TEKOÄLY KUVANTUNNISTUKSESSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
kevät, 2021

Janne Puurtinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Hämeenlinnan korkeakoulukeskus

Tekijä Janne Puurtinen Vuosi 2021

Työn nimi Tekoäly kuvantunnistuksessa

Ohjaajat Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli tuottaa selvitys kahden kaupallisesti saatavan tekoälyjärjestelmän toiminnallisuudesta kuvantunnistuksessa. Tavoitteena oli tiedon lisääminen kuvantunnistusjärjestelmistä ja niiden mahdollisuuksista käyttämällä ja vertailemalla Googlen ja Microsoftin tekoälypalveluita. Opinnäytetyön toimeksiantaja oli Hämeen Ammattikorkeakoulun Smart-tutkimusyksikkö.

Opinnäytetyön teoriaosassa käydään läpi lyhyesti tekoälyn historiaa ja sen jälkeen avataan kuvantunnistukseen liittyviä aihealueita sekä tärkeimpiä käsitteitä. Opinnäytetyön toiminnallisessa osassa ohjelmoidaan sovellukset, joiden avulla kuvantunnistustoimintoja selvitetään tarkemmin. Sovelluksilla kutsutaan Googlen ja Microsoftin tekoälypalveluita rajapinnan kautta sekä niillä kerätään ja tallennetaan tunnistustuloksia. Kuvantunnistustoimintoja testattiin erilaisilla tilannekuvilla, joiden avulla saatiin kerättyä tietoa tekoälyjen toimivuudesta. Lopuksi tuloksia tutkitaan ja vertaillaan palveluiden kesken.

Työn tuloksena saatiin tietoa Googlen ja Microsoftin tekoälyjen kyvystä kuvantunnistuksessa ja toimintojen eroavaisuuksista. Tuloksista selvisi, että Googlen tekoäly oli kokonaisuutena tarkempi havaitsemaan ja tunnistamaan erilaista tietoa kuvasta. Kuvantunnistustoimintojen ominaisuuksissa oli kuitenkin myös eroja palveluiden kesken, eikä kaikkea havaittua kuvatietoa voinut välttämättä verrata keskenään.

Avainsanat Tekoäly, konenäkö, tietokonenäkö, kuvantunnistus

Sivut 43 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Abstract

Author Janne Puurtinen
Year 2021
Subject Artificial intelligence in image recognition
Supervisors Lasse Seppänen

ABSTRACT

The purpose of the thesis was to find out the functionality of two commercially available artificial intelligence (AI) systems in image recognition. The goal was to increase knowledge on image recognition systems and their potential by using and comparing Google´s and Microsoft´s AI-services. The thesis was commissioned by the Smart Research Unit of Häme University of Applied Sciences.

The theoretical part of the thesis briefly reviews the history of AI and then opens the topics and concepts related to image recognition. In the functional part of the thesis, applications are programmed to be used in clarifying image recognition functionalities. The applications make queries via interface to Google´s and Microsoft´s AI services and collect and save the results of recognition. Image recognition functions were tested with various snapshots, which were used to collect information about the functionality of AI. Finally, the results are examined and compared between services.

The work provided information on the capabilities of Google´s and Microsoft´s artificial intelligence in image recognition and differences in functionalities. The results showed that Google´s artificial intelligence was more accurate in detecting and identifying different information about an image. However, there were also differences in the features of image recognition functions between the services, and not all the image information could be compared with each other.

Keywords Artificial intelligence, machine vision, computer vision, image recognition
Pages 43 pages and appendices 1 page

Sanasto

AI	Artificial Intelligence, tekoäly
Algoritmi	Suoritusohjeita ja menetelmiä, jotka tyypillisesti ovat erilaisia matemaattisia laskutoimituksia
API	Application Programming Interface, ohjelmointirajapinta
Konenäkö	Erikoistunut tunnistusjärjestelmä, usein teollisiin tarkoituksiin
Koneoppiminen	Tekoälyn osa-alue, jossa kone oppii itsenäisesti tiedon perusteella
Kuvantunnistus	Termi korkeatasoisesti kuvaa tulkitseville tekniikoille ja metodeille
Neuroverkko	Luonnollisia hermoverkkoja mukaileva laskentamalli
Ohjaamaton oppiminen	Koneoppimismalli oppii ilman merkittyä tietoaineistoa
Ohjattu oppiminen	Koneoppimismalli oppii merkityillä tietoaineistoilla
Ohjelmointirajapinta	Mahdollistaa tietojenvälityksen erilaisten sovellusten välillä
Syväoppiminen	Koneoppimismenetelmä, jossa hyödynnetään neuroverkkoja
Tekoäly	Ihmisen päättelyä ja toimintoja matkiva tietokonejärjestelmä
Tietokonenäkö	Yleistermi ihmismäistä havainnointia jäljitteleville tekniikoille

Sisällys

Kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Usein toistuvia toimintoja sekä töitä on tekniikan kehittyessä pystytty automatisoimaan, jotta tuloksia saataisiin tehokkaammin, nopeammin sekä laadukkaammin. Tietokoneiden laskutehon kasvu on mahdollistanut luoda inhimillistä päättelyä jäljittelevää tekoälyä, jota nykyään käytetään hyvin moneen eri käyttötarkoitukseen, esimerkiksi teollisuuteen, lääketieteeseen ja liikenteeseen. Tekoälyä hyödynnetään yleensä helpottamaan työntekoa tai vähentämään ihmiseltä vaadittavaa työmäärää, joissakin tapauksissa se saattaa poistaa ihmisen osuuden kokonaan. Tekoälyä pystytään opettamaan erilaisilla menetelmillä, mistä puhutaan yleisesti termillä koneoppiminen. Tekoälyn mahdollisuuksista ja hyödyistä kuullaan puhuttavan yhä useammin ja sitä pidetään yhtenä merkittävimpänä teknologian alueena viime vuosikymmenenä ja tulevina aikoina.

Tämän opinnäytetyön aiheen lähtökohtana on Hämeen Ammattikorkeakoulun Smart-tutkimusyksikön kiinnostus kaupallisesti saataviin kuvantunnistuspalveluihin, sekä niiden käyttömahdollisuuksiin. Aihe valikoitui omasta mielenkiinnosta tekoälyä kohtaan, koska koin sen olevan erittäin ajankohtaista eikä opinnoista saatu käytännön tieto tuntunut riittävältä. Aihe sopi mielestäni myös hyvin sovelluskehittäjän opintojen kehykseen.

Toimeksiantaja on rajannut tutkittaviksi kaksi tekoälypalvelua, Microsoft Azure Cognitive Services sekä Google Vision API. Tarkoitus on selvittää kokeilemalla, miten näitä palveluita käytetään kuvantunnistukseen. Työn aikana tekoälyillä tutkitaan materiaalina kuvia ja tarkastellaan mitä tietoa tekoäly löytää näistä. Opinnäytetyön lopussa saatuja tietoja tarkastellaan ja vertaillaan palveluiden kesken.

Opinnäytetyön tutkimuskysymykset ovat:

- Mitä kuvantunnistustoimintoja on käytettävissä ja millä tarkkuudella ne toimivat?
- Mitä parametrejä ja tietoa tekoäly pystyy tunnistamaan videosta ja kuvasta?
- Miten Googlen ja Microsoftin tekoälyä käytetään kuvantunnistuksessa?
- Tunnistaako tekoäly luotettavasti jotain tiettyjä tapahtumia?
- Miten ko. tekoälyjen tulokset eroavat?

2 Tekoäly

Teoriaosassa avataan ensin tekoälyä yleisellä tasolla ja historian kautta sekä tuodaan esiin muutamia tekoälyn osa-alueita ja käytännön esimerkkejä. Tekoäly ja sen määrittely tieteenalana sisältää valtavan määrän käsitteitä, teoriassa keskitytään opinnäytetyöhön ja sen toiminnalliseen osaan liittyviin peruskäsitteisiin.

Termi tekoäly juontuu englannin kielen sanasta Artificial Intelligence, mistä käytetään usein lyhennöstä AI. Sanakirja Merriam-Webster (2017) mukaan tekoäly tarkoittaa joko sitä tietojenkäsittelytieteen alaa, jossa näitä asioita käsitellään tai sitten sitä prosessia, missä tietokoneet simuloivat älykästä käyttäytymistä. Älykkyyden käsite tekoälyssä on huomattavasti laajempi kuin mitä sen yhteydessä yleensä ajatellaan. Pelkästään älyllisyys on jo käsitteenä erittäin monimutkainen ja monitahoinen kokonaisuus. Tekoälyssä voidaan ajatella esiintyvän ihmisiin verrattavan älykkyyden lisäksi myös eläimissä ja asioissa esiintyviä älykkyyden muotoja. Esimerkiksi eläinten eloonjäämistaitoja ja kykyä kommunikoida lajitoverien kanssa voidaan pitää älyllisenä käyttäytymisenä. Ihmisen älykkyys on seurausta aivotoimintamme mentaalisesta prosessoinnista. Omaamme vain rajalliset kyvyt aistia ympäröivää maailmaa, emme esimerkiksi voi havaita kaikkia äänentaajuuksia tai säteilyä. Tekoälyä omaavan koneen tai järjestelmän älykkyyttä määritellessä tulisi näin ollen harkita muitakin älykkyyden näkökulmia kuin inhimillisyyteen verrattavia piirteitä. (Warwick, 2013)

Tekoäly on laajalti käytetty yleiskäsite erilaisille tietokonejärjestelmille, eikä sille ole aivan tarkkaa määrittelyä. Tarkempaa määrittelyä monimutkaistaa se, että nykyään tekoälyllä viitataan säännöllisesti mihin tahansa tekniikkaan, kunhan sen väitetään omaavan älykkääksi toiminnaksi kuvailtavia ominaisuuksia. Useimmiten sillä kuitenkin viitataan järjestelmien kykyyn pystyä ihmismäisiin taitoihin, kuten päättelyyn, oppimiseen, tiedon käsittelyyn tai havainnoimiseen. Määrittely on muuttunut myös ajan myötä ja nykyään tekoälyn käsitetään myös kykenevän itsenäiseen toimintaan jollain tasolla. (Boucher, 2020)

2.1 Tekoälyn historia

Tekoälyn määrittelyn voidaan katsoa alkaneen Alan Turingin esittäessä kysymyksen, voivatko koneet ajatella. Vuoden 1950 artikkelissaan Computing Machinery and Intelligence hän ehdotti testiä, jolla voitaisiin määritellä onko kone älykäs. Kokeessa ihminen asetettaisiin kommunikoimaan koneen kanssa viestien välityksellä. Jos ihminen ei pystyisi päätellä onko vastaajana ihminen vai kone, olisi se älykkään käyttäytymisen määritelmä. Testissä ihminen kommunikoi pelkin viestin, koska Turingin mukaan fyysinen vuorovaikutus ei ole olennaista älykkyyden määrittelyssä. (Joshi, 2017)

Termi tekoäly esiintyi ensimmäistä kertaa vuonna 1955, kun neljä tietojenkäsittelytieteen tutkijaa ehdotti järjestettäväksi Dartmouth Summer Research Project on Artificial Intelligence -nimistä työpajaa Darthmouthin yliopistolla määrittelemään tekoälyä. Tapahtuman perustajajäseniin kuului matematiikan professori John McCarthy, matematiikan kandidaatti Marvin Minsky, sähkötekniikan ja matematiikan maisteri Claude Shannon sekä sähköinsinööri Nathaniel Rochester. Tarkoituksena oli koota ryhmä asiantuntijoita luomaan perusta uudelle kenttätutkimukselle sekä alkaa tutkimaan ja määrittelemään miten tekoälyä voitaisiin ohjelmoida käytettäväksi koneissa. Konferenssi pidettiin seuraavana vuonna 1956 ja osallistujia tuli lukuisilta eri tieteen aloilta, kuten tietojenkäsittelytieteestä, sähkötekniikasta ja psykologiasta. Konferenssilla ja siihen liittyineillä henkilöillä oli tulevina vuosina merkittävä vaikutus erilaisiin tekoälyä soveltaviin aloihin. (McCarthy et al., 2006; Veisdal, 2019)

Darthmouthin konferenssia pidetään tekoälyn tutkimuskentän aloittajana. 1950-luvulla kehiteltiin ensimmäisiä tekoälyyn pohjautuvia järjestelmiä, joita yhdisti heuristiikka eli ongelmanratkaisu käyttäen valmiita ratkaisumalleja. Vuonna 1956 Arthur Samuel kirjoitti maailman ensimmäisen itseoppivan ohjelman, missä tekoäly osasi löytää parhaimman siirtoliikkeen tammea pelatessa. Samuel tuli keksineeksi samalla termin Machine Learning, eli koneoppiminen, esitellessään ohjelmansa vuonna 1959. Vuoden 1956 konferenssin perustajiin kuulunut John McCarthy keksi Lisp -ohjelmointikielen vuonna 1958, mikä pystyi käsittelemään numeroiden lisäksi myös symboleja. Tekoälyn tutkiminen keskittyi vielä 1960-luvun alussa heuristiikkaan ja algoritmeihin, eli loogisiin toimintamalleihin, kunnes siirryttiin tiedon kuvailumenetelmiin (Knowledge representation) ja kielen tunnistukseen (Natural-

language understanding). Vuosina 1965-1970 kehitettiin kemikaalien molekyylirakenteita analysoiva DENDRAL-järjestelmä, joka oli ensimmäinen toimialalle kohdennettu asiantuntijajärjestelmä (Expert system). Asiantuntijajärjestelmä on nimitys erikoisalan monimutkaisia ongelmia ratkovalle tietokoneohjelmalle, joka käyttää asiantuntijatason tietoa omaavaa tekoälyä päätöksentekoon (Guru99, n.d.). 1980-luvulla erilaiset asiantuntijajärjestelmät yleistyivät huomattavasti ja tekoälyteknologian soveltaminen teollisuuden toimintoihin alkoi kasvaa menestyksekkäästi, mikä herätti myös valtiollisen kiinnostuksen tekoälyn kehittämiseen. (Veisdal, 2019; Zhongzhi, 2011)

Varhaiset tekoälymallit pohjautuvat ns. klassiseen tekoälyyn, missä jäljitellään yleisiä ihmisen toiminnan ja käyttäytymisen malleja. Tämänkaltaiset järjestelmät nojaavat logiikkaan, valmiisiin ratkaisumalleihin ja päättelysääntöihin, esimerkiksi ehtolausekkeisiin (if….then). Viimeaikoina tekoälyn tutkimus on syventynyt erilaisiin algoritmitekniikkoihin, jotka mallintavat neuroverkkoja ja genetiikkaa. Näissä mallina toimivat luonnossa esiintyvät biologiset järjestelmät, joissa yksinkertaiset tiedonkäsittelijät, neuronit eli hermosolut, muodostavat toisiinsa kytkeytyneen verkoston. Aivojen neuroverkkojen toimintatapa on perusta aivojen kehitykselle, toiminnalle ja kasvulle. Neuroverkot ohjaavat aivotoimintaa ja osaavat esimerkiksi mukautua reagoimaan erilailla perustuen kokemuksen laatuun. Keinotekoiset neuroverkot on todettu erittäin tehokkaaksi ja mukautuvaksi tekoälyksi. (Reaktor & Helsingin Yliopisto, n.d.; Warwick, 2013)

Tekoälyn osa-alueet keskittyvät nykyään koneiden automaattiseen oppimiseen sekä päättelyyn koneoppimismallien kautta, kykyyn ymmärtää ja käsitellä kieltä (tekstiä ja puhetta), sekä visuaalisen tiedon prosessointiin. Näitä osa-alueita sovelletaan esimerkiksi robotiikkaan ja asiantuntijajärjestelmiin, joissa tekoälyjärjestelmät suoriutuvat tehtävistä itsenäisesti sekä osaavat olla vuorovaikutuksessa sosiaalisin keinoin. (JavaTpoint, n.d.).

2.2 Koneoppiminen

Koneoppiminen on yksi tekoälyn aihepiireistä ja useimmin esiintyvä tekoälyn muoto. Koneoppimisessa ideana on koneen itsenäinen oppiminen ja päätösten teko. Monimutkaisia algoritmejä käyttäen koneelle pyritään opettamaan automaattista tiedon tulkintaa ja laajentamaan sen havainnointikykyä. (Siukonen & Neittaanmäki, 2019)

Oppimisprosesseissa koneelle syötetään tietoa oppimisympäristöstä. Koneen oppimisesta vastaava yksikkö muuntaa tiedon konekieleen ymmärrettävään muotoon ja tekee päättelyä käyttäen tietoon vastaavia algoritmejä. Ennen kuin tieto tallennetaan tietämyskantaan, tiedon kuvailumenetelmät käsittelevät ja jäsentävät sen tiettyjen muodollisten ominaisuuksien säännöin. Tietämyskanta (Knowledge Base) on tietokantaa tarkempi järjestelmä, mikä sisältää asiatietoa, faktoja, merkityksiä, päättelymalleja ja -sääntöjä sekä menettelytapoja. Tietämyskanta toimii logiikan mukaan, jossa uusia faktoja päätellään jo tiedettyjen faktojen perusteella. Samaan logiikkaan perustuen se voi myös laajentua uusista tiedoista, huolimatta päätellyn tiedon luotettavuudesta. Päättelyn ja ennustamisen kautta järjestelmä hankkii ymmärrystä ja kehittää älykkyyttä sekä havainnointikykyä. (Sanastokeskus TSK ry, 1993; Zhongzhi, 2011)

Koneoppiminen jaetaan yleisesti kahteen eri tyyppiseen oppimismalliin perustuen tekoälyn algoritmien käytettävissä olevaan tietoon: ohjattuun ja ohjaamattomaan oppimiseen. Ohjatussa oppimisessa (Supervised) tekoälyllä on käytettävissä oppimismateriaalina tietojoukko (Dataset), tai kokoelma tietojoukkoja, sisältäen luokiteltujen ja nimettyjen (Labeled) parametrien tietokannan. Näiden parametrien avulla tekoälyn algoritmit osaavat tehdä tietojoukosta päätelmiä ja ennakoida lopputuloksia. Oppimisprosessin aikana mallia korjataan virhepäätelmistä ja prosessia jatketaan niin kauan kunnes tekoäly saavuttaa halutun tarkkuuden opeteltavasta materiaalista (Brownlee, 2019).

