SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

ENERGIALAITOKSEN VIDEO-KUVAN PROSESSOINTI TE-KOÄLYN AVULLA

TEKIJÄ: Henri Ahonen

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Sähkö- ja automaatiotekniikan tutkinto-ohjelma	
Työn tekijä(t) Henri Ahonen	
Työn nimi Energialaitoksen videokuvan prosessointi tekoälyn avulla	
Päiväys 16.11.2021	Sivumäärä/Liitteet 31
Toimeksiantaja/Yhteistyökumppani(t) KPA Unicon Oy	

Tiivistelmä

Opinnäytetyön tarkoitus oli selvittää, voidaanko tekoälyyn pohjautuvaa objektin tunnistusta hyödyntää energialaitoksen videokuvan prosessoinnissa. Opinnäytetyön tavoite oli luoda sovellus, joka prosessoi ja tunnistaa objekteja energialaitoksen videokuvasta. Työn toimeksiantajana toimi KPA Unicon Oy, joka luo ja toteuttaa puhtaan energian ratkaisuja.

Opinnäytetyössä luotiin WPF- ja konsolisovellus .NET ympäristöön. WPF-sovellus oli ensimmäinen prototyyppisovellus, jonka perusteella luotiin konsolisovellus. Konsolisovellus käytti olemassa olevia toimeksiantajan projekteja tiedontallennuksessa, sekä valmiiksi opetettuja ONNX-neuroverkkoja GitHubista.

Työn lopputulokseksi saatiin konsoliapplikaatio, joka täyttää toimeksiantajan vaatimat tavoitteet. Sovellus pystyy tunnistamaan hyvällä tarkkuudella videokuvasta objekteja ja tallentamaan ne paikalliseen MongoDB-tietokantaan.

Avainsanat
Objektin tunnistus, Tekoäly, ML.NET

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Electrical and Automation Engineering			
Author(s) Henri Ahonen			
Title of Thesis Analysing Powerplant Surveillance Video with Artificial Intelligence			
Date	16 November 2021	Pages/Appendices	31
Client Organisation /Partners KPA Unicon Oy			

Abstract

The aim of this thesis was to research and create a proof of concept application which uses artificial intelligence to detect objects from an energy plant's surveillance video. The main objective of the application was to detect objects from video and save detection information to a local database.

The proof of concept application was developed using Microsoft's .NET open-source platform and ML.NET framework. The application used existing YoloV4 neural network to process and detect objects from the energy plant's surveillance video.

As a result of this thesis, the created proof of concept application gave useful information to the client on how object detection can be used and implemented in energy plant's surveillance video. The application also gave proof that using pre-trained neural networks can be viable in different environments.

Keywords

Object detection, Artificial intelligence, ML.NET

SISÄLTÖ

KUVALUETTELO

1 JOHDANTO

Suomessa ja maailmalla energia-ala on monipuolinen ja työllistää suuren määrän ihmisiä. Energia-ala on yhteiskunnan tärkeimpiä aloja, sillä sen avulla tuotetaan sähköä, kaukolämpöä ja kaukojäähdytystä. Valtaosa yhteiskunnan toiminnallisuudesta on riippuvainen energialaitoksien tuotannosta. Energialaitoksien ylläpitäminen ja uusien teknologioiden kehittäminen on tärkeää ihmiskunnalle. Uusien keksintöjen ja teknologioiden kehitys energia-alalla on tärkeää, jotta energiantuotannosta saadaan ympäristöystävällisempää, luotettavampaa ja tehokkaampaa.

Opinnäytetyön tarkoitus on luoda konseptitodistus, joka todistaa miten tekoälyä ja objektien tunnistusta voidaan hyödyntää energialaitoksissa. Sovelluksen tarkoitus on tunnistaa energialaitoksen videokuvasta objekteja ja tallentaa sovelluksen havaitsemat tiedot tietokantaa. Sovelluksen avulla laitoksessa työskentelevät voisivat tarkistella laitoksella tapahtuvia asioita, kuten esimerkiksi polttoainerekan käymistä laitoksen pihalla. Laitokset ovat myös usein miehittämättömiä, joten sovelluksen avulla laitoksella tapahtumat voitaisiin helposti tarkistaa myös etävalvonnan kautta. Opinnäytetyön toimeksiantaja on KPA Unicon Oy, joka toteuttaa puhtaan energian ratkaisuja energialaitoksille. Työ on osa KPA:n tekoälyhanketta, jossa tutkitaan, miten tekoälyä voidaan hyödyntää energialaitoksissa.

1.1 KPA Unicon Oy

KPA Unicon Oy on 1990-luvulla perustettu perheyritys, joka toimii energiamarkkinoilla. KPA unicon työllistää noin 200 henkilöä. KPA Unicon toteuttaa ja luo puhtaan energian ratkaisuja, sekä uudistaa olemassa olevia järjestelmiä puhtaalla teknologialla. KPA tuottaa myös PlantSys tuotemerkin nimellä olevia web-moduuleja. Nämä moduulit ovat: Plantsys kunnossapito, PlantSys raportointi, Plantsys päiväkirja ja PlantSys materiaalivirrat. KPA käynnisti vuonna 2020 tekoälyhankkeen, jossa tutkitaan, miten tekoälyä voidaan hyödyntää energiantuotannossa. Tutkimuskohteita ovat mm. laitteiden automaattinen kunnonvalvonta, laitoksen tilanneraportti ja operointiehdotukset.

1.2 Tekoälyn soveltaminen KONE-yrityksessä

Tekoälyä on hyödynnetty ja sovellettu pitkään Kansainvälisissä yrityksissä. Esimerkki tekoälyn soveltamista on suomalainen yritys Kone. Kone huoltaa ja tekee hissejä maailmanlaajuisesti. Koneella on 1.1 miljoonaa hissiä maailmanlaajuisesti. Kone aloitti 2017 datavetoisen suunnitelman, jonka tarkoitus oli kerätä miljoonista koneista tietoja ja analysoida ne tekoälyn avulla. Kerätty tieto prosessoitiin koneoppimisen avulla, joka jaettiin muille operaattoreilla ja huoltoyrityksille. 1980-luvulla Kone aloitti tietokoneiden opettamisen, jolloin prosessorit oli suunniteltu arvioimaan keskimääräisen matkustajamäärän jokaisessa kerroksessa ja sopeutumaan sen mukaan. Nykyään KONE omistaa yli miljoona hissiä, jotka ovat pilveen kytkettynä. Hissit sisältävät sensoreita, joiden avulla saadaan tietoa, milloin hissi lähtenyt ja pysähtynyt kerroksiin, sekä sensoreita, joista saadaan lämpötilamittauksia ja kiihdytystietoa. Näiden tietojen perusteella voidaan arvioida, miten laitteisto toimii ja kuinka ihmiset liikkuvat rakennuksessa. Tämän saman mainitun datan avulla koneoppiminen voi luoda malleja, jonka

perusteella tietokone voi arvioida milloin vikatilanteina ja hajoamisia todennäköisesti tapahtuu. (Marr, 2019)

2 KÄSITTEET

Tekoäly	Tekoälyn tarkoitus on jäljitellä inhimillistä päättelykykyä. Tekoälyä analysoi sille annettua tietoa ja tekee näistä päätöksiä.
Koneoppiminen	Koneoppimisella tarkoitetaan tietokoneen oppimista itsenäisesti syötetyn datan avulla.
Objektin tunnistaminen	Objektin tunnistus on osa tietokonenäköä, jossa tunnistetaan ja paikannetaan haluttuja objekteja kuvista ja videoista.
Objekti	Objektilla tarkoitetaan esinettä tai oliota.
Tietokonenäkö	Tietokonenäkö on tekoälyn osa-alue, jossa tietokone opetetaan tunnistamaan erilaisia objekteja kuvista ja videoista.
Syväoppiminen	Syväoppiminen on osa koneoppimista, joka perustuu esimerkin avulla oppimiseen. Syväoppiminen muistuttaa ihmisen oppimista.
Avoin lähdekoodi	Avoin lähdekoodi perustuu sen avoimuuteen. Lähdekoodi on julkinen ja kaikkien nähtävillä ja muokattavissa.
Neuroverkko	Neuroverkko on syväoppimisen menetelmä, jossa mallinnetaan ihmisen aivojen rakennetta.
NuGet	Microsoftin paketinhallintaohjelma, jonka avulla kehittäjät voivat jakaa tekemiään koodeja muille kehittäjille.
ONNX	(Open Neural Network Exchange) Avoin neuroverkon lähde formaatti, jonka tarkoitus on olla yhteensopiva useiden tekoäly-viitekehyksien kanssa.
.NET	Microsoftin tarjoama avoin viitekehys kehittäjille
XAML	(Extensible Application Markup Language) Microsoftin deklaratiivinen xml-kieli, jota käytetään mm. WPF sovelluksen käyttöliittymän rakentamiseen.

3 TEORIA

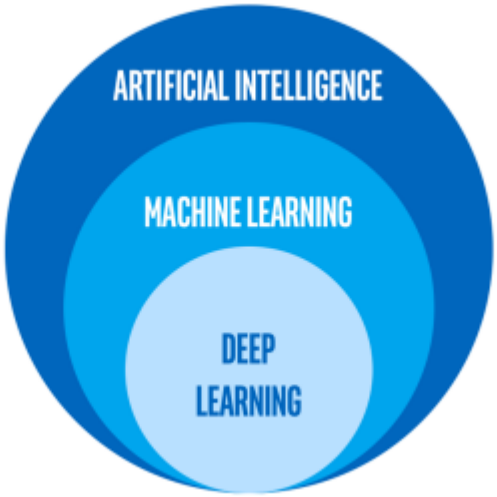
Tämä osuus opinnäytetyöstä käsittelee keskeisempiä käsitteitä ja teoriaa opinnäytetyöstä.

3.1 Tekoäly

Alan Turing kirjoitti 1950-luvulla paperin, jossa hän pohti tietokoneiden kykyä ajatella. Turing keksi kuvitteellisen imitaatiopelin, jossa verrataan koneen suorituskykyä ihmiseen. Pelin idea oli asettaa kone ja ihminen eri huoneisiin, jonka jälkeen ulkopuolinen ihminen kuulustelee, miten kone ja ihminen eroavat toisistaan. Kuulustelija ei tiedä missä huoneessa tietokone tai ihminen on. Turing väitti, että jos kuulustelija ei onnistunut erottamaan ihmistä ja konetta erikseen, niin kone voitaisiin luokitella älykkääksi. Testi osoittautui kuitenkin vaikeaksi, sillä ihmiset ovat luovempia ja järkevimpiä vastauksissaan ja osaavat perustella päätöksiään paremmin kuin tietokone. Tietokoneet ovat toisaalta taas nopeampia ja luotettavampia laskutoimituksissaan. Turing väitti, että tietokoneita voidaan pitää älykkäinä, koska tietokoneet eivät voi olla väärässä laskelmissaan. (Gupta & Mangla, 2020, ss. 2-3)

Vuonna 1956 John McCarthy keksi termin tekoälyn. Tekoälylle on ehdotettu useita eri määrityksiä. Yksi esimerkki siitä, miten tekoälyn voidaan määritellä, on että tekoäly on osa tietojenkäsittelytiedettä, jossa päämääränä on luoda älykkäitä ratkaisuja, jotka matkivat inhimillistä älyä. (Gupta & Mangla, 2020, s. 4)

Tekoäly on laaja käsite, jolla kuvataan tietokoneen kykyä imitoida ihmisen älykkyyttä. Tekoälyä käytetään ennustamiseen, automatisointiin ja optimoimaan asioita, joita ihmiset ovat aiemmin tehneet. Tekoälyä käytetään esimerkiksi kuvantunnistuksessa, äänen tunnistuksessa tai asioiden päättämisessä.



