.). *Roboflow Annotate*. Noudettu 6. huhtikuuta 2023, osoitteesta https://roboflow.com/annotate

Seeed Studio. (2023, tammikuuta 4). *Few-Shot Object Detection | Seeed Studio Wiki*. https://wiki.seeedstudio.com/YOLOv5-Object-Detection-Jetson/

Shah. (2023). *Mean Average Precision (mAP) Explained: Everything You Need to Know*. https://www.v7labs.com/blog/mean-average-precision

Simplilearn. (2021, huhtikuuta 28). *What Is Computer Vision: Applications, Benefits and How to Learn It*. Simplilearn.Com. https://www.simplilearn.com/computer-vision-article

Solawetz, J. (2020a, toukokuuta 6). *Mean Average Precision (mAP) in Object Detection*. Roboflow Blog. https://blog.roboflow.com/mean-average-precision/

Solawetz, J. (2020b, kesäkuuta 29). *What is YOLOv5? A Guide for Beginners.* Roboflow Blog. https://blog.roboflow.com/yolov5-improvements-and-evaluation/

Theophilus, S. (2022, huhtikuuta 26). Roboflow: Converting Annotations for Object Detection. *Analytics Vidhya*. https://medium.com/analytics-vidhya/converting-annotations-for-object-detection-using-roboflow-5d0760bd5871

Ultralytics. (ei pvm.-a). *Google Colaboratory*. Noudettu 13. huhtikuuta 2023, osoitteesta https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

Ultralytics. (ei pvm.-b). *Train Custom Data · ultralytics/yolov5 Wiki*. Noudettu 20. maaliskuuta 2023, osoitteesta https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

Ultralytics. (ei pvm.-c). *YOLOv8 Docs*. Noudettu 6. huhtikuuta 2023, osoitteesta https://docs.ultralytics.com/#where-to-start

Vaith, A. (2021, lokakuuta 2). Save precious time with Image Augmentation in Object Detection Tasks. *Medium*. https://alex-vaith.medium.com/save-precious-time-with-image-augmentation-in-object-detection-tasks-2f9111abb851

Valtionvarainministeriö. (ei pvm.). *Pilkahduksia tulevaisuuteen. Tietopolitiikka, tekoäly ja robotisaatio hyvinvoinnin ja taloudellisen menestyksen mahdollistajana Suomessa*.

Xiao, Y., Tian, Z., Yu, J., Zhang, Y., Liu, S., Du, S., & Lan, X. (2020). A review of object detection based on deep learning. *Multimedia Tools and Applications*, *79*(33), 23729–23791. https://doi.org/10.1007/s11042-020-08976-6

Zvornicanin, E. (2022, kesäkuuta 11). *What is YOLO Algorithm? | Baeldung on Computer Science*. https://www.baeldung.com/cs/yolo-algorithm

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojektin aikana pidetään päiväkirjan kaltaista dokumenttia, johon kerätään teknistä tietoa projektista. Tämä tieto analysoidaan opinnäytetyötä varten. Lisäksi projektin eri vaiheissa syntyy kuvia kuvakaappausten avulla, joita käytetään opinnäytetyössä. Kuvista poistetaan kaikki salattava tai tietoherkkä data. Päiväkirjaa ja kuvia säilytetään tekijän OneDrive-pilvitallennustilassa, ja niistä tehdään säännöllisesti varmuuskopioita tekijän tietokoneen ulkoiselle kiintolevylle. Päiväkirjaa ja kuvia säilytetään ulkoisella kiintolevyllä ainakin vuoden verran opinnäytetyön valmistumisesta.

Opinnäytetyön aineiston ja tulokset omistaa toimeksiantoyritys. Tässä opinnäytetyössä ei käsitellä luottamuksellisia tai salassa pidettäviä tietoja eikä henkilötietoja.