Ohjaamattomassa oppimisessa (Unsupervised) oppimismateriaalina on luokittelematon tai nimeämätön (Unlabeled) tietojoukko. Tekoälyn algoritmien tehtävä on päätellä parhaimman mukaan tietojoukon rakenteesta yleisiä sääntöjä, esimerkiksi järjestämällä tietoa samankaltaisuuden tai toistuvien piirteiden kautta. (JavaTpoint, n.d.; Joshi, 2017)

2.3 Algoritmi

Algoritmilla tarkoitetaan yksityiskohtaisia, vaiheittain suoritettavia suoritusohjeita ja menetelmiä, jotka tyypillisesti ovat erilaisia matemaattisia laskutoimituksia. Näiden avulla algoritmien tarkoitus on ratkaista jokin ongelma tai saavuttaa tietty päämäärä, kun sille syötetään reaalimaailmasta saatua dataa. Jotta tekoälyllä voidaan suorittaa yhä monimutkaisempia toimintoja, algoritmejä voidaan myös yhdistää käytettäväksi vaiheittain.

Erilaisten algoritmien kirjo on erittäin suuri, mutta ne voidaan jakaa karkeasti yleisimpiin pääluokkiin perustuen ratkottavien tehtävien samankaltaisuuteen. (Fry & Iso-Markku, 2019)

Suodattamisalgoritmien (Clustering) tehtävä on erottaa tai eristää asioita annetusta tiedosta, tarkoituksena yleensä löytää samankaltaisuuksia muiden tietojen kesken. Nämä järjestelmät toimivat ohjaamattoman mallin tavalla etsiessä parasta tulosta: sisältöperusteisesti tai yhteistoiminnallisesti. Sisältöpohjaisessa (Content-based filtering) tarkoitus on löytää yhteneväisyyksiä annetun kuvauksen, esimerkiksi käyttäjäprofiilin tai tuotteen ominaisuuksien, ja tietojoukon välillä. Järjestelmä luo kuvauksesta ja tietojoukosta ominaisuuslistan, mitä vertaamalla ehdotetaan parhaiten sopivia tuloksia, esimerkiksi suositellakseen jotakin tuotetta tai palvelua . Yhteistoiminnallisessa (Collaborative filtering) ennustetaan tulosta perustuen muiden käyttäjien antamiin mieltymyksiin ja suositteluihin. Erilaiset hakukoneet ja suosittelujärjestelmät käyttävät suodattamisalgoritmejä etsiessä käyttäjälle olennaista tietoa tai tuotettatuotetta. (Fry & Iso-Markku, 2019; Kirzhner, 2018)

Luokittelualgoritmit (Classification) perustuvat opetettuun oppimismalliin, jossa valmiiden tietojoukkojen pohjalta yritetään ennustaa tai luokitella tietoa. Luokittelun tuloksena syntyy yleensä diskreettiä dataa, eli lueteltavia tietoja kuten sukupuoli, tosi/epätosi tai lukumäärä. Tietyissä tilanteissa tulos voi myös olla muuttuja kuten todennäköisyysluku, mikä voidaan muuttaa luokka-arvoksi (Lee, 2019). Tyypillisiä esimerkkejä luokittelusta ovat erilaiset tilasto- ja statistiikkaennusteet ja todennäköisyyslaskenta. Yksi yleinen käytännön esimerkki luokittelevasta sovelluksesta on roskapostisuodatin. Hyvin samankaltaisena algoritmimallina pidetään regressioalgoritmejä (Regression), joissa tietojoukosta ennustetaan numeerisia muuttujia, kuten hintaa tai lämpötilaa. Regression avulla voidaan tehdä esimerkiksi sääennusteita tai hintakehitystä. (Fry & Iso-Markku, 2019; JavaTpoint, n.d.; Tilastokeskus, n.d.)

Yhdistämisalgoritmien (Association rule) tehtävä on etsiä tietojen välisiä yhteyksiä sekä sidoksia laajoista tietokannoista ja luoda niistä yleistettäviä sääntöjä. Tieto perustuu opetettuihin tietojoukkoihin, joista löydettyjen sidossääntöjen mukaan ehdotetaan sopivia vastaavuuksia. Yhdistämisalgoritmejä sovelletaan yleensä tehokkaasti kaupallisiin tarkoituksiin, kuten tuotesuosituksiin ja -ehdotuksiin. (Fry & Iso-Markku, 2019; Garg, 2018)

Priorisointialgoritmi (Priority scheduling) käyttää matemaattisia ratkaisumalleja ennustaakseen opetetuista tietojoukoista halutun prioriteetin mukaisen tuloksen. Järjestelmä ennustaa joukon tuloksia ja luo niistä vaihtoehtovalikoiman ongelman ratkaisemiseksi. Valikoimaa yritetään järjestää sopivaksi eri tilanteisiin, tilanne kerrallaan. Tämänkaltaista algoritmia hyödyntävät erilaiset navigointijärjestelmät, missä haetaan esimerkiksi nopeinta tai lyhintä reittiä. (Fry & Iso-Markku, 2019)

2.4 Tietokonenäkö

Tietokonenäkö (Computer vision) on monitahoinen ja useita tieteenaloja yhdistävä tekoälyn osa-alue, mikä tekee siitä todella monimutkaisen termin. Tieteen näkökulmasta sillä tarkoitetaan tekoälyjärjestelmän teoriapohjaa mikä liittyy tiedon erottamiseen kuvista. Teoriapohjaan sisältyy lukuisia luonnontieteen alueita, kuten fysiikka, matematiikka, biologia ja tekniikka, jotka tarjoavat erilaisia lähestymistapoja ihmisen näkökyvyn saavuttamiseen ja siihen liittyvien tekniikoiden ja metodien suunnitteluun. Tietokoneälyllä ei ole oppikirjamaista määrittelyä laajan ja risteävien tieteenalojen ja teknologioiden vuoksi. Sen voidaan ajatella liittyvän kaikkeen visuaalista dataa käsitteleviin algoritmeihin, metodeihin ja teknologioihin. Keskeisessä asemassa on mahdollistaa koneelle kykyä nähdä, havaita ja ymmärtää ympäröivää maailmaa. (Devopedia, 2020; Tazehkandi, 2018)

Tietokonenäön osa-alueet keskittyvät tietynlaisiin käsittely- ja tunnistustehtäviin ja ne jakavat usein samankaltaisuuksia keskenään tekniikoissa ja algoritmeissa. Keskeisimpiä osa-alueita ovat konenäkö, kuvankäsittely (Image processing) ja kuvantunnistus (Image recognition). Kuvankäsittelyn tekniikat keskittyvät digitaalisen kuvan muodostamiseen ja muokkaamiseen niin, että siitä voidaan havaita asioita. Kuvantunnistus viittaa digitaalisesta kuvasta tai videosta saatavan kohteen tai ominaisuuden havaitsemiseen ja tunnistamiseen. Aiemmin tietokone- ja konenäöllä viitattiin yleisemmin kuvankäsittelyn prosessien metodeihin ja tekniikoihin. Moderni näkemys määrittelee tietokonenäön yleiskäsitteeksi ihmisnäköä automatisoiville ja mahdollistaville prosesseille, mitkä pystyvät tunnistamaan, käsittelemään ja kuvailemaan asioita kuvasta. Tietokonenäön osa-alueiden jaottelu on joissain määrin subjektiivista asiantuntijoiden kesken ja termejä käytetään edelleen paljon ristiin. (Appen, 2019; Tazehkandi, 2018; Tryolabs, n.d.)

Kone- ja tietokonenäköjärjestelmien toiminta perustuu yleisesti ottaen tulkitsemaan havaittua kuvainformaatiota käyttäen samoja tai samankaltaisia tekniikoita ja metodeja, mutta ne voidaan erottaa käyttötapauksiensa mukaan. Koneälyssä toiminnan ydin on kuvan hankinnassa ja prosessoinnissa, ja sitä käytetään tyypillisesti teollisissa ympäristöissä osana isompaa prosessia. Tietokonenäkö keskittyy yksityiskohtaisemmin ja korkeatasoisemmin kuvainformaation havainnointiin, tulkintaan, sekä ennustamiseen. Tietokonenäön tavoitteena on kuvailla ja tulkita kuvasta tunnistettuja asioita mahdollisimman monipuolisesti ja antaa syvempää ymmärrystä, mikä vaatii huomattavasti enemmän opetettua tietoa kuin koneälyjärjestelmät. Tietokonenäkö käsittelee yleensä valmiiksi saatavilla olevaa digitaalista kuvatietoa ja se pystyy usein toimimaan itsenäisesti omana järjestelmänä, mihin tulee vain liittää ulkoinen kaksi- tai kolmiulotteinen kuvanlähde. Koneälyjärjestelmät puolestaan vaativat toimiakseen kameran, prosessorin ja ohjelmiston kokonaisuuden. Tietokonenäköä ja sen tekniikoita käytetään usein konenäköjärjestelmien yhteydessä kuvien tulkitsijana, mahdollistaen monimutkaisempia ja älykkäämpiä koneita käytettäväksi muuallakin kuin teollisuuden toiminnoissa. (Appen, 2019; Clearview Imaging, 2018; Hornberg, 2017)

Tietokonenäköä käytetään nykypäivänä hyvin laajalti ja onnistuneesti eri toimialoilla ja käytössä olevia sovelluksia löytyy valtavasti niin teollisuuteen kuin kuluttajatason sovelluksiin. Tuotantoteollisuuden yleisimpiä käyttökohteita ovat tuotantoprosessien tarkistus ja laadunvalvonta. Autoteollisuus on hyödyntänyt tietokonenäköä erilaisiin turvallisuutta ja kuljettajaa avustaviin toiminnallisuuksiin, kuten kaistoja, esteitä ja vaaroja havaitsevia järjestelmiä, itseajavia autoja tietenkään unohtamatta. Lääketieteen tekniikassa on käytössä monenlaisia kuvantamisjärjestelmiä, millä voidaan tunnistaa esimerkiksi syöpäkudoksia ja kasvaimia. Kasvojen- ja hahmontunnistus on laajalti käytössä niin kauppojen kuin puolustus- ja turvallisuusalan valvontajärjestelmissä. Kiinan on uutisoitu käyttävän kasvojentunnistusta julkisilla paikoilla valvomaan lähes 1.4 miljardia kansalaista, pitääkseen yllä järjestyslakeja. (Ng, 2020; Tazehkandi, 2018; Tryolabs, n.d.)

Tietokonenäköä hyödyntäviä sovelluksia löytyy myös nykyään älypuhelimista. Google julkisti hiljattain sykkeen ja hengitystiheyden monitoroinnin, mikä käyttää älypuhelimen kameraa seuratakseen rintakehän liikkumista ja sormenjälkilukijaa havaitakseen sormen pään verenkierron värimuutoksia. (Wetsman, 2021)

2.4.1 Perinteiset ja modernit tekniikat

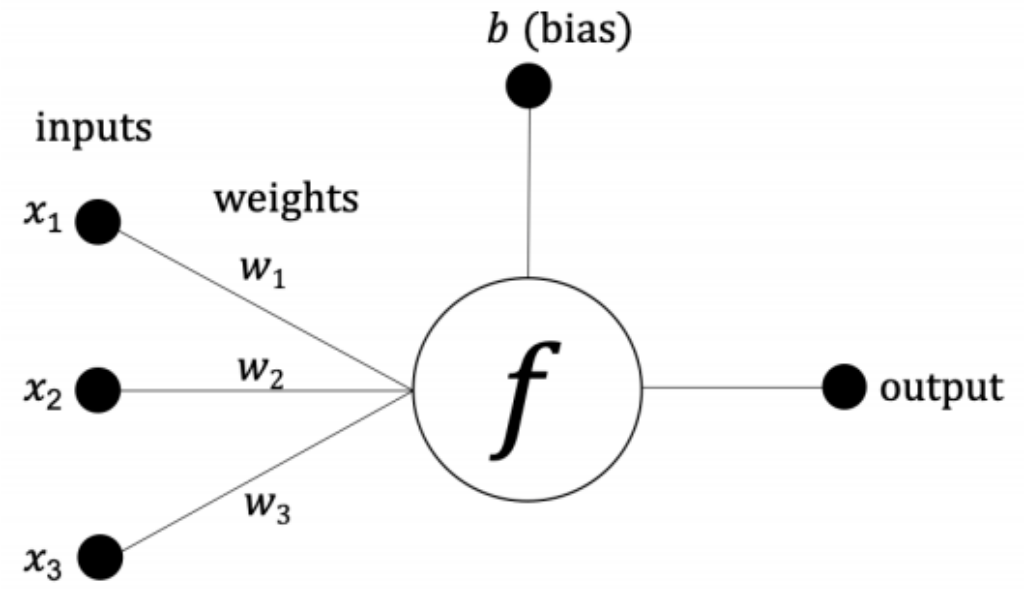
Tietokoneälyn kehityshistoriassa algoritmien metodit ja tekniikat ovat yrittäneet ratkaista yhä monimutkaisempia visuaaliseen havainnointiin ja ymmärrykseen liittyviä tehtäviä. Yleisin haaste kuvainformaation tunnistuksessa on kuvatiedon monimutkaisuus. Kuva saattaa sisältää satoja eri asennoissa olevia objekteja ja ne voivat olla vain osittain näkyvissä. Tietokonenäköä alettiin kehittämään 1970-luvun alkupuolella kohti kuvan maiseman (Scene) eli kuvasisällön ymmärtämistä kaksi- ja kolmiulotteisessa muodossa. Varhaiset ns. perinteiset tekniikat ja metodit perustuvat loogisiin ja kaavamaisiin päättelymalleihin (Pattern recognition/classification), jotka vaativat yleensä tarkkoja ominaisuusrakenteiden määritelmiä ja laatimisia. Kaiken havaittavan tiedon ennalta määritteleminen ja opettaminen algoritmien tunnistettavaksi on kuitenkin erittäin työlästä. Yhä vaikeampien tunnistustehtävien ratkaisemiseksi alettiin soveltamaan koneoppimista ja oppimistietokantoja, joilla kyettiin siirtämään ja automatisoimaan manuaalinen kuvatiedon opetus koneen tehtäväksi. Tietokoneiden tehon, muistin ja tallennustilan kasvu on mahdollistanut luoda algoritmeille laajoja opetettavia tietokantoja sekä kehittyneempiä koneoppimismetodeja. (Prince, 2012; Richmond, 2020; Szeliski, 2009)

Perinteisten tekniikoiden tunnistusmallit perustuvat muotojen hahmottamiseen sekä tilastollisiin todennäköisyyksiin, eli vertaavat kuvatietoa ennalta tiedettyihin ominaisuuksiin ja piirteisiin. Nämä tekniikat ovat omiaan automatisoiduissa kuvatehtävissä, missä tiedetään tarkasti kuvasta saatava tiedon laatu ja tyyppi, sekä niiden kuvausolosuhteet ovat kontrolloituja. Tästä esimerkkinä erilaiset teollisten prosessien objektien, tekstien ja viivakoodien luku, tai vaikkapa sormenjälkien tunnistustehtävät. Koneoppimisen tavoitteena kuvantunnistuksessa on siirtyä tilastollisista malleista kohti matemaattiseen todennäköisyyteen ja sen optimointiin kuvatiedon mittauksessa ja tunnistuksessa. Tämänkaltaisessa tunnistusmallissa parhaan tuloksen ennustaminen ei perustu tiedettyihin todennäköisyyksiin, vaan koneen on luotava uusi päättelymalli opetetusta tietoaineistosta ja pääteltävä todennäköisin tulos päättelymallin vaihtoehdoista. (Davies, 2017; Kenton Will, 2020)