Kuva 1 Tekoälyn osa-alueet (Intel, 2021)

Tekoäly koostu osa-alueista (kuva 1). Nämä osa alueet ovat: koneoppiminen (engl. machine learning) ja syväoppiminen (engl. deep learning). Jokainen näistä alueista on osajoukko toisesta komponentista. Esimerkiksi Koneoppiminen on tekoälyn osajoukko ja syväoppiminen on osajoukko koneoppimisesta. (IBM ClIBM Cloud Education, IBM Cloud Education, 2020).

3.2 Koneoppiminen

Koneoppiminen on tekoälyn osa-alue, jossa tietokone oppii kokemuksen avulla. Koneoppimisen inspiraatio on tullut ihmisen tavasta oppia asioita. 1900-luvusta lähtien, ymmärrys ihmisen oppimisesta on kasvanut huomattavasti. (Mechelli & Vieira, 2019, s. 2)

Koneoppiminen on yksiselitteisesti tapa, jossa tietokoneen oppiminen on automatisoitu. Nykypäivänä koneoppiminen on laaja käsite, joka kattaa monenlaisia sovelluksia ja ohjelmia.

Koneoppiminen voidaan jakaa kahteen eri oppimismalli: valvottuun ja ei valvottuun oppimiseen. Valvotussa oppimisessa käyttäjä opettaa tietokoneen tiedetyn tiedon avulla. Ei-valvotussa oppimisessa tietokone kehittää vastauksen tuntemattoman tiedon perusteella. Ei-valvottua oppimista käytetään usein kuvioiden etsimiseen uudesta tietokokoelmasta. (Wehle, 2017)

Datan laatu on tärkeä koneoppimisessa. Esimerkiksi datan huono laatu ja datan puute voivat aiheuttaa ongelmia koneoppimisessa. Ideaalisessa tilanteessa tietoaineisto on valmiissa muodossa ja sitä ei tarvitse muokata. Muuten tietoaineiston kerääminen yhdestä tai useasta tietolähteestä on tarpeellista. Tietolähteitä voivat olla esimerkiksi tiedostot, tietokannat ja web-palvelimet. Kun tietoa kerätään, on tärkeää siivota data eri tekniikoilla. Esimerkiksi datan kopioiden poisto tai puuttuvan datan täydentäminen ovat menetelmiä, joita datan siivoamisessa käytetään. (Campesato, 2020, ss. 24-31)

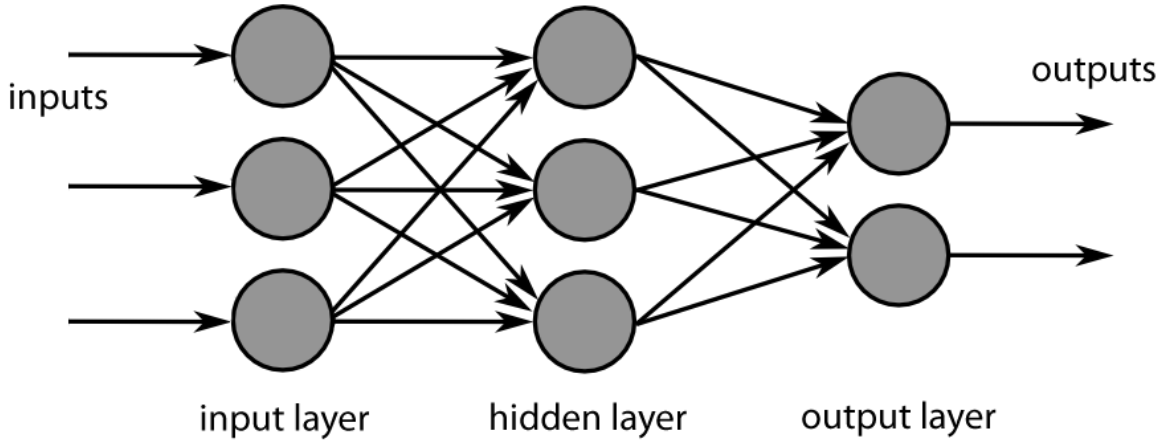
Koneoppimisen ydinideana on vastata kysymykseen, miten voidaan luoda ohjelma, joka oppii ja paranee kokemuksen kautta. Koneoppimiselle ei ole universaalista määritelmä, mutta usein koneoppimisella tarkoitetaan tietokoneen kykyä tunnistaa kuvioita annetusta datasta ja käyttää näitä opittuja

kuvioita uuden datan avulla ennustamiseen (Mechelli & Vieira, 2019, s. 4). Perinteisessä ohjelmoinnissa luodaan itse sovellukselle logiikka, jonka kone suorittaa. Koneoppiminen eroaa tästä, sillä ohjelmoija itse ei luo logiikkaa, vaan tietokone oppii esimerkkien ja kokemuksen avulla ja luo tämän perusteella logiikan. (Newnham, 2018, s. 8)

Neuroverkot

Neuroverkko imitoi ihmisen aivojen rakennetta. Neuroverkko kostuu neljästä eri pääkomponentista: sisääntulosta, painoista, vakiotermeistä ja ulostulosta.

Neuroverkko koostuu neuroneista, jotka yhdistyvät toisiinsa. Verkko koostuu useasta kerroksesta. Verkko pystytään kouluttamaan ja tunnistamaan kaavoja äänestä ja kuvista. (The MathWorks, Inc, s. a.)



Kuva 2. Neuroverkon rakenne (Chrislb, 2010)

Kuvasta 1 nähdään, miten neuroverkko rakentuu. Verkko koostuu sisääntulokerroksesta (engl. input layer), piilotetuista kerroksista (engl. hidden layer) ja ulostulokerroksesta (engl. output layer). Jokaisella neuronilla on niin sanottu paino, jonka avulla neuronin signaalin vahvuus vaihtelee. Neuroverkkoa luetaan usein vasemmalta oikealle. (Arnx, 2019)

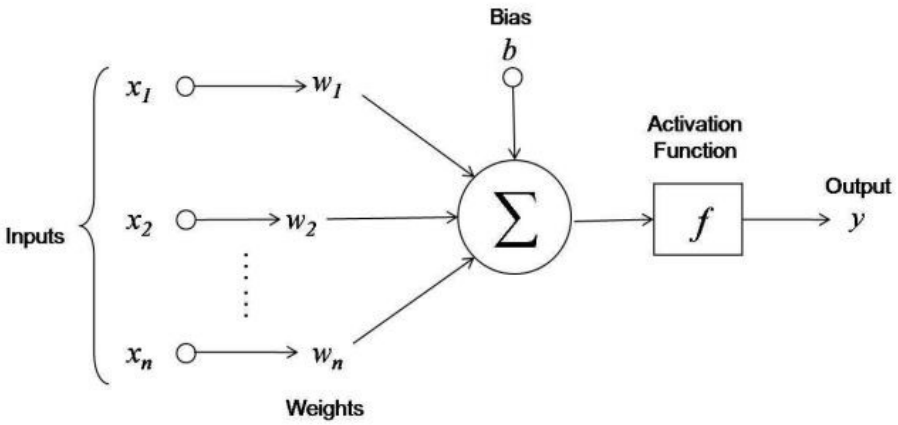


Figure 2 — Operations done by a neuron

Kuva 3 Neuronin operointoi (Arnx, towardsdatascience, 2019)

Kuvasta 2 nähdään miten neuroni operoi. Neuroni lisää kaikki arvot yhteen, jotka tulevat toisista neuroneista (arvot x1, x2 ja x3). Saadut arvot kerrotaan ennen kuin ne lisätään yhteen. Kertoja toimii ns. paino (engl. weight), joka määrittää liitoksen kahden eri neuronien välillä. Jokaisella liitoksella on oma paino, jota voidaan muokata oppimisprosessin aikana. Painojen lisäksi on olemassa vakiotermi (engl. bias), joka voidaan lisätä laskettuun arvoon. Vakiotermi ei tule tietyltä neuronilta, vaan se voidaan valita ennen oppimisprosessia.

Neuroverkon opetusprosessi alkaa sisääntuloarvoista, joista verkko pyrkii saamaan halutun tuloksen. Sisääntuloarvo sisältää leiman, joka selittää minkä arvon neuroverkon pitäisi arvata lopputulokseksi. Jos neuroverkon valinta osui oikeaan, painot pidetään samana ja verkolle annetaan uusi sisääntuloarvo. Jos verkko ei kuitenkaan saa oikea arvoa lopputulokseksi, aikaisemmin mainittuja painoja muutetaan. Verkolla on myös oppimisnopeusparametri, joka määrää kuinka paljon painoja muutetaan. (Arnx, 2019)

3.3 Syväoppiminen

Syväoppiminen on neuroverkon osajoukko. Syväoppimisella viitataan neuroverkon kerroksien syvyyteen. Neuroverkko, joka koostu useammasta kuin kolmesta kerroksesta voidaan luokitella syväoppimisen algoritmiksi. Syväoppiminen tarvitsee pienemmän määrän ihmisen väliintuloa verrattuna perinteiseen koneoppimiseen. Toisaalta syväoppimisen on rakenteeltaan monimutkaisempaa ja tarvitsee suuremman määrän dataa sen koulutukseen. (IBM ClIBM Cloud Education, IBM Cloud Education, 2020)

Syväoppiminen muistuttaa hyvin paljon ihmisen kykyä oppia asioita. Syväoppiminen oppii tunnistamaan kuvioita ja luokittelemaan tietoa tietolähteestä, samoin tavoin kuin ihmisen aivot. Aivot vertaavat uutta tietoa jo tunnettuun tietoon saadakseen selvyyttä. Tämä idea on sama syväoppimisessa.

Syväoppimisella tietokone opetetaan samoin tavoin kuin ihmiset ja eläimet: esimerkin avulla. Syväoppimisessa kone oppii suoraan kuvan, tekstin tai äänen perusteella, jonka perusteella se luokittelee saamansa datan. Syväoppimista käytetään esimerkiksi kuvan tunnistuksessa, äänen tunnistuksessa ja tekstin tunnistuksessa. Kone voi oppia hyvinkin tarkaksi syväoppimisen avulla ja jopa ylittää ihmisen kyvyt. (The MathWorks, Inc, s. a.)

3.4 Tietokonenäkö ja tunnistus

Ihmiset pystyvät tulkitsemaan monimutkaisia kuvia ja tekemään niistä yleisiä johtopäätöksiä. Nykyajan tietokoneet eivät pysty yhtä monimutkaisiin tulkitsemiseen, mutta toimivat silti tärkeinä järjestelminä jokapäiväisessä elämässä. Tietokoneet käyttävät tunnistusta esimerkiksi viivakoodien lukuun, pankkien sekkien prosessointiin tai tuotteen laadunvalvonnassa.

