SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN ALA

AUTOMAATIOLAITTEIDEN TUNNISTAMINEN ANDROID-SOVELLUKSELLA

TEKIJÄ Mikko Möttönen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Mikko Möttönen	
Työn nimi Automaatiolaitteiden tunnistaminen Android-sovelluksella	
Päiväys 17.12.2025	Sivumäärä / Liitteet 24
Ohjaaja(t) Jussi Koistinen, Janne Koponen	
Toimeksiantaja/Yhteistyökumppani(t) ATC-Automation	

Tämän opinnäytetyön tavoitteena oli kehittää Android-mobiilisovellus, jonka avulla teollisuusympäristössä käytettäviä automaatiolaitteita voidaan tunnistaa kuvantunnistuksen avulla. Tunnistuksen jälkeen käyttäjälle avautuu kyseistä laitetta koskeva laitekortti, joka voi sisältää esimerkiksi teknisiä tietoja, ohjeita ja muuta laitteen käyttöön liittyvää materiaalia. Toimeksiantajan toiveena oli, että laitekortti olisi mahdollista hakea nopeasti ja vaivattomasti kuvaamalla se, sen sijaan että käyttäjän tarvitsisi selata pitkää laiteluetteloa käytännön työympäristössä.

Sovellus toteutettiin Android Studiolla käyttäen Java- ja Kotlin-ohjelmointikieliä Kuvantunnistus perustui MobileNetV2-arkkitehtuuriin pohjautuvaan konvoluutioneuroverkkoon, joka koulutettiin TensorFlow-kirjastolla ja otettiin käyttöön Android-sovelluksessa TensorFlow Lite -kirjaston avulla. Lisäksi projektissa hyödynnettiin Googlen tarjoamia pilvipalveluita, kuten Firebasea ja Firestoragea, sekä Google Colab -ympäristöä mallien koulutukseen ja testaamiseen.

Työn lopputuloksena syntyi sovellusprototyyppi, joka vastaa asetettuihin tavoitteisiin ja tarjoaa hyvän pohjan jatkokehitykselle. Sovellusta voidaan kehittää edelleen tarpeiden mukaan esimerkiksi parantamalla tunnistuksen tarkkuutta, lisäämällä uusia laitteita tai laajentamalla sovelluksen toiminnallisuuksia.

Avainsanat

SISÄLTÖ

KUVALUETTELO

SANASTO

Android	Android on käyttöjärjestelmä, joka on kehitetty pääasiassa mobiililaitteille, kuten älypuhelimille ja tablet-laitteille. Se perustuu Linux-ytimen ja avoimen ohjelmistoon ja sitä ylläpitää ja kehittää Google. Android tarjoaa käyttäjille monipuoliset toiminnot ja sovellukset, kuten internet-selaamisen, viestinnän, sosiaalisen median käytön ja paljon muuta.
Git	Git on hajautettu versionhallintajärjestelmä, joka on suunniteltu helpottamaan ohjelmistokehitystyötä. Se mahdollistaa monien kehittäjien yhteistyön saman projektin parissa ja auttaa hallitsemaan muutoksia projektin koodissa.
IDE	Integrated Development Environment, eli ohjelmistoympäristö. Sisältää debuggerin, kääntäjän, verkkopalvelimen, komentotulkkipäätteen, kaikki samassa.
Java	Java on yleiskäyttöinen, korkean tason ohjelmointikieli, joka on suunniteltu olemaan riippumaton alustasta. Se tunnetaan erityisesti sen kyvystä luoda monialustaisia sovelluksia, mikä tarkoittaa, että samaa koodia voidaan suorittaa eri käyttöjärjestelmämissä ilman suuria muutoksia. Javaa käytetään laajasti erilaisissa sovelluksissa, kuten web-sovelluksissa, mobiilisovelluksissa, tietokantajärjestelmissä ja pelikehityksessä.
Kotlin	Kotlin on monialustainen ja staattisesti tyypitetty ohjelmointikieli, joka on täysin yhteensopiva Javan kanssa. Kielen tyyppipäätelmän ansiosta sen syntaksi on tiiviimpi, mutta se pystyy silti käyttämään Javan luokkakirjastoa.
OCR	Optical Character Recognition (OCR), eli Optinen merkintätunnistus on teknologia, joka mahdollistaa kirjoitetun, käsin kirjoitetun tai painetun tekstin muuntamisen kuvista konekoodatuksi tekstiksi.

1 JOHDANTO

Tässä opinnäytetyössä tutkitaan mahdollisuutta tehdä mobiilisovellus teollisuuden kunnossapitoyrityksen tarpeisiin, jolla voitaisiin hakea helpommin automaatiolaitteiden asennus-, kalibrointi- ja muut ohjeet kunnossapitotöitä varten. Projektin toimeksiantajana on Kuopiolainen ATC-Automation -niminen yritys, joka on erikoistunut energia-, petrokemian- sekä paperiteollisuuden kunnossapitoratkaisuihin.

Opinnäytetyön aihe on saanut alkunsa aiemmasta yhteistyöstä toimeksiantajan kanssa, kun samalle yritykselle toteutettiin ohjelmistotuotanto-kurssin yhteydessä ryhmätyönä Android-pohjainen tarkastuslistasovellus. Kurssin päätyttyä toimeksiantaja ilmaisi kiinnostuksensa sovelluksen jatkokehittämiseen ja esitti idean sovelluksesta, jonka avulla teollisuusympäristössä olevia laitteita voitaisiin tunnistaa kuvantunnistuksen avulla.