Koneoppimista ja erilaisia neuroverkkoja mallintavia toimintatapoja yhdistäviä tekniikoita kutsutaan yleisemmin nimellä syväoppiminen (Deep learning). Syväoppimisen yleistyminen

vasta viime vuosikymmenen aikana on johtunut siinä vaadittavan opetusmateriaalin paljoudesta, sen saatavuudesta, sekä laskentatehon vaativuudesta. Syväoppimisen ideana on luoda ennustusmalliksi keinotekoinen neuroverkko, jota on mahdollista opettaa tiedostoaineistoilla. Malli on käytännössä matemaattinen funktio, joka saa sisääntulona (input) dataa ja antaa lähtönä (output) ennusteen (Kuva 1). Data voi olla kuvia, videoita, tekstiä tai vaikkapa ääntä. Opetusvaiheessa mallia harjoitetaan syöttämällä sille sisääntulona tietoaineistoa, josta lasketaan jokaisen tiedon painokerroin (Weight) sekä lopuksi painoarvojen summa. Tämän summan arvo syötetään neuronille, eli ns. aktivaatiofunktiolle, joka luokittelee arvon välillä 0–1 ja antaa tämän lähtönä. Syntyy siis todennäköisyysarvo mikä voi olla esimerkiksi Kyllä tai Ei. Jos arvo on tarpeeksi vahva, aktivoituu neutroni ja arvo voidaan lähettää seuraavaan neutroniin uutta luokittelua varten. Neuroverkko koostuu useammasta toisiin kytketyistä aktivointifunktioista muodostaen monitasoisen verkostoidun järjestelmän. Opettamisen aikana tuloksia tutkitaan ja neuronien aktivoitumisherkkyyttä säädetään sovittamalla (Bias) niiden painokertoimia, kunnes ennustusmalli on riittävän tarkka. (Kananen & Puolitaival, 2019; Koul et al., 2019; Richmond, 2020)

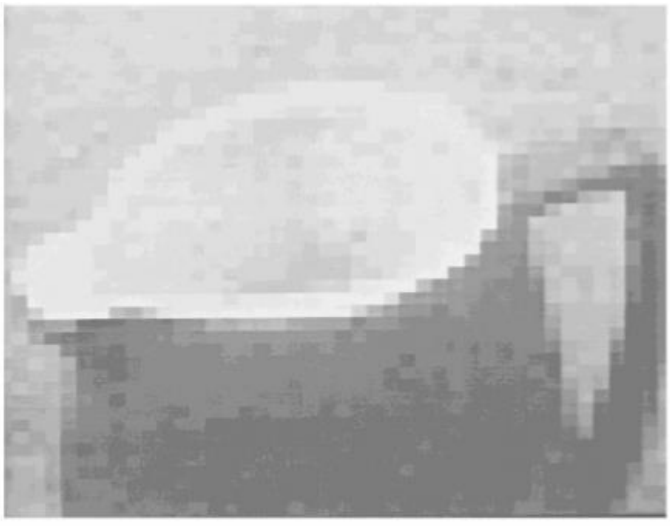
Kuva 1 Yksittäisen neuronin eli aktivointifunktion toimintaperiaate (Koul et al., 2019)



2.4.2 Kuvankäsittely

Kuvankäsittely (Image processing) viittaa prosesseihin, joiden tarkoitus on helpottaa kuvainformaation tulkitsemista erottamalla asioita kuvasta. Kuvankäsittelyllä tarkoitetaan myös kuvatiedon muuntamista digitaaliseen muotoon, jolloin sitä voidaan käsitellä ja esittää kaksi- ja kolmiulotteisesti. Kuvankäsittelyllä katsotaan olevan yhtä merkittävä osa järjestelmän tehokkuuteen kuin kuvaa tulkitsevilla prosesseilla. (Chakraborty, 2019; Szeliski, 2009). Ennen kuin kuvainformaatiota voidaan tulkita, täytyy kamerasta tai sensorista saatu analoginen kuvasignaali muuntaa tietokoneelle käsiteltävään digitaaliseen muotoon. Jokaisen yksittäisen kuvapikselin kirkkauden arvo tallennetaan binäärinumeroina (lukujen 0-256 arvoväli) kuvamatriisiin eli taulukkoon, yleensä harmaasävyinä (Kuva 2). Kuvan tulkinnassa harmaasävy riittää usein käytettäväksi, jos väri ei ole olennaista informaation tulkinnassa. Värikuvat helpottavat kuvan tunnistusta tapauskohtaisesti, mutta vaativat järjestelmältä enemmän tallennustilaa ja tehoa. Värikuvia esittäessä luodaan vastaavat matriisit erikseen punaisen, vihreän ja sinisen kirkkauden arvoista ja lopuksi ne yhdistetään. Usein kuvanlaatu ei ole sellaisenaan riittävä ja sitä joudutaan ehostamaan kuvankäsittelymetodein. Erilaisia suodattavia ja ehostavia algoritmeja käytetään esimerkiksi parantamaan terävyyttä, vähentää kohinaa ja säätämään valaistuksesta riippuvia tekijöitä kuten kirkkautta tai kontrastia. Segmentoinnilla kuva voidaan rajata olennaista tietoa sisältäviin osa-alueisiin, jonka jälkeen niistä voidaan erottaa tietoa sisältäviä ominaisuuksia, kuten muotoja, tekstiä tai kuvioita. Algoritmit osaavat myös muuntaa kuvan perspektiivin vääristymiä. (Beyerer et al., 2015; Hornberg, 2017; Warwick, 2013)

Kuva 2 Digitaalinen kuvapikseleistä muodostettu kuva (Davies, 2017)



Reunantunnistus (Edge detection) oli ensimmäisiä tekniikoita luoda maisemaa ymmärtäviä algoritmeja. Nämä algoritmit tunnistavat kuvapikselien jyrkkiä kirkkauden vaihteluita (epäjatkuvuuksia) ja muodostavat reunalinjoja yhdistämällä näitä kohtia. Reunalinjoista muodostetusta ominaisuudesta voidaan mitata ja päätellä erilaisia kuvaominaisuuksia, kuten muutoksia pinnan syvyydessä, suunnassa tai materiaalin muutoksesta. Reunoista ja viivoista johdetuista ominaisuuksista on mahdollista muodostaa objekteja ja näin mahdollistaa kuvista tunnistettavia asioita. Kuvan segmentaatio (Image segmentation) on toinen varhaisista kuvanrajaustekniikoista, jonka tavoitteena on erottaa objekteja taustasta merkitsemällä kuvapikseleitä erilaisten kynnysarvojen (Threshold) avulla. Segmentointi rajaa kuvapikselit omiin alueisiinsa, joita yhdistää jonkinasteinen yhtenäisyys, kuten kirkkaus, väri tai tekstuuri. Kynnysarvojen käyttö ja segmentointi on reunantunnistukseen verrattuna epätarkempaa, mutta silti tehokas tulkitsemaan asioita yksinkertaisemmissa kuvantunnistustehtävissä. Reunoja ja erilaisien alueita tunnistavien tekniikoiden tehokkuuteen vaikuttaa yhtä lailla kuvaolosuhteet, esimerkiksi varjot, valaistuksen vaihtelevuus ja kuvan kohina (Image noise). (Davies, 2017; Richmond, 2020; Szeliski, 2009)

2.4.3 Kuvantunnistus

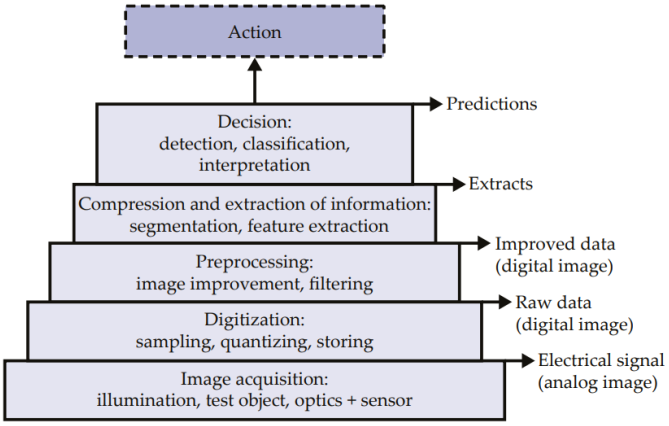
Kuvantunnistus (Image recognition) viittaa tietokonenäössä yleisiin metodeihin ja tekniikoihin, joiden tehtävä on tunnistaa ja nimetä kuvasta löytyviä asioita perustuen tiedettyihin objektien ja asioiden ominaisuuksiin. Kuvantunnistuksen katsotaan olevan tietokoneella suoritettavista visuaalisista tehtävistä kaikkein haastavin. Tarkoitus on tunnistaa esimerkiksi paikkoja, ihmisiä ja objekteja tai muita tunnistettavia elementtejä. Kuvantunnistus voidaan käsittää olevan vain osa tietokonenäön päättelyn automaatiota. Tyypillisiä kuvantunnistustehtäviä ovat kuvien luokittelu tunnistettujen ominaisuuksien perusteella, tunnistettujen ominaisuuksien paikallistaminen ja rajaaminen, hahmojen ääriviivojen rajaus (ns. maskin luominen), hahmon seuranta ja kasvojentunnistus. Hahmoilla tarkoitetaan paitsi ihmisiä, myös kaikenlaisia havaittavia objekteja ja kokonaisuuksia. Usein kuva käsittää monimutkaisia tilanteita ja maisemia, jonka vuoksi kuvantunnistusjärjestelmiä opetetaan tunnistusongelmaa vastaavilla merkityillä tietoaineistoilla (Dataseteillä). Kuvantunnistuksen opettamiseen käytetään nykyään koneoppimistekniikoita, sekä neuroverkkoja käyttävää syväoppimista. Opettamisen tuloksena algoritmit kykenevät

ennustamaan, eli tunnistamaan ja luokittelemaan kuvasta löytyvää tietoa. (Deepomatic, n.d.; Szeliski, 2009; Tryolabs, n.d.)

2.4.4 Konenäkö

Konenäöllä viitataan erikoistuneeseen järjestelmään, joka on yleensä suunniteltu teollisuuden toimialoja varten. Järjestelmät koostuvat yleensä kamerasta tai kuvan tallentimesta (kuvasensori), tietokoneesta ja kuvan tulkitsevasta ohjelmasta. Tyypillisesti tehtäviin kuuluu kohteiden ja esineiden kuvaamista ja hahmottamista. Kamerasta tai sensorista saatu kuva muutetaan digitaaliseksi ja usein käsitellään kuvanparannuskeinoin, ennen kuin kuvasta yritetään tulkita tietoa. Varsinaisessa tunnistusvaiheessa algoritmien tehtävä on erottaa kuvasta olennainen ja haluttu informaatio. Järjestelmä käyttää usein ohjattua koneoppimista, jossa sille on opetettu lukuisia mallikuvia tunnistaakseen tehtävälle tarkoituksenmukaisia ominaisuuksia ja piirteitä. Pyramidimalli (Kuva 3) havainnollistaa konenäköjärjestelmässä tapahtuvien prosessien vaiheet, jossa tapahtumat etenevät alhaalta ylöspäin. (Beyerer et al., 2015)

Kuva 3 Konenäköjärjestelmän prosessiketju (Beyerer et al., 2015)



Koneälyjärjestelmien algoritmien toimintamallit ja kaavat ovat yleensä suunniteltu kohteesta halutun informaation tunnistamiseen tai mittaamiseen. Hiljattain järjestelmien suunnittelussa on kuitenkin pyritty välttämään liiallista kaavamaisuutta ottamalla käyttöön esimerkiksi laajempia koneoppimisympäristöjä. Esineistä voidaan mitata esimerkiksi muotoa, mittasuhteita, väriä, tekstuuria tai muita pinnanmuotojen ominaisuuksia. Näiden ominaisuuksien perusteella voidaan tunnistaa esineitä ja kohteita sekä niiden materiaalia,

asentoa, ulottuvuuksia ja tarkistetaan täydellisyyttä. Viimeisenä mainitusta esimerkkinä kokoonpanovaiheet, joissa voidaan tarkistaa komponenttien määriä, mittoja ja asentoja, selvittäen puutoksia tai virheitä ennen kasausta, sen aikana ja sen jälkeen. Koodinluku kuuluu myös konenäön työtehtäviin, missä viivakoodeja tai muita esineiden pintojen merkintöjä tai kuvioita käytetään esineiden tunnistamiseen. Käytännön konenäköjärjestelmät usein yhdistelevät edellä mainittuja mittaus- ja tunnistustoimintoja tehtävissään. (Beyerer et al., 2015; Hornberg, 2017)

Konenäköä käytetään tehokkaasti havaitsemaan ja analysoimaan muutoksia teollisuuden tuotantoprosesseissa sekä mittaamaan nopeasti ja tarkasti erilaisia arvoja. Visuaalinen tarkistus on olennainen osa erilaisten tuotanto- ja kulutushyödykkeiden valmistuksessa sekä laadunvalvonnassa tekniikan ja rakentamisen sektoreilla. Konenäöllä voidaan automatisoida ja tehostaa muutoin ihmiselle rasittavia visuaalisia tai logistisia työtehtäviä, kuten valmistusvirheiden tarkistusta ja laadunvalvontaa tai materiaalinkäsittelyä ja ohjausta. Korkeat työvoimakustannukset ja tietokoneiden jatkuva kehitys ovat olleet pääsyy koneälyn yleistymisessä. (Beyerer et al., 2015; Hornberg, 2017)

2.5 Datasetit ja avoin data

Koneoppimista ja varsinkin syväoppimista varten tarvitaan paljon tietoaineistoa, jotta niillä voidaan opettaa tekoälyä tunnistamaan asioita ja tekemään tarkkoja ennusteita. Opettamisen tuloksena syntyneet tekoälymallit ovat käytännössä referenssejä, joiden avulla tekoäly ja sen algoritmit tekevät johtopäätöksiä. Opetettava tietoaineisto eli datasetti voi olla esimerkiksi tilastotietoa, mittaustuloksia, rekistereitä tai sensoridataa. Osa tekoälyn aiheista ja tehtävistä, kuten kuvantunnistus, vaativat kuitenkin monimutkaisempia ja kattavampia tietoaineistoja malleiksi, joita on jouduttu merkitsemään ja luokittelemaan manuaalisesti. Näissä malleissa tietoaineistoon on merkitty oikein ja väärin pareja, tai vastaavanlaisia esimerkkejä oikeanlaisista tai vääränlaisista tuloksista, joilla ohjataan oppimista. Tekoälymallia rakennetaan tunnistamaan haluttuja ominaisuuksia ja tavoitteena on saada erotettua oleelliset ominaisuudet. Mallien luominen syväoppimista ja neuroverkkoja varten manuaalisesti on aikaa vievää, mutta ovat ennustustuloksiltaan tarkempia verrattuna perinteisiin muutamia satoja tai tuhansia esimerkkejä sisältäviin tietomalleihin. (Kananen & Puolitaival, 2019; Koul et al., 2019; Szeliski, 2009)

Dataa syntyy joka päivä valtavia määriä, esimerkiksi kuvista, videoista, teksteistä ja erilaisista liiketoiminnan prosesseista. Datan keräys ja datasettien kehittäminen on muodostunut arvokkaaksi osa-alueeksi yrityksille, ja sen ympärille on syntynyt oma liiketoiminnan ala. Dataa ja datasettejä luodaan ja kehitetään myös avoimeen käyttöön (Open data), esimerkiksi suurien organisaatioiden, laitoksien, hallituksien ja oppilaitoksien toimesta. Tekoälyn ja tietokonenäön tutkimukseen panostetaan monelta suunnalta yhteistoiminnan kautta ja tarjolla on myös työkaluja sekä algoritmeja datasettien kehittämiseen. Avoimet datasetit sisältävät tänä päivänä esimerkiksi jo miljoonia luokiteltuja kuvia ja videoita, yhtenä merkittävimpänä datasettinä Googlen kuvatunnisteet (9 miljoonaa) ja videotunnisteet (6 miljoonaa). (Kananen & Puolitaival, 2019; Koul et al., 2019)

2.6 Google Cloud Vision API

Cloud Vision AI on Googlen kehittämä tekoälypalvelu ja koneoppimismalli, joka toimii Googlen omalla pilvipalvelualustalla. Palvelu sisältää kaksi tuotetta kuvatiedostojen tunnistamistehtäviä varten. AutoML Vision -palvelulla voi luoda omia koneoppimismalleja, eli käytännössä opettaa tekoälyä tunnistamaan haluttuja asioita omalla tietoaineistolla. Vision API -palvelu tarjoaa Googlen opettamia koneoppimismalleja käytettäväksi erilaisiin kuvantunnistustehtäviin. Päätoimintoina ovat kuvan asioiden nimeäminen ja luokittelu, objektien sekä kasvojen tunnistus sekä tekstintunnistus. Vision API toimii nimensä mukaisesti APIn eli ohjelmointirajapinnan kautta, jota voidaan käyttää yleisimmillä ohjelmointikielillä, kuten Javalla, Pythonilla tai C#. Tuettuina kuvatiedostoina ovat yleisimmät formaatit kuten JPEG, BMP, PNG ja RAW sekä myös PDF. Google suosittaa kuvatiedostojen minimikooksi 640 x 480 pikseliä, tästä poiketen tekstintunnistuksessa 1024 x 768 sekä kasvojentunnistuksessa 1600x1200 pikseliä. Kuvatiedostoja voidaan lähettää tutkittavaksi joko paikallisesta lähteestä, lataamalla ne Googlen oman Cloud Storage-tallennustilaan, tai osoittamalla URL-osoite. Lähetettävä tiedostokoko saa olla maksimissaan 20 megatavua. (Google, 2020)