Hahmontunnistuksen prosessissa on neljä eri vaihetta: mittaus, esikäsittely, piirreirrotus ja luokittelu. Mittauksen tarkoitus on muuttaa analoginen data digitaaliseen muotoon. Datan esikäsitellyssä prosessiin kuuluu erilaisia tekniikoita, kuten esimerkiksi kohinan vähennystä, datan parannusta, datan palauttamista ja segmentointia. Piirreirrotuksessa alkuperäisestä datasta etsitään piirteitä, jotta data voidaan muuttaa prosessointia varten helpommaksi. Viimeisessä vaiheessa eli luokittelussa, saatujen piirteiden perusteella luokitellaan tuntemattomasta kuvasta kuvioita.

Kuvantunnistuksessa on kolme peruslähestymistapaa: syntaktinen hahmontunnistus, tilastollinen hahmontunnistus ja kolmannessa tavassa hyödynnetään neuroverkkoja.

Kolmannessa peruslähestymistavassa eli neuroverkon hahmontunnistuksessa, neuroverkot pystyvät oppimaan ja luomaan tunnistettavia piirteitä itse, joita se hyödyntää luokittelussa. Tämä tapahtuu antamalla neuroverkolle data raakamuodossa, jossa ei ole valmiiksi määriteltyjä piirteitä. Jokainen aikaisemmin mainittu menetelmä tarvitsee joko valmiiksi määritellyt parametrit tai opitut piirteet, joiden avulla ratkaistaan haluttu ongelma. Valmiiksi tiedettyjen luokkien piirteet voidaan määritellä etukäteen tai jättää määrittelemättä.

Kun työskennellään määritellyn datan parissa, usein data jaetaan kolmeen eri ryhmään: Koulutus-, validointi- ja testausaineistoon. Koulutusaineiston idea on kouluttaa luokittelija tunnistamaan datasta piirteitä. Koulutusaineiston datan piirteet ovat määriteltyjä, jota voidaan myöhemmin korjata tarvittaessa, jos luokitellussa tapahtuu erehdyksiä. Validoinnissa tarkistetaan luokittelun suorituskyky, jonka perusteella päätellään luokittelun onnistuminen. Viimeisessä vaiheessa testataan, miten luokittelu onnistuu käytännössä. Testauksessa käytetään dataa, jota luokittelija ei ole aikaisemmin nähnyt. Jos testauksen tai koulutuksen lopputulos on hyväksyttävä, mutta testaus ei, niin saatua systeemiä kutsutaan ylioppineeksi.

(Rafael C. Gonzalez, 2018)

4 TEKNOLOGIAT

4.1 ONNX

ONNX (Open Neural Network Exchange) on avoin lähde formaatti, joka tukee yhteensopivuutta viitekehyksien välillä. Neuroniverkko, joka on koulutettu eri viitekehyksessä, voidaan muuttaa ONNX muotoon, jota voidaan käyttää toisessa viitekehyksessä (ks. Kuva 3). Esimerkiksi PyTorch-viitekehyksellä koulutettu malli voidaan muuntaa ONNX-malliksi, jota voidaan käyttää toisessa viitekehyksessä, kuten ML.NET. (Microsoft, 2021)

Kuva 4. ONNX-malli (Microsoft Corporation Oy, 2021)

ONNX GitHubin sivuilla on valmiiksi opetettuja malleja, joita voi käyttää Apache v2-isenssin alla.

4.2 ML.NET

ML.NET on Microsoftin kehittämä tekoälyyn pohjautuva avoin ja ilmainen viitekehys .NET sovellusalustaan. ML.NET:in avulla voidaan rakentaa ja opettaa koneoppimiseen perustuvia malleja sekä käyttää niitä melkein missä tahansa .NET pohjautuvissa sovelluksissa. ML.NET sisältää työkaluja joiden tarkoitus on helpottaa koneoppimista itsetehtyihin sovelluksiin. ML.NET:in avulla voidaan luoda monta eri sovelluksen käyttökohdetta: hinnan ennustaminen, kuvan tunnistus, objektin tunnistus ja liikevaihdon ennustaminen. Microsoftilla on olemassa valmiita esimerkkejä ML.NET:in sovelluksista GitHubissa.

ML.NET:in käyttäminen ei vaadi kehittäjältä laajaa tietoa tekoälystä tai koneoppimisesta. ML.NET kirjaston käyttö ja sen integroiminen .NET sovellusalustoihin on helppoa.

4.3 Netron

Netron on sovellus, jonka avulla voidaan katsoa neuroverkon, syväoppimisen ja koneoppimisen malleja. Netron tukee seuraavia käyttöjärjestelmiä: macOS, Linux, Windows. Netron pyörii myös tavallisessa nettiselaimessa. (Roeder, s. a.)

Netron on hyödyllinen työkalu, sillä sen avulla voidaan nopeasti tarkastella ONNX-mallin sisään- ja ulostuloparametreja. Netronin avulla voidaan myös tarkistella tarkemmin mallin kerroksia.

5 TYÖ

5.1 Työn aloittaminen

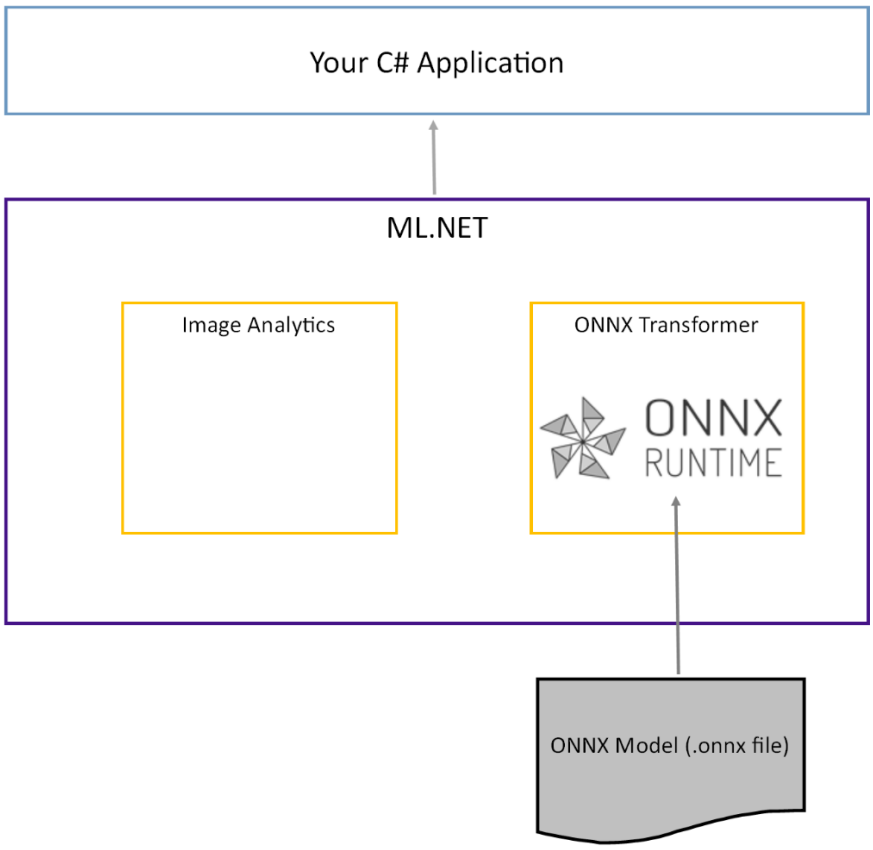
Konseptitodistuksen kehitysympäristöksi valittiin Microsoftin .NET ympäristö, jotta sovellus voidaan integroida toimeksiantajan järjestelmään helposti. Sovelluksen ensimmäinen tavoite oli tunnistaa energialaitoksen videokuvasta objekteja käyttäen hyväksi valmiiksi opetettuja malleja. Valmiiksi opetettuja ONNX-malleja löytyy GitHubista, jota kehittäjät voivat käyttää vapaasti.

Ensimmäinen prototyyppisovellus tehtiin käyttäen .NET-viitekehyksen WPF- ja ML.NET-kirjastoa. WPF:n avulla voidaan luoda graafisia työpöytäsovelluksia. WPF-sovellus sopii hyvin prototyyppisovelluksen tekemiseen, koska sen avulla voidaan tarkastella objektin tunnistusta visuaalisesti.

Sovelluksen tekeminen alkoi luomalla .NET WPF-projekti. Sovellukseen asennettiin tarvittavat ML.NET kirjastot: Microsoft.ML, Microsoft.ML.ImageAnalytics, Microsoft.ML.OnnxTransformer ja Microsoft.ML.OnnxRuntime.

5.1.1 Valmiiksi opetettujen ONNX-mallien käyttö

ONNX-GitHubista löytyy laaja valikoima opetettuja malleja esimerkiksi tietokonenäköön ja äänen- ja puheentunnistukseen. Objektin tunnistukseen valikoitu TinyYoloV2, joka on supistettu versio YoloV2 mallista. TinyYolov2 pystyy tunnistamaan 20 eri objektityyppiä kuvasta. TinyYoloV2:n käytöstä on olemassa valmis Microsoftin esimerkki, joka perehdyttää, kuinka ONNX-mallia voidaan käyttää ML.NET kirjastoa hyödyntäen.



Kuva 5. ONNX integroiminen ML.NET viitekehykseen (Microsoft Corporation Oy, 2021)

ONNX:n ajonaikainen ympäristö käyttää keskusprosessoria. ONNX tukee myös graafisenprosessointiyksikön ajonaikaista ympäristöä, joka on huomattavasti nopeampi kuin keskusprosessori. Opinnäytetyön sovelluksen pyörittämiseen käytettiin kannettavaa tietokonetta.

Käytetyn tietokoneen ominaisuudet:

- Lenovo ThinkPad
- Intel Core i5-8265U CPU @ 1.60 GHz, 4-ydintä
- Windows 10 Business – 10.0.19042 build 19042
- 8GB-keskusmuistia

5.1.2 Videokuvan käsittely

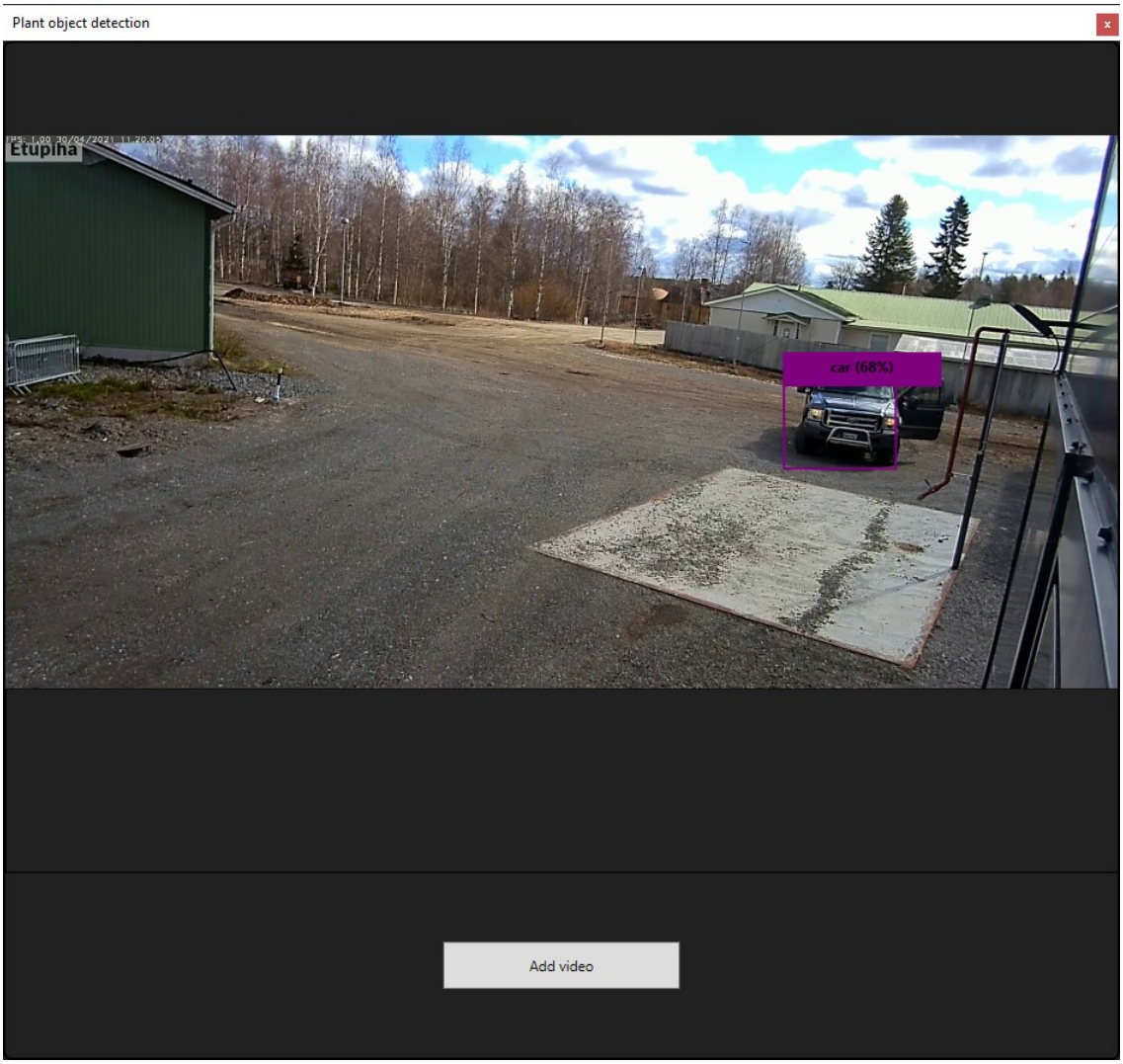
Prototyypissä käytettiin videoleikkeitä, jotka on kuvattu energialaitoksen pihasta. Videoleikkeissä esiintyi erilaisia tunnistettavia objekteja, esimerkiksi rekkoja, ihmisiä, autoja ja traktoreita. Videokaappauksen kuvataajuus oli 1 per sekunti. Videokuvan käsittelyyn käytettiin OpenCv-kirjastoa, josta on olemassa valmis kirjasto .NET-ympäristöön.

Sovellus käy videon jokaisen kuvaruudun läpi, josta malli pystyy tunnistamaan objekteja. ML.NET-mallin rakentaja osaa automaattisesti käsitellä kuvan, joten kuvaa ei tarvitse etukäteen käsitellä. Kun malli on käsitellyt kuvan, se antaa tuloksen, jossa on objektien nimet, koordinaatit ja tunnistamistodennäköisyydet. Annetuilla koordinaatinarvoilla, voidaan rajata objektien päälle laatikot.

Kuvan laatu ja kuvakulma vaikuttavat myös objektin tunnistuksen tarkkuuteen. Opinnäytetyössä käytettävä videomateriaalin videolaatu on täysteräväpiirto ja sen kuvakulma on hieman viistossa (kuva 5).

Kuva 6. Kuvankaappaus laitoksen videokuvasta (KPA Unicon, 2021)

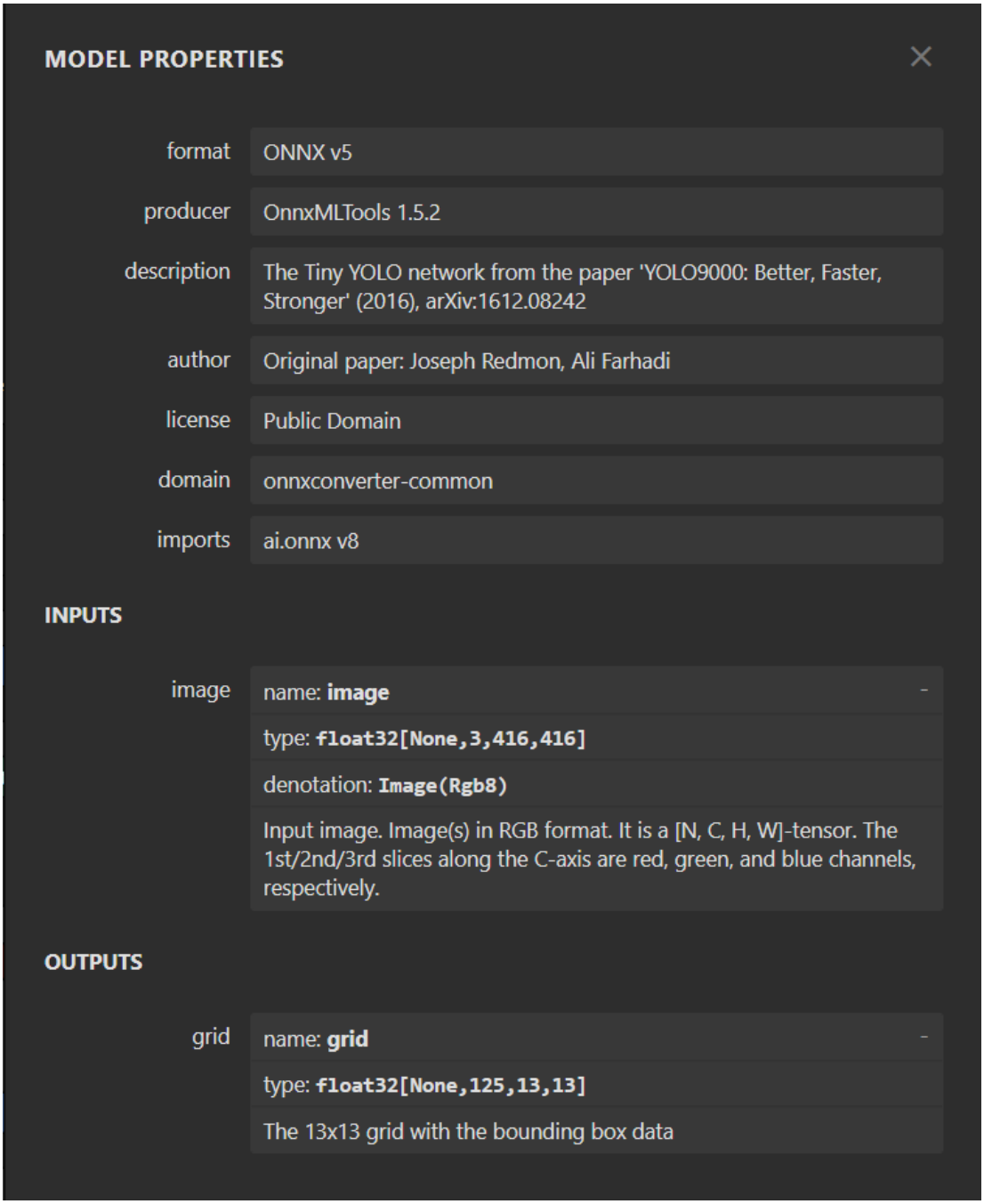
5.2 WPF sovellus



Kuva 7. Kuvankaappaus WPF-sovelluksen pääikkunasta (Henri Ahonen, 2021)

WPF-sovelluksen periaate on näyttää käyttäjälle visuaalisesti objektitunnistus reaaliajassa (Kuva 5). Sovelluksen päänäkymästä pystyy avaamaan videon haluamastaan kansiosta. Kun video on valittu, alkaa sovellus prosessoimaan videota ja renderöimään kuvankaappaus kerrallaan tunnistettuja objekteja. Näin nähdään reaaliaikaisesti, miten käytetty ONNX-malli tunnistaa objekteja videosta.

5.3 TinyYoloV2 malli



Kuva 8. TinyYoloV2 sisään- ja ulostuloparametrit

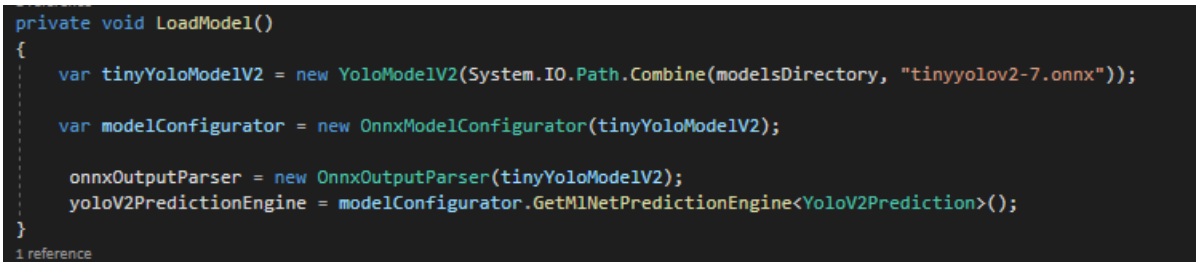
TinyYoloV2-malli on opetettu käyttäen Pascal VOC tietoaineistoa ja se koostuu 15 kerroksesta. ONNX-malleja voidaan tarkastella Netron-sovelluksessa, jonka avulla voidaan nähdä sisään- ja ulostuloparametrit. Sisään – ja ulostuloparametrien tietorakenne on tensorimuodossa. TinyYolov2:n sisääntulokerroksen parametrin nimi on ”image” ja se odottaa tensorin olevan 3x416x416 muodossa. Ulostuloparametrin nimi on ”grid” ja se generoi tensorin 125x13x13 muodossa. (Microsoft, 2021)

Kun mallin sisääntulo ja ulostulot ovat selvillä, annetaan ne parametreina ONNX transformer-funktiolle (kuva 7).

```
0 references
private ITransformer SetupMlNetModel(IOnnxModel onnxModel)
{
    var dataView = mLContext.Data.LoadFromEnumerable(new List<ImageInputData>());

    var pipeline = mLContext.Transforms.ResizeImages(resizing: ImageResizingEstimator.ResizingKind.Fill,
        outputColumnName: onnxModel.ModelInput, imageWidth: 416, imageHeight: 416, inputColumnName: nameof(ImageInputData.Image))
        .Append(mLContext.Transforms.ExtractPixels(outputColumnName: onnxModel.ModelInput))
        .Append(mLContext.Transforms.ApplyOnnxModel(modelFile: onnxModel.ModelPath, outputColumnName:
        onnxModel.ModelOutput, inputColumnName: onnxModel.ModelInput, gpuDeviceId: 0));

    var mlNetModel = pipeline.Fit(dataView);

    return mlNetModel;
}
```

Kuva 9. ML.NET - Mallin rakentaminen (Henri Ahonen, 2021)

Mallin rakentaminen ML.NET:n avulla on yksinkertaista. Kuvasta 8 nähdään, miten malli luodaan. ML.NET-mallien luonnissa annetaan kuvan sisään- ja ulostuloparametrit, sekä itse mallin tiedostosijainti.



```
private void LoadModel()
{
    var tinyYoloModelV2 = new YoloModelV2(System.IO.Path.Combine(modelsDirectory, "tinyyolov2-7.onnx"));

    var modelConfigurator = new OnnxModelConfigurator(tinyYoloModelV2);

     onnxOutputParser = new OnnxOutputParser(tinyYoloModelV2);
     yoloV2PredictionEngine = modelConfigurator.GetMlNetPredictionEngine<YoloV2Prediction>();
}
1 reference
```

Kuva 10. ML.NET -TinyYoloV2 mallin lataaminen (Henri Ahonen, 2021)

Mallin lataaminen tapahtuu helposti, kun mallin konfiguraatio on tehty. Malli ladataan heti sovelluksen alussa (kuva 9).

```
2 references
public string[] Labels { get; } =
{
    "aeroplane", "bicycle", "bird", "boat", "bottle",
    "bus", "car", "cat", "chair", "cow",
    "diningtable", "dog", "horse", "motorbike", "person",
    "pottedplant", "sheep", "sofa", "train", "tvmonitor"
};
```

Kuva 11. TinyYolov2-mallin tunnistettavat objektit (Henri Ahonen, 2021)

TinyYoloV2-malli tunnistaa 20 eri luokan objektia kuvasta (kuva 10). Näistä kaksi objektia esiintyy todennäköisesti laitoksen videokuvassa: ihminen ja auto.