Vision APIa käytetään usein yhdessä AutoML:n ja muiden tekoälypalveluiden kanssa osana automatisoituja ohjelmistoja. Google mainostaa tarjoavansa toimialansa tarkimman ennustemallin tietokonenäköön. Esimerkkinä Vision APIn käytöstä Google mainitsee kuvien arkistointi- ja organisointijärjestelmiä, tekstien jäljentämistä sekä tuotehakua. Googlen

pilvipalvelumalli tarjoaa asiakkaalle laskentatehon, tallennustilan sekä tekoälytekniikan käytettäväksi. Vision APIn kuvantunnistustoimintoja voi kokeilla ilmaiseksi 1000 hakua kuukaudessa, jonka jälkeen ylittyvät haut laskutetaan tuhannen haun erissä. Tunnistustoiminnon mukaan hinta on aluksi 1.50–3.50 dollaria / 1000 hakua / 5 miljoonaa hakua. Hinta skaalautuu halvemmaksi, kun hakumäärät ylittävät 5 miljoonaa kertaa kuukaudessa (0.60–1.50 dollaria / 1000 hakua). Yksi haku on yhden kuvatunnistustoiminnon (esim. kasvojen tunnistus) kutsun rajapinnan kautta. (Google, 2020)

2.7 Microsoft Computer Vision API

Computer Vision API on osa Microsoftin kehittämää Cognitive Services -tekoälypalvelua, joka keskittyy kuvantunnistukseen. Cognitive Services sisältää kokonaisuudessaan yli 50 erilaista tekoälypalvelua pilvipalveluna APIn eli rajapinnan kautta. Computer Vision API tarjoaa kuvantunnistustehtäviin käytettäväksi Microsoftin luomia koneoppimismalleja. Palvelun tunnistustehtäviin kuuluu esimerkiksi objektien tunnistus, kuvan luokittelu, kuvaus sisällöstä lauseenomaisesti, kasvojen tunnistus sekä tekstin tunnistus. Tutkittava kuvatiedosto voidaan lähettää paikallisena tiedostona tai antamalla URL-osoite. Palvelu tukee JPEG-, PNG-, GIF- tai BMP-formaatteja ja maksimissaan 4 megatavun tiedostokokoa. Kuvan minimi pikselikoossa tulee olla vähintään 51x51 pikseliä. (Microsoft, 2019)

Microsoft mainostaa video- ja kuvantunnistusteknologiansa olevan kehityksen kärkeä sekä tietoturvallisuuden olevan markkinoiden johtava. Microsoftin mukaan Azure-pilvipalvelualusta täyttää muihin pilvipalveluihin verrattuna eniten määräysten ja säädösten mukaisia sertifikaatteja. Computer Vision APIa voi kokeilla ilmaiseksi rajoitetuin hakumäärin, 20 hakua per minuutti ja yhteensä 5000 hakua per kuukausi. Yksi haku on yhden tietyn tunnistustoiminnon (esim. kasvojen tai tekstin tunnistus) kutsu APIn kautta. Suuremmat käyttömäärät laskutetaan tuhannen haun erissä ja hinta vaihtelee tunnistustoiminnon mukaan, aluksi ollen 0.844–1.265 euroa / tuhat hakua / miljoona hakua kuukaudessa. Hinta skaalautuu asteittain, kun hakumäärä ylittää miljoonan, kymmenen-, satojen- sekä yli sadan miljoonan. (Microsoft, 2019)

3 Kehittämistyön tavoite ja tarkoitus

Opinnäytetyön tavoitteena on selvittää miten Googlen ja Microsoftin palveluita käytetään paikallisesti kuvantunnistukseen sekä mitä tietoa niillä voidaan tunnistaa. Työssä selvitetään kahden yleisen kaupallisesti saatavilla olevan kuvantunnistuspalvelujen toiminnallisuutta sekä niiden käyttömahdollisuuksia. Googlen ja Microsoftin kuvantunnistusmetodeja testataan ja tuloksia arvioidaan sekä verrataan palveluiden kesken. Työssä ohjelmoidaan sovellus tai sovelluksia, joilla palveluita testataan ja pyritään vastaamaan tutkimuskysymyksiin sekä tutkitaan esimerkiksi, löytyykö tekoälyllä jokin luotettavasti tunnistettava tapahtuma. Toimeksiantajan päätavoitteena on selvitys kaupallisesti olevista kuvantunnistuspalveluista ja tiedon lisääminen.

Opinnäytetyö on toiminnallinen opinnäytetyö. Työn alussa tuotoksena rakentuu ohjelmia, joilla käytetään tekoälypalveluita. Toisessa osassa palveluita ja ohjelmilla saatuja tuloksia tutkitaan ja verrataan. Käyttöä varten ohjelmoidaan Python-ohjelmointikielellä sovellus, mikä käyttää palveluntarjoajan tekoälypalvelua rajapinnan kautta. Tekoälyä ja sen kuvantunnistusmenetelmiä testataan erilaisia tilanteita sisältävillä kuvilla.

4 Tekoälyn käyttö kuvantunnistuksessa

Opinnäytetyön toiminnallisessa osuudessa tekoälypalveluita käytettiin paikallisesti rajapintojen (API) kutsujen kautta Python-ohjelmakielellä luodulla ohjelmilla. Ohjelmakoodin kirjoittamiseen käytettiin opintojen aikana tutuksi tullutta Visual Studio Codea (lyhennetään VS Code), joka on ilmainen ja lukuisia ohjelmakieliä tukeva tekstieditori. Editoriin lisättiin Python-kielen tuki laajennuksena editorin oman laajennusalustan kautta.

Googlen kuvantunnistuspalveluiden käyttöä testattiin sitä varten luodulla ilmaisella kokeilutilillä (free trial). Tiliasetukset tehtiin käyttämällä Google Cloud Platform – pilvipalvelun verkkosivustoa. Tekoälypalveluiden käyttöönottoon löytyy esimerkilliset ohjeet palveluntarjoajan omilta verkkosivuilta, joiden avulla luotiin käyttötili ja ohjelmoitiin tunnistusohjelma.

Microsoftin kuvantunnistuspalveluiden käyttöä testattiin toimeksiantajan olemassa olevalla toimialueella Microsoft Azure-pilvipalvelussa, jonka käytöstä oli aiempaa kokemusta opintojen työelämäprojektin kautta. Kuten Googlen pilvipalvelussa ja sen tekoälypalveluissa, käyttöönottoon löytyy ohjeet Microsoftin omilta verkkosivuilta. Tiliasetukset tehtiin Microsoftin Azure – verkkosivustolla.

Googlen ja Microsoftin kuvantunnistuspalveluita testattiin ensin yksitellen tutkimalla rajapinnasta saatavia vastauksia. Molemmille palveluntarjoajille luotiin oma ohjelma, mihin luotiin useampaa kuvantunnistustoimintoa kutsuva kokonaisuus. Ohjelmaan luotiin myös erilaisia tietojen sekä tiedoston lukuun ja tallentamiseen liittyviä toimintoja, joiden avulla voitiin kerätä ja tallentaa tietoja palveluista. Näin saatuja tuloksia voitiin käyttää Googlen ja Microsoftin palveluiden vertailuun. Ohjelmista jätettiin pois muutamia kuvantunnistusmetodeja, jotka eivät olleet olennaisia palveluiden vertailussa, kuten värejä tunnistavat metodit.

4.1 Tunnistusohjelma

Ohjelmassa pääohjelma rakennettiin main-funktioon ja kuvantunnistustoiminnot sekä tiedon tallennustoiminto aliohjelmiin eli pienempiin funktioihin. Ohjelman käynnistyessä

pääohjelma ottaa yhteyden rajapintaan ja kutsuu aliohjelmien kuvantunnistusmetodeja, joiden avulla kerätään kuvasta halutut tiedot. Jokainen aliohjelma ohjelmoitiin poimimaan rajapinnasta työtä varten relevantteja asioita. Rajapinta lähettää vastauksen JSON-muodossa (Ohjelmakoodi 1), joka sisältää erilaisia tietoja kyseisestä kuvantunnistusoperaatiosta, esimerkiksi kuvauksien nimen (description/name), x- ja y-koordinaatit objektin rajaamiseksi (bound/bounding box) tai tekoälyn varmuuden kuvaukselle (score/confidence). Lopuksi ohjelma tallentaa aliohjelmista palautuvat tiedot taulukkomuotoon ja paikalliseen csv-tekstitiedostoon.

Ohjelmakoodi 1 Esimerkki Vision APIn JSON-vastauksesta

```
label_annotations {
  mid: "/m/02_41"
  description: "Fire"
  score: 0.8198
  topicality: 0.8198
}
```

Ohjelmalla kerättävien kuvantunnistustietojen tallentamista paikalliseksi csv-tiedostoksi varten luotiin save_csv-aliohjelma (Ohjelmakoodi 2). Tallennettava tieto saadaan pääohjelmalta funktion parametrina (df_new-muuttuja). Funktio tutkii aluksi exists()-metodilla löytyykö ennalta määritetty tiedostopolku file-muuttujasta, eli tutkii onko samanniminen tiedosto jo olemassa. Jos tiedostoa ei ole, suoritetaan lopun else-lauseen komento, mikä tallentaa pääohjelmalta saadun taulukon uuteen csv-tiedostoon kutsumalla to_csv-metodia. Jos tiedosto löytyy, funktiolla on kaksi vaihtoehtoa suorittaa tiedon tallennus csv-tiedostoon: Rivin lisäys olemassa olevaan taulukkoon, tai uuden taulukon luominen tyhjään tiedostoon. Funktio suorittaa ensin try-lauseen sisään käärityt komennot. Read_csv-metodilla avataan paikallinen tiedosto ja tallennetaan se DataFrame-taulukoksi df_from_csv-muuttujaan. Tämän jälkeen pääohjelmalta saatu taulukko ja avattu taulukko lisätään uuteen listaan frames-muuttujaan. Concat()-metodilla yhdistetään taulukoiden sarakerivit ja uusi yhdistetty taulukko tallennetaan df_edited-muuttujaan. Lopuksi taulu tallennetaan csv-tiedostoksi kutsumalla df_edited-muuttujaa to_scv-metodilla. Jos tiedosto on tyhjä, eli sinne ei ole lisätty taulukkomuotoista tietoa, except-lauseen komento tallentaa suoraan pääohjelmalta saadun taulukon csv-tiedostoksi.

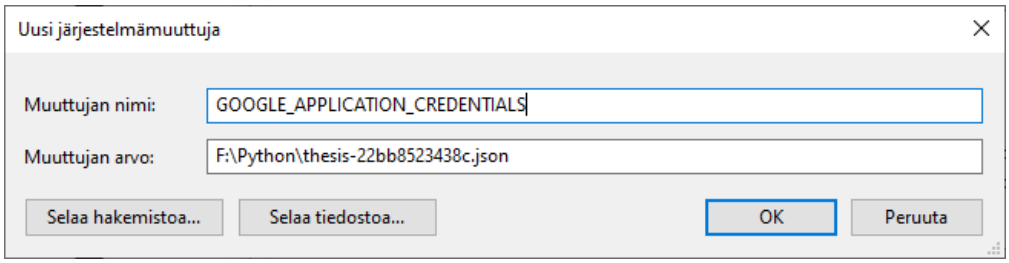
Ohjelmakoodi 2 CSV-tiedoston tallennusfunktio

```
def save_csv(df_new):
    if file.exists ():
        try:
            df_from_csv = pd.read_csv(csv_file)
            frames = [df_from_csv, df_new]
            df_edited = pd.concat(frames)
            df_edited.to_csv(csv_file, index=False)

        except pd.errors.EmptyDataError:
            df_new.to_csv(csv_file, index=False)

    else:
        print('No file, creating new')
        df_new.to_csv(csv_file, index=False)
```

4.2 Google Vision API -tunnistusohjelma

Ennen kuin Googlen palveluita pystyy käyttämään rajapintojen kautta, tulee luoda uusi projekti käyttötilille sekä ottaa laskutus käyttöön kyseiselle projektille. Tämän jälkeen on mahdollista aktivoida varsinainen tekoälypalvelu Vision API -projektin käyttöön. Googlen palveluiden käyttö rajapinnan kautta vaati todennusavaimen käyttöä, jota varten luotiin uusi todennuskäytäntö Vision APIn ja projektin välille. Luotu avain tallentuu JSON-tiedostomuotoon, joka tallennettiin ja lisättiin Windows-ympäristömuuttujiin (Kuva 4). Tämän avulla todennusavainta ei tarvitse määritellä ohjelmaan, mikä selkeytti koodia.

Kuva 4 Todennusavaimen lisäys ympäristömuuttujiin



Tämän jälkeen asennettiin Vision Client Libraries-ohjelmakirjasto Pythonia ja rajapintaa varten käyttäen komentoriviä (Komento 1). Vision Client Libraries sisältää valmiita moduuleja eli koodikirjastoja, jotta rajapintaa voidaan kutsua yksinkertaisin komennoin. Datan visualisoimista ja käsittelyä helpottamiseksi asennettiin myös Pandas-ohjelmakirjasto.

Komento 1 Ohjelmakirjastojen asennus Pythoniin (Google Vision API -ohjelma)

```
python -m pip install google-cloud-vision
Python -m pip install pandas
```

Ohjelman rakentaminen aloitettiin tuomalla ensin tarvittavia moduuleja (Ohjelmakoodi 3), jotka sisältävät eri ohjelman toimintoon vaadittavia metodeja käytettäväksi. Io-moduulin avulla voidaan hallita luettavaa ja tallennettavaa tietovirtaa, Os-moduuli tarjoaa työkaluja tiedostojen ja hakemistojen käsittelyyn. Aiemmin asennetuista ohjelmakirjastoista tuotiin vision-luokka ja pandas-moduuli. Vision sisältää Google Vision APIn sisältämät tunnistuspalvelut ja panda erilaisia taulukointiin tarvittavia työkaluja. Pathlib-moduulin avulla voidaan työskennellä helposti objektien tiedostopolkujen kanssa.

Ohjelmakoodi 3 Moduulien tuonti ohjelmaan

```
import io
import os
from google.cloud import vision_v1 as vision
import pandas as pd
import pathlib
```

Seuraavaksi luotiin globaaleja muuttujia ohjelmaan, jotta tiedostoja ja tiedostopolkuja voidaan käyttää helposti ja lyhennetyin nimin (Ohjelmakoodi 4). Tutkittavat kuvatiedostot oli tallennettu paikallisesti tietokoneen asemalle, jota varten luotiin muuttuja local_image. Tämä muuttuja määrittää kuvatiedoston kokonaisen tiedostopolun. Pelkkä tiedostonimi määriteltiin muuttujaan filename, joka tallentuu tietojen taulukointivaiheessa ensimmäisen sarakkeen tiedoksi. Ohjelma tallentaa tutkitun kuvan tiedot lopuksi csv-tiedostomuotoon, jonka nimi määriteltiin muuttujalle csv_file. Muuttuja ei sisällä tiedostopolkua, jolloin se tallentuu samaan paikkaa mistä ohjelmaa suoritetaan. Tiedostopolku tallennettiin erikseen file-muuttujaan käyttämällä pathlib-moduulin metodia pathlib.Path().

Ohjelmakoodi 4 Globaalien muuttujien määritys

```
local_image = 'F:/Python/Kuvat/Crowd/protest_3.jpg'
filename = os.path.basename(local_image)
csv_file = "google-results.csv"
file = pathlib.Path(csv_file)
```

Pääohjelmaa varten luotiin main-funktio, joka määritetään Pythonilla def main(): -lauseella (Ohjelmakoodi 5). Ensiksi ohjelma luo ilmentymän vision-luokasta client-muuttujaan sekä luo vision-luokan kuvaobjektin paikallisesta tiedostosta. Vision-luokan Ilmentymää sekä vision-luokan kuvaobjektia tarvitaan aina kun kuvantunnistusmetodia kutsutaan rajapinnan kautta. With io.open() metodi avaa ensin kuvatiedoston local_image-muuttujasta ja tallentaa sen content-muuttujaan, minkä jälkeen siitä luodaan kuvaobjekti metodilla vision.types.Image(). Seuraavaksi ohjelma luo 2-ulotteisen DataFrame-taulukon df-muuttujaan käyttämällä metodia DataFrame(). Dataframe-taulukkoon lisätään eri kuvantunnistuksen tiedot omaan sarakkeeseen lauseella df[] = []. Sarakkeen otsikko määritetään nimeämällä haluttu teksti df-muuttujan perään hakasulkeisiin. Tämän jälkeen yhtäsuuruusmerkin jälkeiseen hakasulkeisiin tulee soluun lisättävä tieto. Filename-muuttujasta saatava tiedostonimi tallennetaan ensimmäisen soluun. Seuraavien solujen tiedot haetaan kutsumalla kuvantunnistusaliohjelmia. Kuvantunnistusaliohjelmaa kutsuttaessa sille lähetetään image- ja client-muuttujat parametreina ja vastauksena palautuu tunnistusoperaatiosta saadut tiedot. Pääohjelma kutsuu viimeisenä tiedot tallentavaa save_csv metodia ja lähettää sille parametrinä kuvantunnistuksien tiedot sisältävän df-taulukon. Pääohjelman jälkeen luotiin if- ehtolauseke, joka käskee ohjelman käynnistyessä suorittamaan vain main-funktion ja sen kautta kutsutut aliohjelmat.