5.3.1 TinyYoloV2 tarkkuus

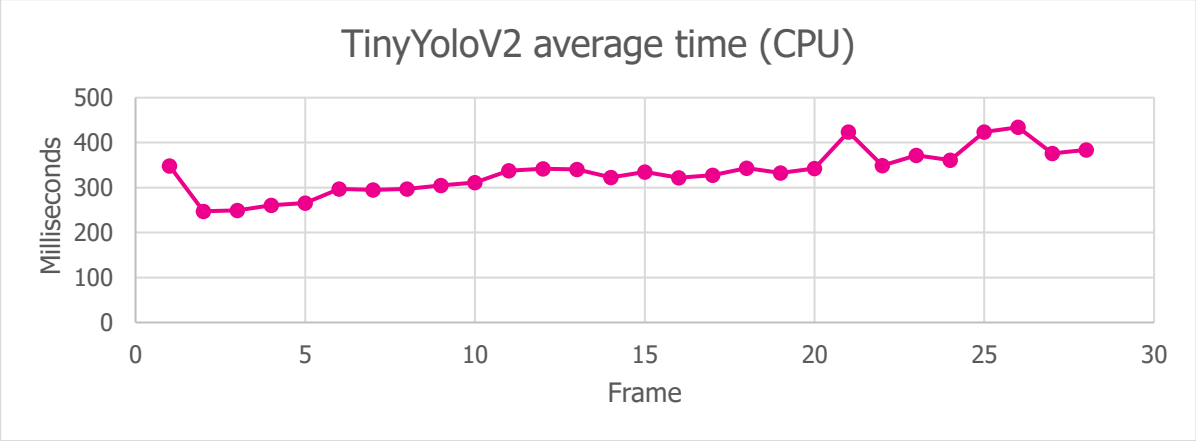
Syötin TinyYoloV2-malille eri videoita laitoksen pihasta. Syötetyissä videoissa esiintyy erilaisia objekteja mm. autoja, ihmisiä, rekkoja ja traktoreita. Katselmoin mallin tunnistuskykyä ja huomasin joissakin kuvankaappauksissa, että malli joko ei tunnistanut objektia tai se luokitteli sen vääränlaiseksi. Esimerkiksi TinyYoloV2-malli joissakin kuvankaappauksissa ei pystynyt tunnistamaan laitoksella ajavaa autoa, sekä tunnisti ihmisen yhdessä kuvaruudussa koiraksi. Osittainen selitys miksi malli tunnistaa objekteja vääränlaiseksi, voi johtua kuvakulmasta. Ihminen kävelee videossa kuvaruudun reunalle, joten malli ei pysty tunnistamaan ihmisille olennaisia piirteitä tarpeeksi hyvin, vaan tulkitsee kuvassa esiintyvän ihmisen koiraksi.

Kuva 12. Kuvankaappaus TinyYoloV2 tunnistuksesta (KPA Unicon, 2021)

Kuvasta 11 nähdään tunnistettu auto, jonka yläpuolella on kerrottu objektin nimi ja mallin varmuus tunnistuksesta. Jokaiselle objektille on oma väri tunnistuslaatikossa, joka helpottaa visuaalisesti käyttäjää tunnistamaan useita eri objekteja.

5.3.2 TinyYolov2 Suorituskyky

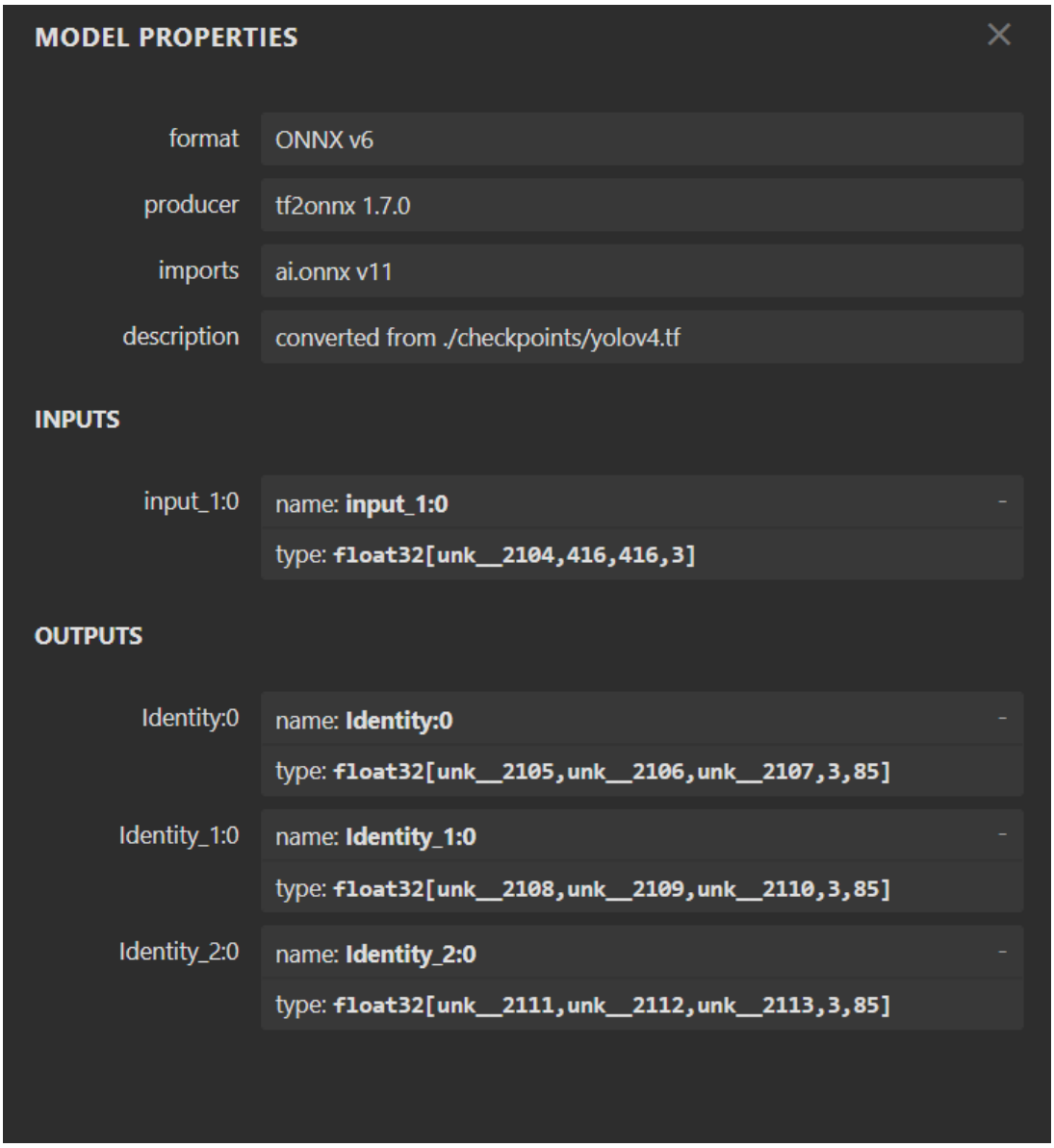
TinyYoloV2-mallilla meni noin 200-400ms yhden kuvan tunnistamiseen. Sovellus käytti ONNX-ajoympäristössään keskusprosessoria. Suoritusaika on melko pitkä, joten sen käyttäminen reaaliaikaiseen tunnistukseen ei ole mahdollista, mutta malli on tarpeeksi nopea prototyypissä käytettävän videon prosessointiin.



Kuva 13. Pistekaavio TinyYoloV2 keskimääräisestä suoritusajasta

5.4 YoloV4 malli

5.4.1 YoloV4



Kuva 14. YoloV4 mallin sisään- ja ulostuloparametrit

YoloV4 ONNX-mallia voidaan tarkistella Netronin avulla (Kuva 13). YoloV4 sisääntuloparametrin nimi on ”input_1:0” ja se odottaa datan olevan 1x416x415x3 tensori tietorakenteen muodossa. Ulostuloja on kolme, joiden nimet ovat: ”Identity:0”, ”Identity_1:0” ja ”Identity_2:0”. Näiden tietojen perusteella projektiin voidaan luoda objektimalli, jota käytetään ONNX-mallin luomisessa.

```
private static void LoadYoloV4Model()
{

    var yoloModelV4Path = System.IO.Path.Combine(modelsDirectory, "yolov4.onnx");

    var modelconf = new OnnxModelConfigurator(yoloModelV4Path);

    yoloV4PredictionEngine = modelconf.GetYoloV4PredictionEngine();

}
```

Kuva 15. ML.NET - YoloV4 mallin luominen (Henri Ahonen, 2021)

Mallin luominen tapahtuu melkein samalla tavalla kuin TinyYolov2 (Kuva 14).

```
//List of all classes that YOLOV4 can detect
static readonly string[] classesNames = new string[] { "person", "bicycle", "car", "motorbike", "aeroplane", "bus",
    "train", "truck", "boat", "traffic light","fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
    "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack","umbrella", "handbag", "tie",
    "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball bat", "baseball glove", "skateboard",
    "surfboard","tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl", "banana", "apple",
    "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza","donut", "cake", "chair", "sofa", "pottedplant",
    "bed", "diningtable", "toilet", "tvmonitor", "laptop", "mouse", "remote", "keyboard", "cell phone", "microwave",
    "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush" };
```

Kuva 16. YoloV4 tunnistettavat objektit (Henri Ahonen, 2021)

YoloV4 tunnistaa 80 eri luokan objektia videokuvasta (kuva 15). Tämä on nelinkertainen määrä, kun verrataan TinyYoloV2-mallin tunnistettavien objektien määrään. Mallissa on kuitenkin paljon hyödyttömiä objekteja, joita tuskin laitoksen videokuvassa esiintyy.

5.4.2 YoloV4 mallin tarkkuus

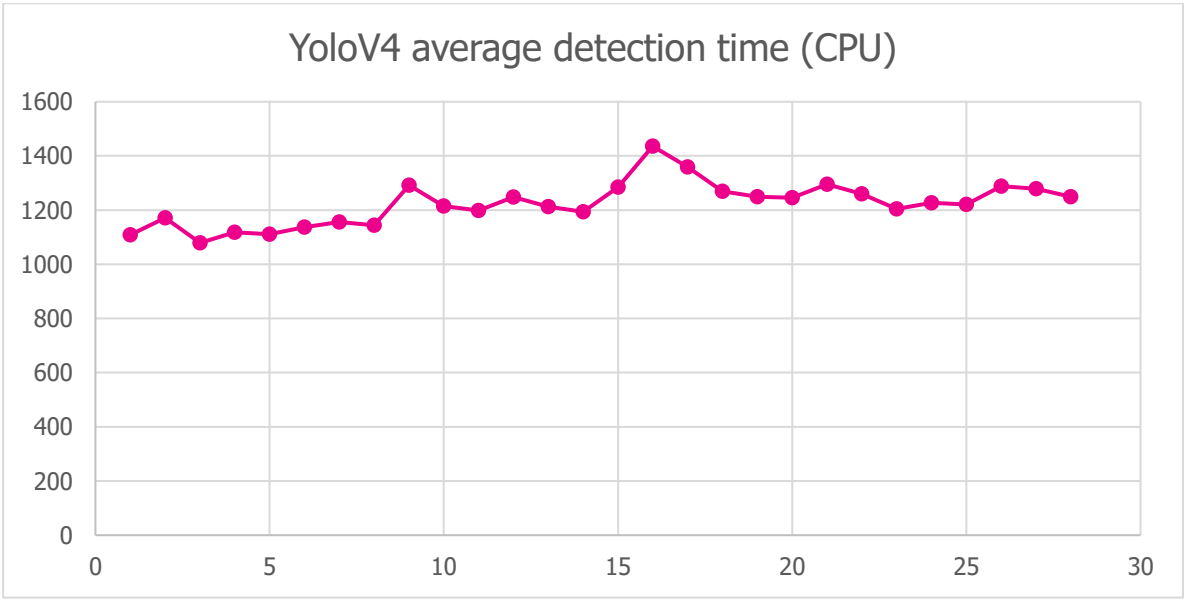
Syötin YoloV4-mallille samat videot, joita TinyYoloV2 myös prosessoi. Näin pystyin vertailemaan mallien suorituskykyä ja tunnistamista. Huomasin ensimmäisenä havaintona, että YoloV4-malli tunnistaa objekteja suuremmalla tarkkuudella, sekä kauempaa esiintyviä objekteja, kun vertailukohteena on TinyYoloV2-malli. Toisaalta TinyYoloV2-mallin kuvankaappauksen prosessoinnissa kului huomattavasti lyhyempi aika, kuin YoloV4-mallilla. YoloV4 on koulutettu suuremmalla koulutusaineistoilla verrattuna TinyYoloV2-malliin, jonka takia sen tunnistus on myös tarkempi ja tunnistettavien objektien

luokkien lukumäärä suurempi. Joissakin tapauksissa YoloV4 kuitenkin tunnisti objekteja väärän-laiseksi. Esimerkiksi rekka, jonka ajokoppia ei näkynyt, YoloV4-malli tunnisti sen junaksi.

Kuva 17. Kuvankaappaus YoloV4 tunnistuksesta (KPA Unicon, 2021)

5.4.3 YoloV4 suorituskyky

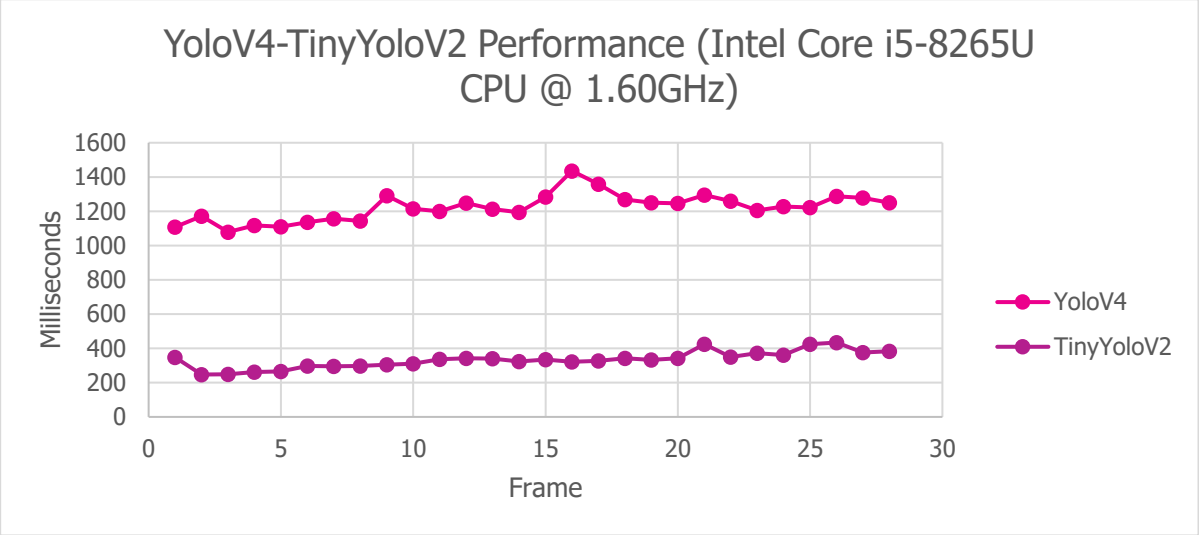
YoloV4 tunnistukseen menevä aika vaihtelee noin 1150-1400ms välillä, kun tunnistuksen ajoympä-ristö käyttää keskusprosessoria. Mallilla menee siis huomattavasti pidempään tunnistukseen, kuin aikaisemman TinyYoloV2-mallilla.



Kuva 18. Pistekaavio keskimääräisestä suoritusajasta

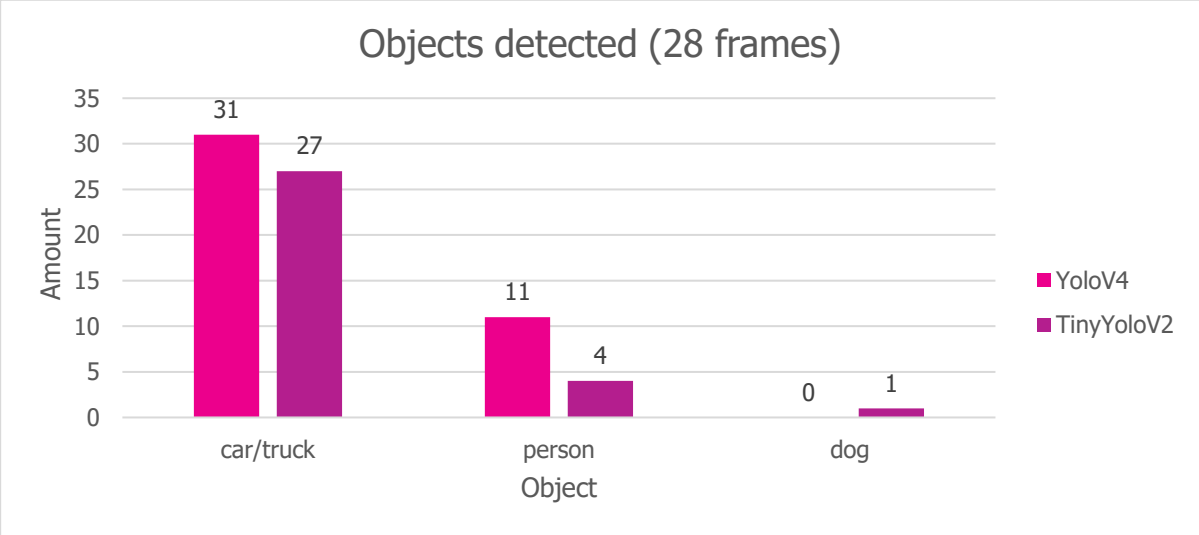
5.5 ONNX-Mallien vertailu

Molemmat mallit käyttivät ONNX suoritusympäristössään keskusprosessoria. Vertailussa käytettiin samaa videota.



Kuva 18. Vertailu YoloV4 ja TinyYoloV2 keskimääräisistä suoritusajoista

Kun malleja ajettiin keskusprosessointiyksikön avulla, huomataan että mallien suorituskyvyllä on huomattava ero. Kuvasta 19 nähdään, että TinyYoloV2 pystyy tunnistamaan jopa kolme kertaa nopeammin kuin YoloV4 malli.



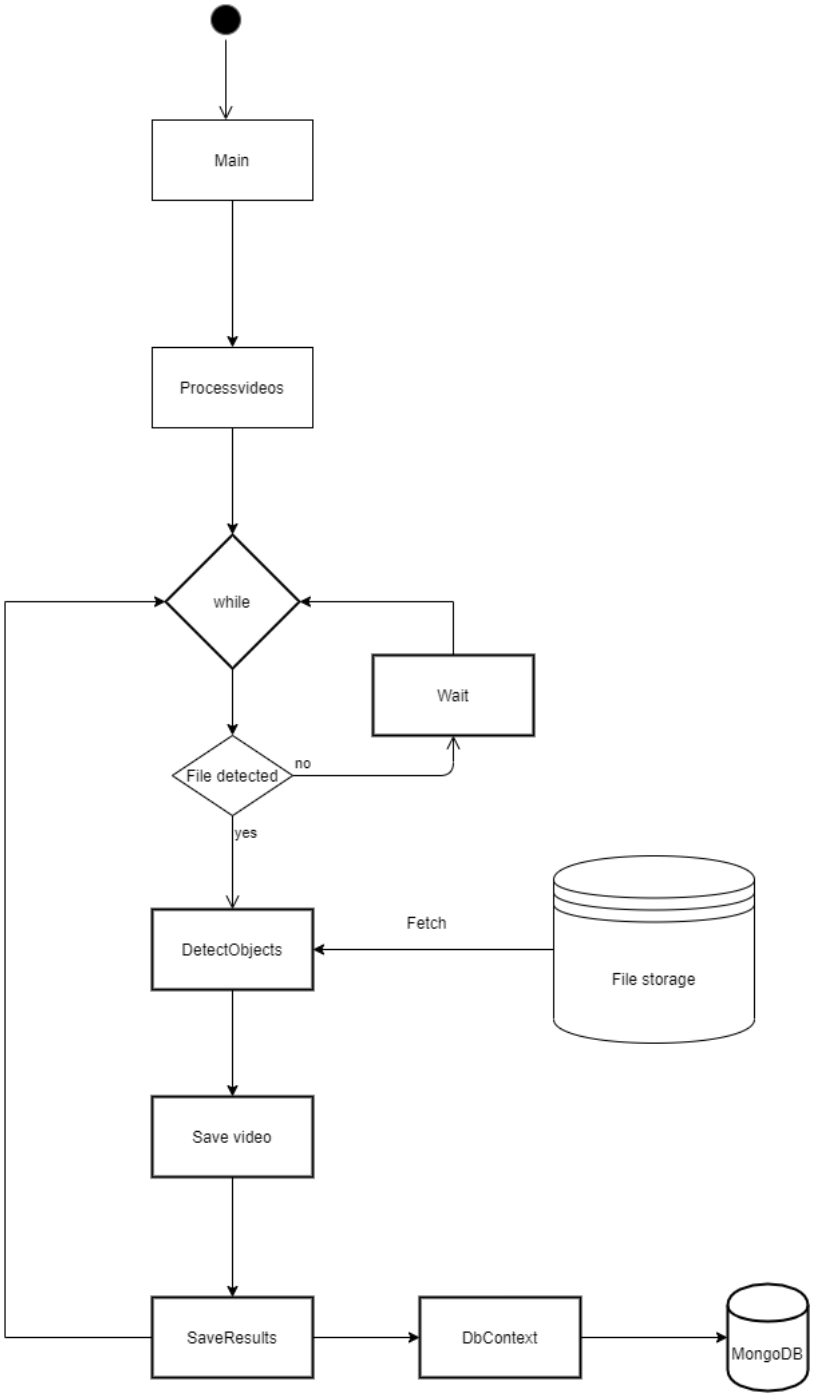
Kuva 19. TinyYoloV2 ja YoloV4 tunnistetut objektit (engl. Objects detected)

Kun taas verrataan mallien tunnistuksien tuloksia, huomataan, että YoloV4 on tarkempi malli tunnistuksissa. Esimerkiksi huomasin, että YoloV4-malli tunnistaa ihmisen jokaisesta kuvasta, missä ihminen näkyy. Kun taas TinyYoloV2 toisaalta ei tunnista ihmistä jokaisesta kuvasta, sekä sen varmuus tunnistuksessa on huomattavasti pienempi kuin YoloV4-mallin.