Ohjelmakoodi 5 Pääohjelman main-funktio

```
def main():
    client = vision.ImageAnnotatorClient()
    with io.open(local_image, 'rb') as image_file:
        content = image_file.read()
    image = vision.types.Image(content=content)
    df = pd.DataFrame()
    df['Filename'] = [filename]
    df['Labels'] = [labels(image,client)]
    df['Objects'] = [objects(image,client)]
    df['Websearch'] = [web(image,client)]
    df['Logos'] = [logos(image,client)]
    df['Violence'] = [safesearch(image,client)]
    df['Faces'] = [faces(image,client)]
    df['Landmarks'] = [landmarks(image,client)]
    df['Texts'] = [texts(image,client)]
    save_csv(df)

if __name__ == "__main__":
    main()
```

4.2.1 Label detection

Label detection -metodi antaa vastauksena kymmenen tekoälyn varmimmin tunnistamaa kuvausta kuvan sisällöstä. Kuvaus sisältää mid-arvon (kuvauksen yksilötunniste tietokannassa), kuvauksen nimen, tekoälyn varmuuden kuvauksesta desimaaliarvona sekä topicality-arvon desimaalina. Topicality-arvon eroa varmuuden arvoon ei Google ole selventänyt ja niiden arvot olivat testaushetkellä samat. Kuvaustentunnistus-aliohjelmaksi luotiin labels-funktio, joka kutsuu ilmentymän label_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 6). Seuraavaksi erotetaan kuvauksien nimet response-muuttujasta käyttäen for-silmukkaa ja tallennetaan labels-muuttujaan listaksi. Lopuksi lista lähetetään pääohjelmalle return-lauseella.

Ohjelmakoodi 6 Kuvaustunnisteita tunnistava labels-funktio

```
def labels(image, client):
    response = client.label_detection(image=image)
    labels = [label.description for label in response.label_annotations]
    return labels
```

4.2.2 Object localization

Object localization -metodi antaa vastauksena kuvasta löydetyt objektit sekä niiden sijainnit. Vastaus sisältää objektin nimen, mid-arvon, tekoälyn varmuuden objektista sekä koordinaatit objektin rajaukselle. Objektientunnistus-aliohjelmaksi luotiin objects-funktio, joka kutsuu ilmentymän object_localization-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 7). Havaittujen objektien nimet tallennetaan listaksi objects-muuttujaan erottamalla ne response-muttujasta for-silmukalla. Lopuksi lista lähetetään return-lauseella takaisin pääohjelmalle.

Ohjelmakoodi 7 Objekteja tunnistava objects-funktio

```
def objects(image, client):
    response = client.object_localization(image=image)
    objects = [object.name for object in response.localized_object_annotations]
    return objects
```

4.2.3 Text detection

Text detection -metodi antaa vastauksena tunnistettavat tekstit, niistä jokaisen kirjaimen eriteltynä, sekä tunnistaa niiden kielen sekä sijainnin (rajauksen x- ja y-koordinaatit). Tekstintunnistus-aliohjelmaksi luotiin texts-funktio, joka kutsuu ilmentymän text_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 8). Tekstit erotetaan response-muuttujasta käyttäen for-silmukkaa ja tallennetaan texts-muuttujaan listaksi. Lopuksi lista lähetetään pääohjelmalle return-lauseella.

Ohjelmakoodi 8 Tekstiä tunnistava texts-funktio

```
def texts(image, client):
    response = client.text_detection(image=image)
    texts = [text.description for text in response.text_annotations]
    return texts
```

4.2.4 Landmark detection

Landmark detection -metodi antaa vastauksena kuvauksen tunnistetuista maamerkeistä. Kuvaus sisältää mid-arvon, kuvauksen nimen, varmuuden kuvauksesta sekä sijainnin x- ja y-arvot rajaukselle. Maamerkin tunnistus-aliohjelmaksi luotiin landmarks-funktio, joka kutsuu ilmentymän landmark_detection-metodia ja tallentaa vastauksen response muuttujaan (Ohjelmakoodi 9). Kuvausten nimet erotetaan response-muuttujasta for-silmukalla ja tallennetaan listaksi landmarks-muuttujaan. Lopuksi lista lähetetään pääohjelmalle return-lauseella.

Ohjelmakoodi 9 Maamerkkejä tunnistava landmarks-funktio

```
def landmarks(image, client):
    response = client.landmark_detection(image=image)
    landmarks = [landmark.description for landmark in response.landmark_annotations]
    return landmarks
```

4.2.5 Logo detection

Logo detection -metodi antaa vastauksena tunnistetut logot ja niiden kuvauksen. Kuvaus sisältää mid-arvon, kuvauksen nimen, varmuuden kuvauksesta, ja objektin sijainnin x- ja y-arvot rajaukselle. Logon tunnistus-aliohjelmaksi luotiin logos-funktio, joka kutsuu ilmentymän logo_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 10). Kuvausten nimet erotetaan response-muuttujasta listaksi logos-muuttujaan käyttämällä for-silmukkaa. Lopuksi lähetetään lista pääohjelmalle return-lauseella

Ohjelmakoodi 10 Logoja tunnistava logos-funktio

```
def logos(image, client):
    response = client.logo_detection(image=image)
    logos = [logo.description for logo in response.logo_annotations]
    return logos
```

4.2.6 SafeSearch detection

SafeSearch detection -metodi antaa vastauksena viiden eri kategorian todennäköisyyden epäsiveelliselle sisällölle avain-arvo-pareina eli sanakirjana (Ohjelmakoodi 11). Avaimia kategorioille ovat adult, spoof, medical, violence ja racy. Arvot ovat Googlen etukäteen määrittelemät todennäköisyysnimet ”UNKNOWN”, ”VERY UNLIKELY”, ”UNLIKELY”, ”POSSIBLE”, ”LIKELY ”ja ”VERY LIKELY”. SafeSearch-aliohjelmaksi luotiin safesearch-funktio, joka kutsuu ilmentymän safe_search_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 12). Response-muuttujasta erotetaan väkivaltaan (violence) liittyvä todennäköisyys for-silmukalla ja tallennetaan filter_violence-muuttujaan. Todennäköisyysnimen arvo tallennetaan violence_value-muuttujaan poimimalla violence-

avaimen arvo filter_violence-muuttujasta. Lopuksi arvo lähetetään pääohjelmalle return-lauseella.

Ohjelmakoodi 11 Avain-arvopareista muodostuva sanakirja

```
adult: UNLIKELY
spoof: VERY_UNLIKELY
medical: VERY_UNLIKELY
violence: UNLIKELY
racy: POSSIBLE
```

Ohjelmakoodi 12 Epäsiveellistä sisältöä tunnistava safesearch-funktio

```
def safesearch(image, client):
    likelihood_name = ('UNKNOWN', 'VERY_UNLIKELY', 'UNLIKELY', 'POSSIBLE',
                       'LIKELY', 'VERY_LIKELY')
    response = client.safe_search_detection(image=image)
    filter_violence = dict(violence='{}'.format(likelihood_name[response.safe_search_annotation.violence]))
    violence_value = filter_violence['violence']
    return violence_value
```

4.2.7 Face detection

Face detection -metodi antaa vastauksena tunnistetuille kasvoille lukuisia ominaisuuksia. Kasvon sijainnista saadaan x- ja y-arvot rajaukselle. Metodi tunnistaa kasvonpiirteinä (landmarks) silmät, kulmakarvat, nenän, huulet, suun, korvat ja leuan, sekä antaa niiden kolmiulotteiset x,y,z-koordinaatit. Pään asennosta (angle) saadaan akselien (roll, pan, tilt) asteluvut desimaaliarvoina. Kasvojen- sekä kasvonpiirteiden tunnistusvarmuudelle saadaan desimaaliarvot. Metodi antaa vielä todennäköisyysarvot (likelihood-arvot kuten safesearch-metodissa) kasvojen tunteille sekä sumennetuille tai päähineitä sisältäville kasvoille. Tuloksia ja niiden vertailua varten tiedoista tallennettiin vain tunnistettavien kasvojen lukumäärä. Kasvojentunnistus-aliohjelmaksi luotiin faces-funktio, joka kutsuu ilmentymän face_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 13). Tunnistettujen kasvojen lukumäärä lasketaan len()-metodilla response-muuttujasta ja tallennetaan face_count-muuttujaan. Lopuksi lukumäärä lähetetään pääohjelmalle return-lauseella.

Ohjelmakoodi 13 Kasvoja tunnistava faces-funktio

```
def faces(image, client):
    response = client.face_detection(image=image)
    face_count = len(response.face_annotations)
    return face_count
```

4.2.8 Web detection

Web detection -metodi antaa vastauksena erilaisia verkkohaulla löytyviä tuloksia kuvasta löydettyihin tietoihin. Metodi antaa maksimissaan kymmenen tulosta jokaisesta seuraavasta asiasta: Tekoälyn mielestä osuvimmat kuvaukset (web_entities), url-osoite saman resoluutioiseen kuvaan (full_matching_images), url osoite eri resoluution kuvaan (partial_matching_images), url-osoite sivuun mistä kuva löytyy ja sivun otsikkotieto (pages_with_matching_images), url-osoite vastaavanlaiseen kuvaan (visually_similar images). Viimeinen metodin antama tieto ja ainoa mikä valittiin käytettäväksi, on web-haun osuvin arvaus kuvan sisällöstä (best_guess_labels). Verkkohaku-aliohjelmaksi luotiin web-funktio, joka kutsuu ilmentymän web_detection-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 14). Verkkohaun osuvin arvaus kuvauksesta erotetaan response-muuttujasta käyttämällä for-silmukkaa ja tallennetaan best_guess-muuttujaan. Lopuksi kuvaus lähetetään lista pääohjelmalle return-lauseella.

Ohjelmakoodi 14 Verkkohakua suorittava web-funktio

```
def web(image, client):
    response = client.web_detection(image=image)
    best_guess = [label.label for label in response.web_detection.best_guess_labels]
    return best_guess
```

4.3 Microsoft Computer Vision API -tunnistusohjelma

Microsoftin kuvantunnistuspalveluiden käyttö rajapinnan kautta vaatii todennusavaimen ja endpoint-tunnisteen. Näitä varten luotiin uusi Computer Vision -resurssi olemassa olevaan toimialueeseen ja resurssiryhmään. Resurssin luomisen jälkeen avain ja tunniste löytyy

kopioitavaksi resurssin hallintapaneelista. Tämän jälkeen asennettiin Pythoniin Cognitive Services Computer Vision- sekä Pandas-ohjelmakirjastot käyttäen komentoriviä (Komento 2).

Komento 2 Ohjelmakirjastojen asennus Pythoniin (Microsoft Vision API -ohjelma)

```
python -m -pip install azure-cognitiveservices-vision-computervision
Python -m pip install pandas
```

Ohjelma rakennettiin samantyylisesti kuin Googlen tunnistusohjelma. Pääohjelmaksi rakennettiin main-funktio, joka ohjelman käynnistyessä ottaa yhteyden rajapintaan ja kerää kuvantunnistustietoja kutsumalla aliohjelmien kuvantunnistusmetodeja. Lopuksi ohjelma tallentaa tiedot taulukkomuotoon ja paikalliseen csv-tekstitiedostoon. Ohjelman rakentaminen aloitettiin tuomalla ensin ohjelman toimintoon vaadittavia moduuleja (Ohjelmakoodi 15). Aiemmin asennetuista ohjelmakirjastoista tuotiin ComputerVisionClient- ja CognitiveServicesCredentials-luokka sekä pandas-moduuli. Os-, pandas ja Pathlib-moduulit tuotiin työkaluiksi tiedostojen ja hakemistojen käsittelyyn sekä datan taulukointiin.

Ohjelmakoodi 15 Moduulien tuonti ohjelmaan

```
import os
from azure.cognitiveservices.vision.computervision import ComputerVisionClient as vision
from msrest.authentication import CognitiveServicesCredentials as credentials
import pandas as pd
import pathlib
```

Seuraavaksi ohjelmaan lisättiin globaaleja muuttujia (Ohjelmakoodi 16). Rajapinnan kutsuminen vaatii todentamista, jota varten todennusavain ja endpoint-tunniste tallennetiin omiin muuttujiin subscrition_key ja endpoint. Tutkittavan paikallisen kuvatiedoston polku määritettiin muuttujaksi local_image. Pelkkä tiedostonimi eroteltiin filename-muuttujaan, joka tallentuu taulukointivaiheessa ensimmäisenä tietona. Ohjelma tallentaa lopuksi kuvatiedot csv-tiedostoon, jonka nimeä varten luotiin csv_file-muuttuja. Lopuksi csv:n tiedostopolkua varten luotiin oma file-muuttuja.

Ohjelmakoodi 16 Globaalien muuttujien lisääminen

```
subscription_key = "419845bd62504a[redacted]"
endpoint = "https://thesis-cognitive.cognitiveservices.azure.com/"

local_image = 'F:/Python/Kuvat/Crowd/capitol_1.jpg'
filename = os.path.basename(local_image)
csv_file = "microsoft-results.csv"
file = pathlib.Path(csv_file)
```

Pääohjelmaa varten luotiin main-funktio, joka suoritetaan aina ohjelman käynnistyessä (Ohjelmakoodi 17). Main-funktio luo ensiksi ilmentymän vision-luokasta, jonka parametreiksi annetaan todennukseen tarvittava avain sekä endpoint-tunniste. Ilmentymää tarvitaan aina, kun kuvantunnistusmetodia kutsutaan rajapinnan kautta aliohjelmassa. Seuraavaksi ohjelma luo DataFrame-taulukon df-muuttujaan, johon kerätään eri kuvantunnistusmetodien tulokset. Taulukon ensimmäisen sarakkeen soluun tallennetaan tiedostonimi muuttujasta filename. Ohjelma lisää sarake kerrallaan tietoa taulukkoon lauseella df[] = []. Ensimmäinen hakasulje määrittää sarakkeen nimen ja jälkimmäinen lisää soluun kuvantunnistusmetodin tiedon kutsumalla aliohjelmaa. Aliohjelmaa kutsuttaessa sille annetaan vision-luokan ilmentymä parametrina (client-muuttuja). Aliohjelma kutsuu rajapintaa käyttäen ilmentymän kuvantunnistusmetodia ja lähettää antaa parametrina paikallisen kuvatiedoston muuttujana). Viimeiseksi pääohjelma tallentaa tiedot kutsumalla save_csv-funktiota, antaen parametriksi kuvantunnistuksien tiedot sisältävän df-muuttujan.

Ohjelmakoodi 17 Pääohjelma ja main-funktio

```
def main():
    client = vision(endpoint, credentials(subscription_key))
    df = pd.DataFrame()
    df['Filename'] = [filename]
    df['Tags'] = [tags(client)]
    df['Objects'] = [objects(client)]
    df['Describe'] = [describe(client)]
    df['Category'] = [category(client)]
    df['Logos'] = [logos(client)]
    df['Faces'] = [faces(client)]
    df['Landmarks'] = [domain(client)]
    df['Texts'] = [texts(client)]
    save_csv(df)

if __name__ == "__main__":
    main()
```

4.3.1 Content tags

Content tags -metodi antaa vastauksena tunnisteita kuvasta löytyneistä asioista ja objekteista. Tunniste sisältää kuvauksen nimen ja desimaaliarvon tekoälyn varmuudelle (confidence). Kuvatunniste-aliohjelmaksi luotiin tags-funktio, joka kutsuu ilmentymän tag_image_in_stream-metodia (Ohjelmakoodi 18). Vastaus tallennetaan response-muuttujaan, mistä erotetaan for-silmukan avulla pelkät kuvausten nimet ja tallennetaan edelleen tags-muuttujaan. Lopuksi lista lähetetään takaisin pääohjelmalle return-lauseella.

Ohjelmakoodi 18 Kuvatunnisteita tunnistava tags-funktio

```
def tags(client):
    image = open(local_image, "rb")
    response = client.tag_image_in_stream(image)
    tags = [tag.name for tag in response.tags]
    return tags
```

4.3.2 Detect objects

Detect objects -metodi tunnistaa objekteja sekä niiden sijainnin kuvasta. Metodin vastaus sisältää objektin nimen, tekoälyn varmuuden desimaaliarvona sekä koordinaatit sijainnin rajaukselle. Objektintunnistus-aliohjelmaksi luotiin objects-funktio, joka kutsuu ilmentymän detect_objects_in_stream-metodia (Ohjelmakoodi 19).

Ohjelmakoodi 19 Objekteja tunnistava objects-funktio

```
def objects(client):
    image = open(local_image, "rb")
    response = client.detect_objects_in_stream(image)
    objects = [obj.object_property for obj in response.objects]
    return objects
```

4.3.3 Image Descriptions

Image descriptions -metodi antaa vastauksena lauseenomaisia kuvauksia kuvan sisällöstä, tekoälyn varmuuden kuvailuista desimaaliarvona sekä tunnisteiden nimiä. Content tags -

metodin tunnisteiden erona Microsoft selittää niiden olevan monipuolisempia Image descriptions -metodissa, kun taas Content tags -metodi on optimoitu tarkkuuteen. Kuvaus-aliohjelmaksi luotiin describe-funktion, joka kutsuu ilmentymän describe_image_in_stream-metodia (Ohjelmakoodi 20). Metodin vastaus tallennetaan response-muuttujaan, jonka jälkeen kuvauslauseet erotellaan for-silmukan avulla listaksi ja tallennetaan text-muuttujaan. Lopuksi lähetetään lista pääohjelmalle return-lauseella.

Ohjelmakoodi 20 Kuvauksia tunnistava describe-funktio

```
def describe(client):
    image = open(local_image, "rb")
    response = client.describe_image_in_stream(image)
    text = [caption.text for caption in response.captions]
    return text
```

4.3.4 Image Categorization

Image categorization -metodi antaa vastauksena taksonomiapohjaisen luokittelun kuvasta löydetyille asioille. Luokittelu eroaa tunnisteista sen hierarkkisesti ryhmitetyn järjestelyn mukaan. Luokittelussa on vain 86-kategoriaa, verrattuna tunnisteiden tuhansiin vaihtoehtoihin. Luokittelu-aliohjelmaksi luotiin category-funktio, joka kutsuu ilmentymän analyze_image_in_stream-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 21). Metodin parametriksi annetaan kuvaobjektin lisäksi features-muuttujaan määritelty ”categories”. Luokittelujen nimet tallennetaan listaksi categories-muuttujaan käyttämällä for-silmukkaa. Lopuksi lista lähetetään return-funktiolla takaisin pääohjelmalle.

Ohjelmakoodi 21 Kuvaa luokitteleva category-funktio

```
def category(client):
    image = open(local_image, "rb")
    features = ["categories"]
    response = client.analyze_image_in_stream(image, features)
    categories = [category.name for category in response.categories]
    return categories
```

4.3.5 Brand detection

Brand detection -metodi havaitsee tunnettuja logoja (tuotemerkkejä, myös teksimuotoisia) kuvan sisällöstä. Metodin vastaus sisältää havaitun tuotemerkin nimen, tekoälyn varmuuden desimaaliarvona sekä sijainnin koordinaatit rajaukselle. Tuotemerkki-aliohjelmaksi luotiin logos-funktio, joka kutsuu ilmentymän analyze_image_in_sream-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 22). Metodin parametriksi annetaan kuvaobjektin lisäksi features-muuttujaan määritelty ”brands”. Logot erotetaan response-muuttujasta for-silmukan avulla ja tallennetaan listaksi logos-muuttujaan. Lopuksi lista lähetetään return-lauseella pääohjelmalle.

Ohjelmakoodi 22 Logoja ja tuotemerkkejä tunnistava brands-funktio

```
def logos(client):
    image = open(local_image, "rb")
    features = ["brands"]
    response = client.analyze_image_in_stream(image, features)
    logos = [brand.name for brand in response.brands]
    return logos
```

4.3.6 Adult content detection

Adult content detection -metodi tunnistaa aikuisille suunnattua materiaalia kuvista. Metodi luokittelee kuvan kolmeen kategoriaan: Adult, Racy sekä Gory. Metodi antaa vastauksena kategorioiden tosi/epätosi-arvot sekä tekoälyn varmuuden desimaaliarvona. Adult-kategoria tunnistaa selvästi erottuvan seksuaalisen, esimerkiksi alastomuuden, sisällön. Racy-kategoria tunnistaa luonteeltaan seksuaalisia asioita, enemmin siihen viittaavia asioita kuin esimerkiksi suoranaista alastomuutta tunnistava Adult. Gory-kategoria tunnistaa veristä sisältöä. Metodia ja sen kuvantunnistusta ei sisällytetty ohjelmaan, koska sen tulosten testaaminen ja vertaileminen ei ollut viitekehykseen olennaista.

4.3.7 Face detection

Face detection -metodi tunnistaa kasvoja kuvasta ja antaa vastauksena iän, sukupuolen sekä rajauksen koordinaatit. Kasvojentunnistus-aliohjelmaksi luotiin faces-funktio, joka kutsuu

ilmentymän analyze_image_in_stream-metodia ja tallentaa vastauksen response-muuttujaan (Ohjelmakoodi 23). Metodin parametriksi annetaan kuvaobjektin lisäksi features-muuttujaan määritelty ”faces”. Kasvojen lukumäärä lasketaan len()-funktiolla antamalla sen parametriksi vastauksena saadut kasvo-objektit (response.faces). Lopuksi lukumäärä lähetetään takaisin pääohjelmaan return-lauseella.

Ohjelmakoodi 23 Kasvoja tunnistava faces-funktio

```
def faces(client):
    image = open(local_image, "rb")
    features = ["faces"]
    response = client.analyze_image_in_stream(image, features)
    face_count = len(response.faces)
    return face_count
```

4.3.8 Domain-specific content

Domain-specific content -metodi tunnistaa kuvasta julkisuuden henkilöitä ja maamerkkejä. Vastauksena henkilöstä saadaan nimi, tekoälyn varmuuden desimaaliarvona sekä rajauksen koordinaatit. Maamerkistä saadaan vastauksena nimi sekä tekoälyn varmuus desimaaliarvona. Maamerkintunnistus-funktioksi luotiin domain-funktio, joka kutsuu ilmentymän analyze_image_by_domain_in_stream-metodia (Ohjelmakoodi 24). Metodille annetaan ensimmäiseksi parametriksi ”landmarks”, jolla saatu vastaus tallennetaan response-muuttujaan. Maamerkkien nimet erotetaan for-silmukalla response-muuttujasta ja tallennetaan omaan landmarks-muuttujaan. Lopuksi nimet lähetetään return-lauseella takaisin pääohjelmalle.

Ohjelmakoodi 24 Maamerkkejä tunnistava domain-funktio

```
def domain(client):
    image = open(local_image, "rb")
    response = client.analyze_image_by_domain_in_stream("landmarks", image)
    landmarks = [landmark["name"] for landmark in response.result["landmarks"]]
    return landmarks
```

4.3.9 Optical Character Recognition (OCR)

Optical Character Recognition -metodi tunnistaa tekstiä kuvasta ja erottelee ne vastauksessa hierarkisesti alue/rivi/sana -tyyliin. Metodi antaa vastauksena ensin tunnistetun kielen (27 tuettua kieltä), tekstin kulman radiaaneina sekä tekstin suuntauksen (up, down, left tai right). Tämän jälkeen metodi antaa tekstinä löydetyn sanan sekä koordinaatit tekstin alueesta (regions), tekstirivistä (lines) sekä itse sanasta (word). Tekstintunnistus-aliohjelmaksi luotiin texts-funktio, joka kutsuu ilmentymän recognize_printed_text_in_stream-metodia (Ohjelmakoodi 25). Metodin vastaus tallennetaan response-muuttujaan, josta erotetaan tunnistetut sanalistaksi texts-muuttujaan käyttämällä for-silmukkaa. Lopuksi sanalista lähetään takaisin pääohjelmalle return-lauseella.

Ohjelmakoodi 25 Tekstiä tunnistava texts-funktio

```
def texts(client):
    image = open(local_image, "rb")
    response = client.recognize_printed_text_in_stream(image)
    texts = [word.text for region in response.regions for line in region.lines for word in line.words]
    return texts
```

5 Tuloksien ja palveluiden vertailu

Kuvantunnistusmetodeja testattiin kaikkiaan noin 50:llä eri kuvatiedostolla. Suurin osa kuvista käsitti erilaisia ihmisiä sisältäviä tilannekuvia. Tilanteisiin sisältyi esimerkiksi urheilutapahtumia, julkisia paikkoja sekä tilaisuuksia. Osa kuvantunnistusmetodeista vaati testattavaksi myös selkeämpiä ja yksinkertaisempia objekteja sisältäviä kuvia, jotta niitä pystyttiin vertailemaan. Tästä esimerkkinä logojen ja maamerkkien tunnistus, joiden testaus vaati tarkemmin määriteltyjä kuva-asetelmia. Googlen ja Microsoftin tekoälytekniikkaa ja algoritmeja ei ole avattu ja selitetty julkisesti ja niiden keskinäinen paremmuusvertailu ei välttämättä kerro itse tekoälyn tehokkuudesta, vaan palveluiden toiminnallisuuksien erilaisesta suunnittelusta. Tunnistusmetodi on voitu suunnitella eri painoarvoja käyttäen, kuten tarkkuutta tai monipuolisuutta painottaen, tai tunnistamaan erilaisia lisäominaisuuksia pelkän havaitsemisen lisäksi. Kuvantunnistusmetodeja testattaessa huomattiin kuitenkin käytännön eroja kyvyssä tunnistaa ja tulkita asioita Googlen ja Microsoftin tekoälyjen kesken.

Kuvan sisältöä kuvailevassa metodissa (Labels/Tags) Google tunnisti aina maksimimäärän eli 10 kuvausta/tunnistetta. Kuvaukset olivat hyvin tarkkoja sekä monipuolisia ja sisälsi myös tilannetta tai ympäristöä kuvailevia sanoja. Google pystyi myös erottamaan ihmisen lähikuvasta kasvonpiirteitä ja ilmeitä. Microsoft tunnisti vaihtelevasti kolmesta jopa 34:ään erilaista kuvausta, keskimäärin kuitenkin vähemmän kuin Google. Kuvaukset olivat Googleen verrattuna usein yksinkertaisempia ja sisälsi vähemmän objekteja ja tilannetta kuvaavia tunnisteita.

Sisällön kuvailemiseen oli myös muita metodeja, jotka kuvaavat eri tavalla tai tekniikalla löydettyä sisältöä. Googlen verkkohaun tekevä Websearch-metodi antoi yksinkertaisen kuvauksen koko maisemasta tai tilanteesta. Verkkohaku todennäköisesti käytti kuvaa Googlen kuvahakuun, koska samanlaisia tuloksia sai kuvahakuun ladatessa kyseisen kuvan. Verkkohaku osasi tunnistaa hyvin julkisia paikkoja, rakennuksia ja monumentteja. Googlen väkivaltaa tunnistava SafeSearch-metodi ei tunnistanut kamppailulajeista mitään epäilyttävää, mutta itkevän naisen kuvasta antoi tuloksena ”Possible”. Microsoftin Categorization-metodi luokitteli kuvasta löydettyjä asioita kategorioihin. Käytännössä kategorisoitu tunniste sisälsi parent- ja child-tunnisteen, esimerkiksi object_sculpture tai

people_group. Kategoriatunnisteet eivät antaneet tuloksiin sinällään mitään lisäarvoa, kategorioiden vähyys (86 erilaista) ei pysty luomaan tarpeeksi monipuolisia luokituksia, kuin metodilta olettaisi saavan.

Microsoftin tekoäly pystyi antamaan myös lauseenomaisia kuvauksia (kuvatekstejä) kuvasisällöstä, mitä Googlen tekoäly ei tarjoa. Image Description -metodi antoi lähes kaikille testatuille kuville osuvan kuvatekstin kuvassa tapahtuvasta tilanteesta. Ihmisten aktiviteeteistä sai tarkkoja kuvauksia, kuten ” Firefighters putting out a fire”, ” A group of people holding signs in front of a white building” ja ” 'A person loading boxes into a truck”. Mielenkiintoisena tuloksena metodi antoi kuvatekstin Vapaudenpatsas-kuvasta ”a statue of a person holding a torch”, mitä puolestaan Microsoftin maamerkkien tunnistusmetodi ei tunnistanut Vapaudenpatsaaksi.

Objekteja tunnistavassa metodissa (Object detection/localization) molemmat palvelut tunnistivat suurimmaksi osaksi henkilöitä ja vaatteenosia tai päähineitä. Googlen tekoäly tunnisti selkeästi enemmän henkilöitä, vaihdellen yhdestä neljään enemmän kuin Microsoftin vastaava metodi, sekä pystyi hahmottamaan enemmän satunnaisia esineitä kuten käyttötavaroita. Googlen tekoälyn kuvasta tunnistamat henkilöt vastasivat lukumäärältään likimain niitä henkilöitä, joiden kasvonpiirteet olivat tunnistettavissa. Vasta hiljattain Google muutti käytäntöään henkilöntunnistuksessa, ja siirtyi sukupuolen tunnistamisesta pelkkään ”henkilö”-tunnisteeseen.

Kasvojentunnistus oli Googlen tekoälyllä huomattavasti tarkempaa, niin havainnoinnissa että tunnistuksen tuloksen monimuotoisuudessa. Googlen metodi osasi tunnistaa kymmenittäin eri kasvonpiirteiden osa-alueita, sekä kasvonpiirteiden perusteella pään asennon. Microsoftin metodi antoi tuloksen lisätietoina sukupuolen ja iän. Microsoft tunnisti vain hieman yli viidesosan Googlen tunnistamista kasvoista testikuvissa ja Google saattoi tunnistaa kuvasta jopa 7 kasvoa enemmän.

Maamerkkien tunnistuksessa Googlen palvelu pystyi tunnistamaan paremmin maamerkkejä ja tunnettuja rakennuksia. Microsoft mainitsee palvelunsa tunnistavan yli 9000 maamerkkiä ympäri maailmaa, mutta ei esimerkiksi tunnistanut New Yorkin Vapaudenpatsasta. Googlen kuvahaulla on oletettavasti suuri merkitys Googlen tunnistusmetodin tehokkuudessa.

Tekstintunnistus eroaa palveluiden kesken huomattavasti. Microsoftin tunnistusmetodi on suunniteltu käsin kirjoitettuun tai printattuun tekstiin ja yleisesti ottaen vain tekstiä sisältäviin kuviin. Microsoftin tekoäly pystyi tunnistamaan kuvamaisemasta vain suhteellisen isoja tekstejä, kuten kylttejä tai mainostauluja. Googlen tekoäly pystyi tunnistamaan monimutkaisestakin kuvamaisemasta lähes kaikki silmin erotettavat tekstit, yksittäisen kirjaimen tarkkuudella. Tarkkuus kuitenkin heikentyi huomattavasti, jos teksti oli käsin kirjoitettua tai todella pientä ja pikselöitynyttä. Tämä vaikutti tuloksiin joiltain osin niin, että metodin tunnistamat sanat sisälsivät paljon kirjoitusvirheitä, lähes pisteeseen missä sanaa ei voinut enää edes päätellä.

Logoja ja tuotemerkkejä tunnistava metodi (Logos/Brands) tunnisti Googlen tekoälyllä monipuolisemmin, mutta sisälsivät virheitäkin. Google tunnisti myös muutamia logoja, kuten koulujen ja yliopistojen logoja, mitä kuvatiedosto ei sisältänyt. Thaimaalaisesta mellakkakuvasta Google antoi tuloksena Hammerskins skinhead -järjestön logon, joka herättää epäilyksen, että tunnistus perustui ennemmin tilannekuvan kontekstiin tai verkkohakuun, kuin konkreettisesti havaittuun asiaan. Microsoftin tekoäly ei tunnistanut yhtäkään väärää logoa, joten sen tunnistustehokkuutta voidaan sen vähäisiin tuloksiin nähden kuitenkin hyvänä. Microsoft kertoo tietokantansa sisältävän ”tuhansia” kaupallisia logoja, mutta jätti tunnistamatta varsin tunnettuja Yhdysvaltalaisia logoja, kuten Ford tai Delta Air Lines. Vähemmän yllättävästi molemmat tunnistivat trendaavia logoja, kuten Google, Ebay ja McDonald´s. Logojen ja tuotemerkkien tunnistustesteistä ei selvinnyt Microsoftin osalta oliko sen algoritmi painotettu tehokkuuteen vai oliko tietokanta vain liian pieni.

6 Johtopäätökset ja pohdinta

Kuvantunnistuksen testausta ja vertailua varten ohjelmoitiin Python-ohjelmointikielellä sovellukset, joilla kutsuttiin kuvantunnistuspalveluita paikallisesti. Tekoälypalveluiden käyttöönotto onnistui vaivattomasti, vaikkakin Microsoftin Azure-toimialueen kanssa oli aluksi käyttöoikeusongelmia. Kuvantunnistustoimintojen käyttöön löytyi esimerkkejä sekä Googlen että Microsoftin verkkosivuilta, joiden avulla luotu Python-sovellus rakentui pala kerrallaan. Kehitystyönä luotiin toimivat sovellukset tekoälyjen rajapintakutsuja varten. Sovelluksia avulla saatiin testattua kuvantunnistusmetodeja ja niiden toiminnallisuutta. Testauksen tuloksista saatiin tietoa kuvantunnistuksen mahdollisuuksista ja palveluita pystyttiin vertailemaan.