TinyYolov2 myös virheellisesti tunnistaa kuvasta koiran, vaikka kyseessä on ihminen. YoloV4 myös luokittelee kuvasta nähtävän auton rekkamalliseksi. Kuvasta 19 nähdään, että YoloV4 tunnistaa ihmisen 11 kertaa ja TinyYoloV2 tunnistaa taas ihmisen 4 kertaa, kun kyseissä videossa ihminen esiintyy 18 kuvankaappauksessa. Ero mallien välillä on huomattava

5.6 Konsolisovellus

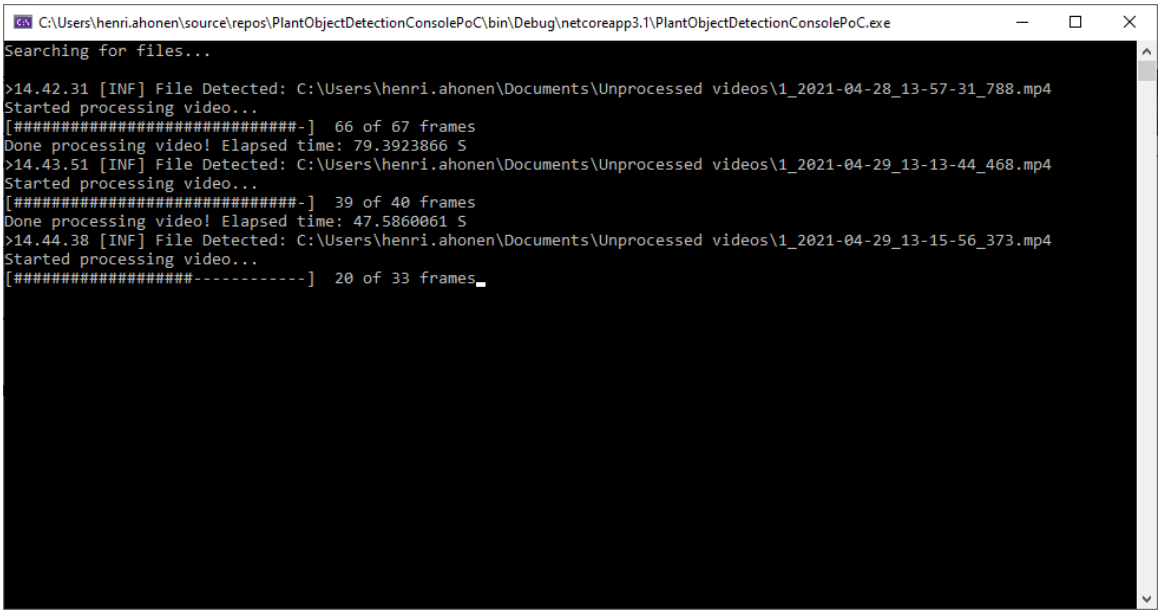
WPF-sovelluksen avulla voidaan katselmoida objektin tunnistusta reaaliaikaisesti. Päädyimme toimeksiantajan kanssa siihen, että konsolipohjaan tehty sovellus olisi lopullinen versio konseptitodistuksesta. Konsolisovelluksen tehtävänä on tarkistaa, onko kovalevyllä uusia videoita, jotka sovellus prosessoi ja tunnistaa niistä esiintyviä objekteja. Kun video on prosessoitu, tallentaa se videon erilliseen kansioon ja poistaa vanhan videon.



Kuva 19. Diagrammi sovelluksesta

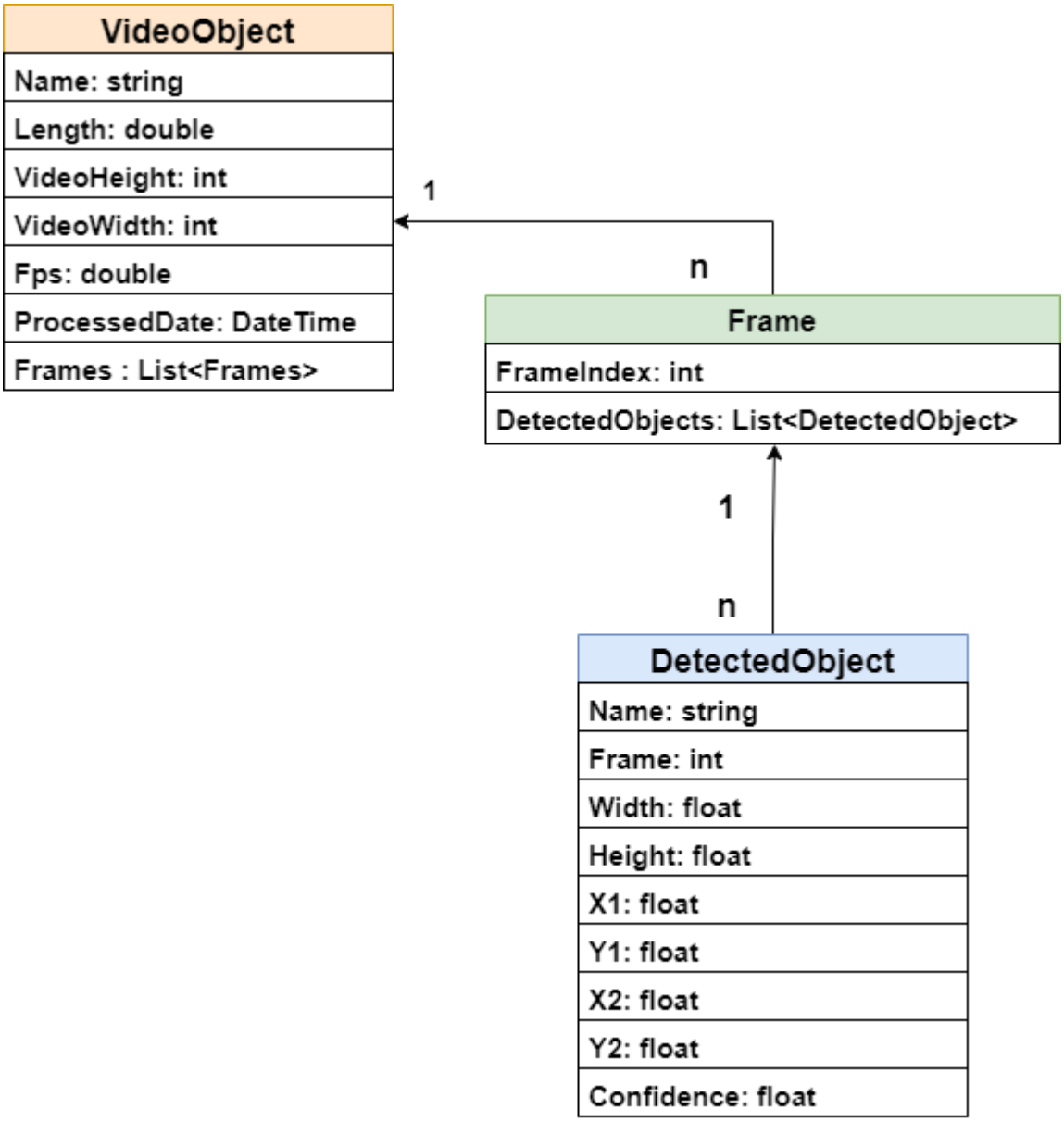
Kuvan 20 diagrammi näyttää, miten sovellus toimii. Sovellus seuraa kansiota ja tarkistaa, onko siihen kansioon ilmestynyt uusia videoita. Kun sovellus huomaa uuden videon, se aloittaa videon prosessoinnin. Sovellus käy silmukan avulla jokaisen kuvankaappauksen läpi ja tallentaa tunnistetut ob-

jektit taulukkoon ja piirtää videoon rajatut laatikot, joissa objektit esiintyvät. Käyttäjä näkee visuaalisesti videoprosessin tilan (Kuva 21). Kun sovellus on prosessoinut videon, se tallentaa prosessoidun videon eri kansioon ja tallentaa paikalliseen tietokantaan saadut tunnistukset. Sovellus aloittaa pollauksen uudelleen, kun se on prosessoinut videon.



Kuva 20. Konsolinäkymä (Henri Ahonen, 2021)

5.7 Tuloksien tallennus



Kuva 21. Tunnistetun objektin tietorakenne (Henri Ahonen, 2021)

Saadut tulokset tallennetaan kuvan 22 mukaiseen objektiin. Objekti koostuu taulukosta kuvankaappauksista, jotka ovat myös objekteja. Kuvankaappaus objektit sisältävät sen kyseisen kuvankaappauksen tunnistetut objektit. Tämän avulla saadaan jokaisesta kuvankaappauksesta tarkat arvot, mitä malli on kokonaisuudessaan tunnistanut videokuvasta.

```
{
  "_id": {
    "$oid": "60bf583f2ee0334bcc75014b"
  },
  "Name": "1_2021-04-29_13-34-30_385.mp4",
  "Length": 39.5,
  "VideoWidth": 1920,
  "VideoHeight": 1080,
  "Fps": 0.8860759493670886,
  "ProcessedDate": {
    "$date": "2021-06-08T11:45:03.157Z"
  },
  "Frames": [
    {
      "FrameIndex": 0,
      "DetectedObjects": [
        {
          "Name": "truck",
          "Frame": 0,
          "Width": 737.427734375,
          "Height": 321.01092529296877,
          "X1": 1181.572265625,
          "Y1": 214.9995574951172,
          "X2": 1919,
          "Y2": 536.010498046875,
          "Confidence": 0.8131089806556702
        },
        {
          "Name": "truck",
          "Frame": 0,
          "Width": 344.7148132324219,
          "Height": 178.7447509765625,
          "X1": 500.9164733886719,
          "Y1": 180.82052612304688,
          "X2": 845.6312866210938,
          "Y2": 359.5652770996094,
          "Confidence": 0.7755470275878906
        }
      ]
    },
```


Kuva 22. Tulokset JSON-muodossa (Henri Ahonen, 2021)

Jokaisesta videosta saadaan kuvankaappauksesta tunnistetut objektit (kuva 23). Jokaisesta videosta saadaan nimi, pituus, ruutunopeus, leveys, pituus ja kuvankaappaustaulukko. Jokaisella tunnistetulla objektilla on ruutunumero x- ja y-koordinaatit, leveys, pituus, objektin nimi ja mallin arvio tarkkuudesta.