Työn tuloksena saatiin tietoa kuvantunnistustoimintojen käytöstä ja ominaisuuksista, sekä konkreettisia tuloksia, joita arvioimalla palveluista huomattiin keskinäisiä eroja. Tuloksia verratessa huomattiin, että Googlen tekoäly omaa selkeästi tarkemman objektien, kasvojen sekä tekstin tunnistuksen. Googlen kasvojentunnistus on kehitetty erittäin monipuoliseksi, joka osaa tunnistaa kasvojenpiirteiden sijainnin ja asennon lisäksi myös pään asentoa. Tämä mahdollistaa käytännössä todella tarkan kasvojen tunnistuksen sekä seurannan.

Microsoftin tekoäly sisälsi tunnistustoiminnon, joka pystyy luomaan lauseenomaisia kuvauksia kuvasisällöstä. Nämä kuvatekstien tyyliset kuvaukset osoittautuivat kertomaan kuvasta käytännössä enemmän, kuin mitä olisi voinut päätellä kaikista muista löydetyistä yksittäisistä tiedoista. Vastaavaa toimintoa ei Googlen palvelussa ole tarjolla.

Testattujen tekoälypalvelun käyttöönotto ja yhdistäminen sovelluksen toiminnallisuuden osaksi on erittäin helppoa ja kynnys palveluiden käyttöönottoon on tehty erittäin matalaksi. Palvelut toimivat pilvipalvelualustalla todella pienellä viiveellä, eikä vaadi käyttäjän puolesta minkäänlaisia tehoresursseja. Googlen ja Microsoftin tekoälyjä on myös mahdollista opettaa kustomoiduilla tunnistusmalleilla ja yhdistää valmiisiin tunnistuspalveluihin, mikä mahdollistaa palveluiden muokkaamisen yrityksille sopivaan käyttöön.

7 Yhteenveto

Opinnäytetyön toiminnallisessa osassa onnistuin luomaan toimivat sovellukset, joilla tekoälypalveluita pystytään käyttämään ja kuvantunnistustoimintojen tuloksia saadaan kerättyä ja tallennettua. Tutkimuskysymyksiin sain vastaukset sovelluksien avulla ja tuloksena lopulta höydyllistä tietoa Googlen ja Microsoftin kuvantunnistustoiminnoista. Videotiedostojen tunnistusta en ehtinyt testaamaan ja näin ollen yksi osa tutkimuskysymyksestä jäi vastaamatta. Kuvantunnistuksessa riitti kuitenkin hyvin laajasti käsiteltävää ja sain kattavasti tietoa niiden toiminnoista ja mahdollisuuksista.

Python-ohjelmointikielestä minulla ei ollut ennestään kovin hyvää osaamista, minkä vuoksi valitsin sen käytettäväksi sovelluksissa. Opinnäytetyön aikana opin paljon uutta Pythonin käytöstä ja koin sen hyvänä valintana, sillä se on hyvin yleisesti käytössä oleva ohjelmointikieli. Datan käsittely oli Pythonilla erittäin tehokasta ja helppoa, jonka vuoksi toiminnallisen osan tuloksia oli helppo käsitellä.

Tekoälyn ja kuvantunnistuksen käyttö ohjelmoinnin tukena osoittautui erittäin helpoksi ja toi mielenkiintoa sen soveltamisessa ohjelmistokehityksessä. Kuvantunnistus on nykyään kaupallisessa käytössä monella eri toimialalla, ja opinnäytetyö vahvisti entisestään näkemystä siitä, että tulevaisuudessa tekoälyätoimintoja käytetään yhä monimuotoisemmin palveluiden tukena. Opinnäytetyö lisäsi ohjelmointitaitojani sekä antoi tietoa tekoälyn mahdollisuuksista sovelluskehityksessä.

Lähteet

Appen. (2019). *Computer Vision vs. Machine Vision — What's the Difference?* Computer Vision vs. Machine Vision — What's the Difference? https://appen.com/blog/computer-vision-vs-machine-vision/

Beyerer, J., León, F. P., & Frese, C. (2015). Machine vision: Automated visual inspection: Theory, practice and applications. In *Machine Vision: Automated Visual Inspection: Theory, Practice and Applications*. https://doi.org/10.1007/978-3-662-47794-6

Boucher, P. (2020). Artificial intelligence: How does it work, why does it matter, and what can we do about it? In *Panel for the Future of Science and Technology*.

Brownlee, J. (2019). *A Tour of Machine Learning Algorithms*. Machine Learning Algorithms. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Chakraborty, A. (2019). *Difference between Image Processing and Computer Vision - GeeksforGeeks*. https://www.geeksforgeeks.org/difference-between-image-processing-and-computer-vision/

Clearview Imaging. (2018). *The difference between computer vision and machine vision*. https://www.clearviewimaging.co.uk/blog/the-difference-between-computer-vision-and-machine-vision

Davies, E. R. (2017). Computer Vision: Principles, Algorithms, Applications, Learning: Fifth Edition. In *Computer Vision: Principles, Algorithms, Applications, Learning: Fifth Edition*.

Deepomatic. (n.d.). *Image Recognition : A Complete Guide - Deepomatic*. Retrieved February 8, 2021, from https://deepomatic.com/en/what-is-image-recognition

Devopedia. (2020). *Computer Vision*. https://devopedia.org/computer-vision

Fry, H., & Iso-Markku, J. (2019). *Hello world: Kuinka selviytyä algoritmien aikakaudella*. Bazar. https://www.ellibslibrary.com/reader/9789522797742

Garg, A. (2018). *Complete guide to Association Rules (1/2)*. Towards Data Science. https://towardsdatascience.com/association-rules-2-aa9a77241654

Google. (2020). *Vision AI | Derive Image Insights via ML | Cloud Vision API*. https://cloud.google.com/vision

Guru99. (n.d.). *Expert System in AI: What is, Applications & Example*. AI Tutorial. Retrieved February 9, 2021, from https://www.guru99.com/expert-systems-with-applications.html

Hornberg, A. (2017). Handbook of Machine and Computer Vision. In *Handbook of Machine*

and Computer Vision. https://doi.org/10.1002/9783527413409

JavaTpoint. (n.d.). *Knowledge Representation in Artificial Intelligence - Javatpoint*. Retrieved January 21, 2021, from https://www.javatpoint.com/knowledge-representation-in-ai

Joshi, P. (2017). Artificial Intelligence with Python. In *Artificial Intelligence with Uncertainty*.

Kananen, H., & Puolitaival, H. (2019). *Tekoäly : bisneksen uudet työkalut*. https://bisneskirjasto-almatalent-fi.ezproxy.hamk.fi/teos/BAXBBXATCBIED#/kohta:TEKO((c4)LY((20)-((20)Bisneksen((20)uudet((20)tyokalut/piste:tU

Kenton Will. (2020). *A Priori Probability Definition & Example*. https://www.investopedia.com/terms/a/apriori.asp

Kirzhner, E. (2018). *Machine Learning . Explanation of Collaborative Filtering vs Content Based Filtering*. Codeburst. https://codeburst.io/explanation-of-recommender-systems-in-information-retrieval-13077e1d916c

Koul, A., Ganju, S., & Kasam, M. (2019). Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow. In *Handbook of Medical Image Computing and Computer Assisted Intervention* (Vol. 8, Issue 5).

Lee, W.-M. (2019). Python® Machine Learning. In *Python® Machine Learning*. https://doi.org/10.1002/9781119557500

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the Dartmouth summer research project on artificial intelligence. *AI Magazine*, *27*(4).

Merriam-Webster. (2017). *Artificial Intelligence | Definition of Artificial Intelligence by Merriam-Webster*. Merriam-Webster Dictionary. https://www.merriam-webster.com/dictionary/artificial intelligence

Microsoft. (2019). *Cognitive Services—APIs for AI Developers | Microsoft Azure*. Azure Cloud. https://azure.microsoft.com/en-us/services/cognitive-services/

Ng, A. (2020). How China uses facial recognition to control human behavior. *CNET*. https://www.cnet.com/news/in-china-facial-recognition-public-shaming-and-control-go-hand-in-hand/

Prince, S. J. D. (2012). *Computer Vision: models, learning and inference*.

Reaktor, & Helsingin Yliopisto. (n.d.). *Neuroverkkojen periaatteet - Elements of AI*. Retrieved January 23, 2021, from https://course.elementsofai.com/fi/5/1

Richmond, A. (2020). *A Beginner's Guide To Computer Vision*.

https://towardsdatascience.com/a-beginners-guide-to-computer-vision-dca81b0e94b4

Sanastokeskus TSK ry. (1993). *Tietohuollon sanasto (TSK 20)*. https://termipankki.fi/tepa/fi/haku/tietämyskanta

Siukonen, T., & Neittaanmäki, P. (2019). *Mitä tulisi tietää tekoälystä | Ellibs Lukuohjelma*. Docendo. https://www.ellibslibrary.com/reader/9789522916549

Szeliski, R. (2009). Computer Vision : Algorithms and Applications. In *Computer Vision : Algorithms and Applications*.

Tazehkandi, A. A. (2018). *Hands-On Algorithms for Computer Vision*. https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=5482823&ppg=212

Tilastokeskus. (n.d.). *Diskreetti ja jatkuva muuttuja. Muuttujan jakauma*. Johdastus Tilastotieteeseen. Retrieved January 26, 2021, from https://tilastokoulu.stat.fi/verkkokoulu_v2.xql?course_id=tkoulu_tilaj&lesson_id=2&subject_id=5&page_type=sisalto

Tryolabs. (n.d.). *An Introductory Guide to Computer Vision*. Retrieved February 5, 2021, from https://tryolabs.com/resources/introductory-guide-computer-vision/

Veisdal, J. (2019). *The Birthplace of AI*. https://medium.com/cantors-paradise/the-birthplace-of-ai-9ab7d4e5fb00

Warwick, K. (2013). Artificial intelligence: The basics. In *Artificial Intelligence: The Basics*. https://doi.org/10.4324/9780203802878

Wetsman, N. (2021). Pixel phones will be able to read your heart rate with their cameras. *The Verge*. https://www.theverge.com/2021/2/4/22265004/google-pixel-camera-heart-rate-breathing-rate-sensor

Zhongzhi, S. (2011). *Advanced Artificial Intelligence*. World Scientific Publishing Company. https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/detail.action?docID=840558.

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyössä ei syntynyt luottamuksellisia tietoja tai salassa pidettäviä aineistoja. Toiminnallisessa osassa saadut tulokset tallennettiin paikalliselle koneelle sekä varmuuskopioitiin säännöllisin ajoin pilvipalveluun. Opinnäytetyössä luotu ohjelmakoodi sekä tulokset tallennettiin lopuksi GitHub-versionhallintajärjestelmään.

Liite 2: Google Vision API -ohjelmakoodi

```
import io
import os
from google.cloud import vision_v1 as vision
import pandas as pd
import pathlib

local_image = 'full file path here.jpg'
filename = os.path.basename(local_image)
csv_file = "google-results.csv"
file = pathlib.Path(csv_file)

def save_csv(df_new):
    if file.exists ():
        try:
            df_from_csv = pd.read_csv(csv_file)
            frames = [df_from_csv, df_new]
            df_edited = pd.concat(frames)
            df_edited.to_csv(csv_file, index=False)

        except pd.errors.EmptyDataError:
            df_new.to_csv(csv_file, index=False)

    else:
        print('No file, creating new')
        df_new.to_csv(csv_file, index=False)

def labels(image, client):
    response = client.label_detection(image=image)
    labels = [label.description for label in response.label_annotations]
    return labels

def objects(image, client):
    response = client.object_localization(image=image)
    objects = [object.name for object in response.localized_object_annotati
ons]
    return objects

def landmarks(image, client):
    response = client.landmark_detection(image=image)
    landmarks = [landmark.description for landmark in response.landmark_ann
otations]
    return landmarks

def logos(image, client):
    response = client.logo_detection(image=image)
    logos = [logo.description for logo in response.logo_annotations]
    return logos

def safesearch(image, client):
    likelihood_name = ('UNKNOWN', 'VERY_UNLIKELY', 'UNLIKELY', 'POSSIBLE',
                       'LIKELY', 'VERY_LIKELY')
    response = client.safe_search_detection(image=image)
    filter_violence = dict(violence='{}'.format(likelihood_name[response.sa
fe_search_annotation.violence]))
    violence_value = filter_violence['violence']
    return violence_value

def faces(image, client):
    response = client.face_detection(image=image)
```

```
    face_count = len(response.face_annotations)
    return face_count

def web(image, client):
    response = client.web_detection(image=image)
    best_guess = [label.label for label in response.web_detection.best_gues
s_labels]
    return best_guess

def texts(image, client):
    response = client.text_detection(image=image)
    texts = [text.description for text in response.text_annotations]
    return texts

def main():
    client = vision.ImageAnnotatorClient()
    with io.open(local_image, 'rb') as image_file:
        content = image_file.read()
    image = vision.types.Image(content=content)
    df = pd.DataFrame()
    df['Filename'] = [filename]
    df['Labels'] = [labels(image,client)]
    df['Objects'] = [objects(image,client)]
    df['Websearch'] = [web(image,client)]
    df['Logos'] = [logos(image,client)]
    df['Violence'] = [safesearch(image,client)]
    df['Faces'] = [faces(image,client)]
    df['Landmarks'] = [landmarks(image,client)]
    df['Texts'] = [texts(image,client)]
    save_csv(df)

if __name__ == "__main__":
    main()
```

Liite 3: Microsoft Computer Vision API -ohjelmakoodi

```
import os
from azure.cognitiveservices.vision.computervision import ComputerVisionCli
ent as vision
from msrest.authentication import CognitiveServicesCredentials as credentia
ls
import pandas as pd
import pathlib

subscription_key = " subscription key "
endpoint = " endpoint-url "

local_image = full image file path.jpg'
filename = os.path.basename(local_image)
csv_file = "microsoft-results.csv"
file = pathlib.Path(csv_file)

def save_csv(df_new):
    if file.exists ():
        try:
            df_from_csv = pd.read_csv(csv_file)
            frames = [df_from_csv, df_new]
            df_edited = pd.concat(frames)
            df_edited.to_csv(csv_file, index=False)

        except pd.errors.EmptyDataError:
            df_new.to_csv(csv_file, index=False)

    else:
        print('No file, creating new')
        df_new.to_csv(csv_file, index=False)

def tags(client):
    image = open(local_image, "rb")
    response = client.tag_image_in_stream(image)
    tags = [tag.name for tag in response.tags]
    return tags

def objects(client):
    image = open(local_image, "rb")
    response = client.detect_objects_in_stream(image)
    objects = [obj.object_property for obj in response.objects]
    return objects

def describe(client):
    image = open(local_image, "rb")
    response = client.describe_image_in_stream(image)
    text = [caption.text for caption in response.captions]
    return text

def category(client):
    image = open(local_image, "rb")
    features = ["categories"]
    response = client.analyze_image_in_stream(image, features)
    categories = [category.name for category in response.categories]
    return categories

def logos(client):
    image = open(local_image, "rb")
    features = ["brands"]
```

```
    response = client.analyze_image_in_stream(image, features)
    logos = [brand.name for brand in response.brands]
    return logos

def adult(client):
    image = open(local_image, "rb")
    features = ["adult"]
    response = client.analyze_image_in_stream(image, features)
    adult = response.adult.is_adult_content
    return adult

def faces(client):
    image = open(local_image, "rb")
    features = ["faces"]
    response = client.analyze_image_in_stream(image, features)
    face_count = len(response.faces)
    return face_count

def domain(client):
    image = open(local_image, "rb")
    response = client.analyze_image_by_domain_in_stream("landmarks", image)
    landmarks = [landmark["name"] for landmark in response.result["landmark
s"]]
    return landmarks

def texts(client):
    image = open(local_image, "rb")
    response = client.recognize_printed_text_in_stream(image)
    texts = [word.text for region in response.regions for line in region.li
nes for word in line.words]
    return texts

def main():
    client = vision(endpoint, credentials(subscription_key))
    df = pd.DataFrame()
    df['Filename'] = [filename]
    df['Tags'] = [tags(client)]
    df['Objects'] = [objects(client)]
    df['Describe'] = [describe(client)]
    df['Category'] = [category(client)]
    df['Logos'] = [logos(client)]
    df['Faces'] = [faces(client)]
    df['Landmarks'] = [domain(client)]
    df['Texts'] = [texts(client)]
    save_csv(df)

if __name__ == "__main__":
    main()
```

Liite 4: Google Vision API testitulokset

Filename	Labels	Objects	Websearch	Logos	Violence	Faces	Landmarks	Texts
airport_1.jpg	['Trousers', 'Custom	['Person', 'Person', '	['arrival in the airport']	['Tequila Herradura	VERY_UNLIKELY	6	[]	["International arrivals\nkernational Arrivais\nlaters
airport_2.jpg	['Architecture', 'Buil	['Person', 'Person', '	['incheon international air	[]	VERY_UNLIKELY	1	[]	['Open to All Travellers...\nSleep\nSnacks\nMassage
arrest.jpg	['Trousers', 'Sky', 'Hi	['Person', 'Person', '	['protest']	[]	UNLIKELY	8	[]	['TROPOLITAN\nPOLICE\n3503052592\n2648\nPOLCE
asema.jpg	['House', 'Plant', 'Bu	['Chair', 'Chair', 'Cha	['real estate']	[]	VERY_UNLIKELY	0	[]	['Amtrak\nSanford\nTICKET\nOFFICE\nRustpoms\n\
asema2.jpg	['Building', 'Luggage	['Person', 'Person', '	['snapshot']	[]	VERY_UNLIKELY	2	[]	['é›»ãƒ—ãƒ©ã‚¶\nstion Plaza\näą¸å·´9\nKiosk\n', 'é›
autopalo.jpg	['Workwear', 'High-\	['Person', 'Person', '	['Traffic collision']	[]	VERY_UNLIKELY	0	[]	[]
axe.jpg	['Sky', 'Shirt', 'Cap', '	['Person', 'Person', '	['tree']	['Libertyville High S	UNLIKELY	1	[]	[]
beach.jpg	['Water', 'Cloud', 'Sk	['Person', 'Person', '	['sea']	[]	VERY_UNLIKELY	0	[]	[]
boxing.jpg	['Headgear', 'Sports	['Shorts', 'Shorts', 'P	['Boxing']	[]	UNLIKELY	6	[]	['TA\n', 'TA']
capitol_1.jpg	['Daytime', 'Sky', 'W	['Person', 'Person', '	['california state capitol m	[]	VERY_UNLIKELY	3	['California State Capitol M	['CANCEL\nANIMAL\n* AG\nSTOP\nFACTORY\nFARM
casino_1.jpg	['Clothing', 'Table', '	['Person', 'Person', '	['casino']	[]	VERY_UNLIKELY	6	[]	[]
colosseum.jpg	['Sky', 'Cloud', 'Land	['Building', 'Building	['colosseum']	[]	VERY_UNLIKELY	0	['Colosseum']	[]
cry.jpg	['Forehead', 'Nose',	['Glasses', 'Person']	['cocaine addiction']	[]	POSSIBLE	1	[]	[]
delivery.jpg	['Building', 'Plant', '\	['Packaged goods', '	['United Parcel Service']	[]	VERY_UNLIKELY	1	[]	['Bowery\natagonia\nSPANK\nTM\nNE ar\nCHPOME
delivery_2.jpg	['Jeans', 'Vehicle', 'C	['Jeans', 'Person', 'B	['car']	[]	VERY_UNLIKELY	1	[]	['INTCO\nINTC\n&01\nDisposable Face Mask\nMate
etiketti.png	['Font', 'Material pr	[]	['food labels']	[]	VERY_UNLIKELY	0	[]	['Nutrition Facts Amount Per Serving\nServing size:
F1.jpg	['Tire', 'World', 'Veh	['Person', 'Person', '	['Formula 1']	[]	VERY_UNLIKELY	0	[]	['TARAPITEAT HEure\nMa\n125) Ye\nVEARS\n', 'TAR
festari.jpg	['White', 'Light', 'Prc	['Person', 'Person', '	['stage']	['Youngstown State	VERY_UNLIKELY	1	[]	['PDOLYTE\nTUS\nRADIO\nROCK\nà°¶à±à°°à±€à°¾\r
fire.jpg	['Muscle', 'Motor ve	['Person', 'Helmet',	['home fire safety']	[]	VERY_UNLIKELY	0	[]	['601\nPACLIC\nKEVLA\nME\n', '601', 'PACLIC', 'KEVL
futis.jpg	['Fan', 'Player', 'Red	['Person', 'Person']	['wsw flares']	['NRMA', 'Le Bon Ma	UNLIKELY	4	[]	['ng your\nall back\nNRMA\nINSURANCE\n', 'ng', 'yo
gym_1.jpg	['Joint', 'Barbell', 'Sh	['Person', 'Shoe', 'Pe	['gym']	[]	UNLIKELY	2	[]	['å£\n', 'å£']
joy.jpg	['Water', 'Smile', 'Pl	['Swimwear', 'Swim	['spike ball water']	[]	VERY_UNLIKELY	3	[]	['SPIKE\nTIVE\nSPIKE\nBALL\n', 'SPIKE', 'TIVE', 'SPIKE
keikka_yo.jpg	['Blue', 'Concert', 'Ei	['Person']	['aesthetic wallpaper stage	[]	VERY_UNLIKELY	0	[]	[]
kick_boxing.jpg	['Hand', 'Shorts', 'Gl	['Shorts', 'Person', 'S	['men fighting']	[]	UNLIKELY	2	[]	['rwiNS\nORMUR\nMUR\nGHTER\nVENUM\n', 'rwiN
knife.jpg	['Food', 'Tableware'	['Kitchen knife', 'Pa	['Food']	[]	VERY_UNLIKELY	0	[]	[]
konsertti.jpg	['Furniture', 'Chair',	['Person', 'Person', '	['frederick p rose hall']	[]	VERY_UNLIKELY	2	[]	[]
liberty.jpg	['Sky', 'Cloud', 'Dayt	['Sculpture']	['statue of liberty national	[]	UNLIKELY	0	['Statue of Liberty Nation	[]
logos.jpg	['Font', 'Rectangle',	[]	['multiple logos']	['YouTube', 'Google	VERY_UNLIKELY	0	[]	['You Tube\nSU\nskype\nGoogle\nWIKIPEDIA\nThe
mall.jpg	['Motor vehicle', 'Cu	[]	['car']	[]	VERY_UNLIKELY	0	[]	['GLG\nAMSUNG\nDIGITAL\nJVC\nndou\nSAM SOFT
mcdonalds.jpg	['Wheel', 'Automoti	['Car', 'Car', 'Buildin	["McDonald's"]	["McDonald's", "Mc	VERY_UNLIKELY	0	[]	["Av. Macheros\nMcDanald's\nsalte\nTUYA\n000\nI
museo_1.jpg	['Dog', 'Picture fram	['Person', 'Person', '	['droste effect']	[]	VERY_UNLIKELY	1	[]	['Delia\nDama Azul\n', 'Delia', 'Dama', 'Azul']
museo_2.jpg	['Musical instrumen	['Person', 'Person', '	['art exhibition']	[]	VERY_UNLIKELY	2	[]	['Â© Berlin-Motive.de\n', 'Â©', 'Berlin-Motive.de']
office_1.jpg	['Computer', 'Table'	['Laptop', 'Laptop', 'I	['management office']	[]	VERY_UNLIKELY	4	[]	['mont\nMade B\nDesign\nTools\nPORTE\n', 'mont'
office_2.jpg	['Computer', 'Table'	['Person', 'Person', '	['you x ventures oalh2moj	[]	VERY_UNLIKELY	4	[]	[]
protest_3.jpg	['Cloud', 'Shorts', 'Sk	['Shorts', 'Shorts', 'P	['crowd']	[]	VERY_UNLIKELY	1	[]	['FUCK\nthe\nPOLICE\nDEFUND\nPOLICE\nABOLISH
protest_4.jpg	['Chair', 'Building', 'S	['Person', 'Person', '	['city']	[]	VERY_UNLIKELY	3	[]	['EOLICE\n', 'EOLICE']
riot_1.jpg	['Clothing', 'Crowd',	['Person', 'Person', '	['crowd']	['Hammerskins']	VERY_UNLIKELY	4	[]	['PILUS\nPOLICE\nPOLKS\nPOLIE\nPALKE\nPILZE\nP
riot_2.jpg	['Sky', 'Cloud', 'Heac	['Person', 'Person', '	['violence hd']	[]	UNLIKELY	10	[]	['IVOL IO\n', 'IVOL', 'IO']
stadium.jpg	['Cloud', 'Sky', 'Builc	[]	['yankee stadium']	['Ford Motor Compa	VERY_UNLIKELY	0	['Yankee Stadium']	['æ—¥\nBankof America\nPITCH SPEED\nBALLS 16 S
station.jpg	['Shorts', 'Standing',	['Person', 'Person', '	['grand central terminal']	[]	VERY_UNLIKELY	2	['Grand Central Terminal']	['HARLEM LINE DEPARTURES\nTIME TRK DESTINATIO
subway.jpg	['Footwear', 'Trouse	['Person', 'Person', '	['nyc subway panhandlers'	[]	VERY_UNLIKELY	3	[]	['Dondle\nidoor\n', 'Dondle', 'idoor']
sulkkis1.jpg	['Sports equipment	['Person', 'Person', '	['team sport']	[]	VERY_UNLIKELY	1	[]	['LANDERSPIEL U17\nFraport. Die Airport Manager.\r
suojatie.jpg	['Urban design', 'Cit	[]	['urban area']	[]	VERY_UNLIKELY	0	[]	['---\n', '---']
workers.jpg	['Sky', 'Cloud', 'Mast	['Person', 'Person', '	['hsinchu city, rainbow bri	[]	VERY_UNLIKELY	0	['Hsinchu City, Rainbow B	[]

Liite 5: Microsoft Computer Vision API testitulokset

Filename	Tags	Objects	Describe	Category	Logos	Faces	Landmarks	Texts
airport_1.jpg	['clothing', 'ceiling', 'pers	['person', 'person', 'pers	['a group of people at a r	['indoor_']	[]	0 []	[]	['e', 'International', 'arrivals', 'FREE/GRATIS/GRATUITFRE', 'T/F
airport_2.jpg	['building', 'person', 'indi	['person', 'person', 'pers	['a group of people in a l	['indoor_', 'outdoor_']	[]	0 []	[]	['UTY', 'FREE', 'NIKE%GOLF', '4th', 'EntrancÃ¸', '1209', 'DUTY', 'FR
arrest.jpg	['person', 'outdoor', 'sky'	['person', 'person', 'pers	['a group of people in un	['people_group']	[]	1 []	[]	['à¸€', 'à¸€', '10', 'àº ']
asema.jpg	['furniture', 'indoor', 'cha	['chair']	['a building with benche	['indoor_']	[]	0 []	[]	[]
asema2.jpg	['clothing', 'person', 'foo	['person', 'plant', 'handb	['a group of people walk	['indoor_', 'outdoor_']	[]	0 []	[]	[]
autopalo.jpg	['firefighter', 'outdoor', '	['Land vehicle', 'person',	['firefighters putting out	['abstract_', 'others_', 'ou	[]	0 []	[]	['10']
axe.jpg	['tree', 'person', 'outdoo	['person']	['a person holding a chai	['others_', 'outdoor_']	[]	0 []	[]	[]
beach.jpg	['sky', 'water', 'outdoor',	['person', 'person', 'person	['a group of people in th	['outdoor_ocean beach']	[]	0 []	[]	[]
boxing.jpg	['sport', 'sports equipme	['football helmet', 'foott	['a boxing match in a ring	['others_', 'outdoor_']	[]	2 []	[]	[]
capitol_1.jpg	['tree', 'outdoor', 'persor	['person', 'person', 'pers	['a group of people holdi	['building_', 'outdoor_']	[]	0 []	[]	['STOP', 'a', 'oat', 'FACTORY', 'FARMS']
casino_1.jpg	['person', 'games', 'gamb	['chair', 'person', 'person	['a group of people arou	['people_many']	[]	4 []	[]	[]
colosseum.jpg	['building', 'sky', 'outdoo	[]	['a large stone building v	['building_', 'others_', 'ou	[]	0 ['Colosseum']	[]	[]
cry.jpg	['indoor', 'human face', 'j	['Glasses', 'person']	['a person with the hand	['people_']	[]	1 []	[]	[]
delivery.jpg	['building', 'outdoor', 'str	['traffic light', 'traffic ligh	['a person carrying a cart	['abstract_', 'building_', '	[]	0 []	[]	['Ã¶M08', 'o', 'o']
delivery_2.jpg	['person', 'outdoor', 'text	['book', 'jeans', 'person']	['a person loading boxes	['abstract_', 'others_', 'ou	[]	0 []	[]	[]
etiketti.png	['text', 'screenshot', 'new	[]	['text, table']	['text_', 'text_sign']	[]	0 []	[]	['Nutrition', 'Facts', 'Amount', 'Per', 'Serving', 'Store', 'bar', '(40g
F1.jpg	['person', 'vehicle', 'land	[]	['a person standing in frc	['outdoor_']	[]	0 []	[]	[]
festari.jpg	['person', 'concert', 'mus	[]	['a band playing on a stag	['people_crowd']	[]	0 []	[]	[]
fire.jpg	['fire', 'smoke', 'hydrant',	['person']	['a firefighter spraying w	['outdoor_']	[]	0 []	[]	[]
futis.jpg	['smoke', 'fireworks', 'crc	['person']	['a large crowd of people	['people_crowd']	[]	0 []	[]	[]
gym_1.jpg	['person', 'road', 'sport', '	['Dumbbell', 'Dumbbell',	['a man working out in a	['outdoor_', 'people_cro	[]	0 []	[]	[]
joy.jpg	['outdoor', 'water', 'sky',	['person', 'person', 'pers	['Yukie Nakayama and w	['outdoor_', 'people_swi	[]	1 []	[]	[]
keikka_yo.jpg	['concert', 'music', 'live n	[]	['a crowd of people at a c	['outdoor_', 'people_sho	[]	0 []	[]	[]
kick_boxing.jpg	['person', 'dance', 'boxin	['person', 'person']	['a couple of men fightin	['others_']	[]	1 []	[]	[]
knife.jpg	['table', 'indoor', 'woode	['Food', 'Kitchen utensil'	['a cutting board with ve	['others_']	[]	0 []	[]	[]
konsertti.jpg	['indoor', 'furniture', 'tab	['chair', 'chair']	['a group of people sittir	['outdoor_', 'people_sho	[]	0 []	[]	[]
liberty.jpg	['statue', 'sculpture', 'out	[]	['a statue of a person hol	['object_sculpture']	[]	0 []	[]	[]
logos.jpg	['text', 'design', 'screens	['golf ball']	['logo, company name']	['others_', 'text_sign']	['Google', 'eBay']	0 []	[]	['Google', 'flickrâec', 'digg', 'WIKIPEDIA']
mall.jpg	['text', 'indoor', 'building	[]	['a large group of people	['indoor_marketstore']	[]	0 []	[]	[]
mcdonalds.jpg	['sky', 'outdoor', 'road', 't	['car', 'car', 'car']	['a street corner with car	['building_street']	["McDonald's"]	0 []	["Av.', 'Macheros', "\'vDona', "'s", "McDonald's"]	
museo_1.jpg	["floor', 'indoor', 'dog', 'c	['person', 'person', 'dog']	['a framed picture of a pe	['others_']	[]	0 []	['DamaAzul']	
museo_2.jpg	['indoor', 'floor', 'musical	['string instrument', 'per	['a man and a woman pla	['indoor_']	[]	0 []	["", 'o', '40W', '08']	
office_1.jpg	['person', 'indoor', 'lapto	['person', 'person', 'pers	['a group of people sittir	['people_many']	[]	1 []	[]	[]
office_2.jpg	['person', 'indoor', 'comp	['person', 'person', 'pers	['a group of people sittir	['others_', 'people_many	[]	0 []	[]	[]
protest_3.jpg	['person', 'outdoor', 'sky'	['Luggage and bags', 'per	['a person holding a sign	['people_crowd', 'people	[]	0 []	['IT', 'ALLI!', 'POLICING', 'FLICK', 'Nxpo;', 'a,dCkUVEg', 'i.creases	
protest_4.jpg	['person', 'clothing', 'ceil	['person', 'person', 'chair	['a group of people in a t	['others_', 'outdoor_', 'pe	[]	1 []	[]	
riot_1.jpg	['person', 'outdoor', 'clot	['person', 'person', 'pers	['a group of people in mi	['people_crowd']	[]	0 []	[]	[]
riot_2.jpg	['sky', 'outdoor', 'person'	['person', 'person', 'pers	['a group of people runn	['outdoor_', 'people_gro	[]	4 []	[]	[]
stadium.jpg	['sky', 'baseball', 'grass', '	['Scoreboard']	['a baseball stadium with	['outdoor_', 'outdoor_sp	[]	0 []	['ARMT', 'ON', '@pepsi', 'Bank', 'of', 'America', 'o', 'e', 'oo', 'o', 'e	
station.jpg	['text', 'clothing', 'persor	['footwear', 'person', 'pe	['a group of people stanc	['abstract_', 'outdoor_']	[]	0 []	['?iiu', 'HARLEM', 'LINE', 'DEPARTURES', 'TIME', 'TRK', 'DESTINA	
subway.jpg	['person', 'musical instru	['headwear', 'Musical ins	['a group of people on a	[]	[]	0 []	[]	[]
sulkkis1.jpg	['person', 'sports equipn	['person', 'person']	['a group of people playi	[]	['Yonex']	0 []	['Fraport.', 'Die', 'Airport', 'Manager.', 'YONEX']	
suojatie.jpg	[]	[]	['a large group of people	['abstract_', 'abstract_ne	[]	0 []	[]	
workers.jpg	['construction', 'sky', 'sca	['person']	['a group of men on a bu	['outdoor_', 'outdoor_str	[]	0 []	[]	['ÙŠÙ"", 'ÙŠÙ....', 'ÙŠÙŠ', ' ', 'ÙŠØ±ÙŠ', 'ÙŠÙŠØ¤', 'Ø³Ø±Ø±Ø±Ø±