5.8 Näytönohjaimen hyödyntäminen

Työn loppuvaiheessa päätin testata myös sovelluksen suorituskykyä käyttämällä näytönohjainta. Onnx-malleilla on olemassa näytönohjainta hyödyntävä suoritusympäristö, josta on olemassa valmis NuGet-paketti Visual Studioon. Suoritusympäristö tarvitsee Nvidian CUDA-rajapinnan omaavan näytönohjaimen toimiakseen ja CUDA-työkalujen lataamisen. CUDA-kehittäjätyökalun lataamiseen tarvitsee käyttäjän rekisteröityä Nvidian kehittäjäohjelman jäseneksi, joka onnistuu heidän nettisivuiltaan (Nvidia, 2021).

Muutamilla parametrien ja koodien muunnoksilla onnistuin asentamaan sovellukseen näytönohjainta käyttävän suoritusympäristön. Huomasin, että yhden kuvankaappauksen käsittelyyn meni noin 70-100ms, joka on alle kymmenesosa verrattuna siihen, kun suoritusympäristö käyttää keskusprosessoria suorituksessa. Vertailu antoi hyvän kuvan siitä, miten pelkän näytönohjaimen käyttö parantaa sovelluksen suorituskykyä.

6 JATKOKEHITYS

Lopputyön tuloksena saatiin toimiva konseptitodistus, jonka välineenä oli sovellus, joka tunnistaa luotettavasti ja tarpeeksi hyvällä tarkkuudella laitoksen videokuvasta erilaisia objekteja. Sovellus täytti asetetut kriteerit onnistuneesti. Sovellukselle jäi kuitenkin useita eri jatkokehitysideoita. Ensimmäinen kehityskohde sovellukselle olisi parantaa sen suorituskykyä. YoloV4 tunnistukseen menevä aika oli noin 1 sekunti per kuva, kun malli käytti suorituksessaan keskusprosessoria. Testasin mallin suorituskykyä näytönohjaimen kanssa, jossa sovelluksella meni tunnistukseen noin 100ms per kuva, ilman mitään muuta optimointia. Myös muita tunnistukseen tarkoitettuja ONNX-malleja voitaisiin hyödyntää ja verrata niiden tarkkuutta ja nopeutta. Opinnäytetyössä jäi neuroverkon siirto-opettaminen tekemättä, joka voisi myös parantaa mallin tunnistuksen suorituskykyä. Prototyyppisovelluksessa käytetty malli tunnisti videokuvasta objekteja hyvin, mutta se luokitteli objekteja joissakin kuvankaappauksissa vääriksi, esimerkiksi joissakin tilanteissa se tunnisti junan, vaikka kyseessä oli rekka. Sovellukseen pystyttäisiin myös luomaan suodatus, joka poistaa videokuvan tuloksista epätodennäköisiä objekteja.

Opinnäytetyössä tehtyä sovellusta ei käytetty oikeassa laitosympäristössä, joten sen testaaminen jäi jatkokehitykseen. Laitoksen valvomotietokoneella pyörii muita sovelluksia, joten olisi tärkeää, että objektin tunnistukseen käytetty sovellus ei veisi liika resursseja tietokoneelta. Sovelluksen heikkous on ehdottomasti sen huono suorituskyky. Ensimmäisessä versiossa sovelluksessa oli myös muistivuoto, joka kuitenkin saatiin korjattua myöhemmin.

Sovelluksessa käytettyä mallia pitäisi myös testata eri laitoksien videomateriaaleilla, sekä eri valaistuksien ja vuodenaikojen kanssa. Tällä tavalla mallista saataisiin parempi käsitys, kuinka hyvin se tunnistaisi objekteja eri ympäristöissä ja olosuhteissa.

7 POHDINTA

Opinnäytetyön lopputuloksena on toimiva ”proof-of-concept” sovellus, joka tunnistaa energialaitoksen pihalta kuvaavan kameran videosta erilaisia objekteja. Työssä valmiiksi opetettu malli tunnistaa hyvällä tarkkuudella objekteja, jotka pystytään tallentamaan tietokantaan myöhempää käsittelyä varten. Työ osoitti, että tekoälyä ja objektin tunnistusta voidaan käyttää hyväksi laitoksen kunnonvalvonnassa ja tapahtumien kirjaamisessa. Laitokset voivat olla miehittämättömiä, joten tekoälyn tunnistuksen avulla voitaisiin tarkkailla laitosta etäyhteyden kautta. Tekoälyä on jatkuvasti kehittyvä ala, jota sovelletaan useissa eri osa-alueissa. Työssä keskityttiin objektin tunnistamiseen, mutta ideaa voidaan soveltaa myös muihin energialaitoksen alueisiin. Tekoälyä voitaisiin käyttää hyväksi esimerkiksi polttoaineen tunnistamiseen, jolloin tiedettäisiin polttoprosessin säätöön tarvittavaa tietoa. Tekoälyä voitaisiin myös hyödyntää polttoprosessin tarkkailuun polttopesässä, jolloin tekoäly voisi huomauttaa esimerkiksi polttoprosessin tilasta.

Aihe on itselleni hyvin mielenkiintoinen ja sen tutkiminen ja tekeminen oli itselleni mieluisaa. Aihe oli itselleni aivan uusi, joten epäilin itseäni, pystynkö toteuttamaan sovelluksen ja käyttämään tekoälyä hyväksi työssäni. Työn alussa ilmeni, että tekoälyyn pohjautuvia ratkaisuja pystyy myös aloittelijatason omaava henkilö tekemään. Työ koostui ongelmista, jotka sain kuitenkin nopealla ja päättäväisellä kyvyllä ratkaistua. Joissakin ongelma kohdissa en löytänyt tietoa suoraan internetistä, vaan jouduin itse selvittämään ja ratkaisemaan ongelman. En kohdannut kuitenkaan suurempia ongelmia ja työn tekeminen ilman suurempia ongelmia. Asetin työn alussa tavoitteeksi oman mallin opettamisen objektin tunnistukseen, joka kuitenkin jäi tekemättä. Kokeilin siirtää työssä käytettyä mallia Matlab-sovellukseen, joka kuitenkin osoittautui vaikeaksi, joten päätin jättää siirto-opetuksen pois työstä.

Työstä jäi käteeni laajempi käsitys, miten tekoälyä voidaan hyödyntää eri sovelluksissa ja ratkaisuissa. Näemme jatkuvaa kehitystä teknologiassa, ja väittäisin että tekoäly on yksi keskeisempiä asioita, jonka avulla voidaan helpottaa ihmiselämää sekä hyödyntää useassa eri osa-alueessa.

LÄHTEET

The MathWorks, Inc. (2021). *Object detection: The MathWorks, Inc.* Noudettu osoitteesta The MathWorks, Inc. sivusto: https://se.mathworks.com/discovery/object-detection.html

Achtman, B. (19. Toukokuu 2020). *Devblog: Microsoft Corporation.* Noudettu osoitteesta Microsoftin sivusto: https://devblogs.microsoft.com/dotnet/ml-net-model-builder-is-now-a-part-of-visual-studio/

Arnx, A. (13. Tammikuu 2019). *First neural network for beginners explained (with code)*. Noudettu osoitteesta https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

Arnx, A. (13. Tammikuu 2019). *towardsdatascience*. Noudettu osoitteesta https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

Campesato, O. (2020). *Artificial Intelligence, Machine Learning, and Deep Learning.* Mercury Learning & Information.

Chrislb. (6. Syyskuu 2010). *wikimedia*. Noudettu osoitteesta https://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetworkBigger_english.png

European parlament. (4. Syyskuu 2020). *What is artificial intelligence and how is it used?: European parlament.* Noudettu osoitteesta European parlamentin sivusto: https://www.europarl.europa.eu/news/en/headlines/society/20200827STO85804/what-is-artificial-intelligence-and-how-is-it-used

Gupta, N.;& Mangla, R. (2020). *Artificial Intelligence Basics: A Self-Teaching Introduction.* Mercury Learning & Information.

IBM ClIBM Cloud Education, IBM Cloud Education. (27. Toukokuu 2020). *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?* Noudettu osoitteesta https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Intel. (2021). Noudettu osoitteesta https://www.intel.la/content/dam/www/public/us/en/ai/images/ai-machine-learning-deep-learning-rwd.png.rendition.intel.web.480.270.png

KPA Unicon Oy. (2021). *Meistä: KPA Unicon Oy*. Noudettu osoitteesta https://www.kpaunicon.com/fi/meista/

Marr, B. (2019). *Artificial Intelligence in Practice : How 50 Successful Companies Used AI and Machine Learning to Solve Problems.* John Wiley & Sons, Incorporated.

Mechelli, A.;& Vieira, S. (2019). *Machine Learning: Methods and Applications to Brain Disorders.* Elsevier Science & Technology.

Microsoft. (4. Huhtikuu 2021). *Deep learning: Microsoft* . Noudettu osoitteesta Microsoftin sivusto: https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

Microsoft. (13. 4 2021). *Tutorial: Detect objects using ONNX in ML.NET*. Noudettu osoitteesta Microsoftin sivusto: https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/object-detection-onnx

Microsoft Corporation. (24. Toukokuu 2019). *An introduction to NuGet: Microsoft Corporation*. Noudettu osoitteesta Microsoft Corporationin sivusto: https://docs.microsoft.com/en-us/nuget/what-is-nuget

Microsoft Corporation. (13. Huhtikuu 2021). *Tutorial: Detect stop signs in images with Model Builder: Microsoft Corporation*. Noudettu osoitteesta Microsoftin dokumentti: https://docs.microsoft.com/fi-fi/dotnet/machine-learning/tutorials/object-detection-model-builder

Microsoft Corporation Oy. (2021). *Microsoft*. Noudettu osoitteesta Microsoftin sivusto: https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/media/object-detection-onnx/onnx-supported-formats.png

Microsoft Corporation Oy. (2021). *Microsoft*. Noudettu osoitteesta https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/media/object-detection-onnx/onnx-ml-net-integration.png

Newnham, J. (2018). *Machine Learning with Core ML : An IOS Developer's Guide to Implementing Machine Learning in Mobile Apps.* Packt Publishing, Limited.

Nvidia. (2021). *NVIDIA Developer Program*. Noudettu osoitteesta https://developer.nvidia.com/developer-program

Rafael C. Gonzalez, R. E. (2018). *Digital Image Processing.* New York: Pearson.

Roeder, L. (2021). *Netron: GitHub repo*. Noudettu osoitteesta https://github.com/lutzroeder/netron

Tazehkandi, A. A. (2018). *Hands-On Algorithms for Computer Vision.* Packt Publishing, Limited.

The MathWorks, Inc. (2021). *Deep Learning: The MathWorks, Inc*. Noudettu osoitteesta The MathWorks, Inc sivusto: https://se.mathworks.com/discovery/deep-learning.html

The MathWorks, Inc. (2021). *Machine Learning: The MathWorks, Inc*. Noudettu osoitteesta The MathWorks, Inc sivusto: https://se.mathworks.com/discovery/machine-learning.html

The MathWorks, Inc. (2021). *Neural Networks: The MathWorks, Inc*. Noudettu osoitteesta The MathWorks, Inc sivusto: https://se.mathworks.com/discovery/neural-network.htm

The MathWorks, Inc. (2021). *Artificial Intelligence: The MathWorks, Inc.* Noudettu osoitteesta The MathWorks, Inc sivusto: https://se.mathworks.com/discovery/artificial-intelligence.htm

Wehle, H.-D. (2017). Machine Learning, Deep Learning, and AI: What's the Difference? *Conference: Data Scientist Innovation Day.*