Työn tavoitteena on tutkia, onko tällainen sovellus kuvantunnistuksella mahdollista toteuttaa realistisella tavalla ja tehdä prototyyppi, mitä kokeillaan muutamilla eri laitteilla.

2 KONEOPPIMINEN

Koneoppiminen on tekoälyn osa-alue, joka on olennainen osa nykyaikaista sovelluskehitystä ja tarjoaa lukuisia mahdollisuuksia älykkäiden sovellusten kehittämiseen. Tässä osiossa käsitellään koneoppimisen peruskäsitteitä, erilaisia koneoppimisen sovellusalueita sekä yleisiä menetelmiä ja algoritmeja, joita käytetään sovelluskehityksessä. (Ojanperä, 2023)

2.1 Koneoppimismalli

Koneoppimisessa mallilla tarkoitetaan datan avulla opetettua rakennetta, joka tekee ennusteita tai päätöksiä uusista syötteistä. Tässä työssä malli on neuroverkkoon perustuva luokittelija, jota käytetään laitteiden tunnistamiseen kuvista. (Russell & Norvig, 2022)

2.2 Koneoppimisen historiaa

Koneoppimisen juuret ulottuvat 1940-luvulle, jolloin tutkijat alkoivat pohtia, voisivatko koneet oppia suorittamaan älykkäitä toimintoja ihmisen tavoin. Yksi keskeisimmistä varhaisista tapahtumista oli vuonna 1956 järjestetty Dartmouthin kesäseminaari, jossa joukko alan tutkijoita kokoontui tarkastelemaan tekoälyn tutkimuksen tulevaisuutta. Seminaarissa esiteltiin ensimmäistä kertaa termi tekoäly, jonka toi esiin John McCarthy. Tapahtumaa pidetään yleisesti merkittävänä lähtökohtana tekoälytutkimuksen kehitykselle. (Ojanperä, 2023)

2.3 Koneoppimisen määritelmä

Koneoppiminen on tekoälyn osa-alue, jossa tietokoneohjelmat suunnitellaan oppimaan datasta ja parantamaan suoritustaan tietyissä tehtävissä ilman, että ohjelmoija määrittelee tarkkoja sääntöjä jokaiseen tilanteeseen. Koneoppimisen keskeinen piirre on kyky yleistää havaittuja kuvioita ja riippuvuuksia uuteen, aiemmin näkemättömään dataan, mikä mahdollistaa esimerkiksi luokittelun ja ennustamisen. Tätä lähestymistapaa käytetään laajalti eri sovelluksissa, kuten kuvantunnistuksessa ja luonnollisen kielen käsittelyssä. (Russell & Norvig, 2022)

2.4 Neuroverkot ja konvoluutioneuroverkot

Keinotekoiset neuroverkot ovat koneoppimismalleja, jotka koostuvat toisiinsa kytketyistä solmuista eli neuroneista. Neuroverkot oppivat säätämällä solmujen välisiä painoja datan avulla, mikä mahdollistaa monimutkaisten kuvioiden tunnistamisen. Syväoppimisessa neuroverkkoja kasvatetaan useampaan kerrokseen, jolloin ne pystyvät oppimaan monimutkaisempia piirteitä datasta kuin yksikerroksiset menetelmät. (IBM, 2025a)

Konvoluutioneuroverkot (CNN) ovat neuroverkkoarkkitehtuureja, jotka on suunniteltu erityisesti kuvadataan. Ne hyödyntävät konvoluutiokerroksia, jotka oppivat automaattisesti tunnistamaan paikallisia piirteitä, kuten reunoja ja muotoja. CNN:ien avulla voidaan ratkaista tehokkaasti kuvapohjaisia luokitteluongelmia, kuten esineiden ja laitteiden tunnistusta kuvista. (IBM, 2025b)

MobileNetV2 on kevytrakenteinen CNN-arkkitehtuuri, joka on suunniteltu mobiililaitteisiin. Se tarjoaa hyvän kompromissin tunnistustarkkuuden ja laskentatehokkuuden välillä. MobileNetV2 soveltuu erinomaisesti Android-sovelluksiin, joissa halutaan suorittaa reaaliaikaista kuvantunnistusta rajoitetulla laitteistolla. (IBM, 2025b)

2.5 Ohjattu oppiminen

Ohjatussa oppimisessa kone opetetaan tunnistamaan elementtejä esimerkiksi kuvista tai teksteistä siten, että sille on ennalta määritelty luokat tai mallit, joita sen tulee oppia tunnistamaan. Ennusteiden muodostaminen perustuu opetusdataan, jossa malli pyrkii oppimaan selittävien ja selitettävien muuttujien välisen riippuvuuden. Yleisesti käytettyjä menetelmiä ovat regressio, jossa ennustetaan jatkuvia arvoja, sekä luokittelu, jossa data luokitellaan tiettyihin luokkiin. (Ojanperä, 2023; Pietikäinen & Silvén, 2021)

2.6 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa kone analysoi dataa ilman valmiiksi määriteltyjä luokkia tai malleja. Malli pyrkii löytämään datasta rakenteita, kuvioita tai ryhmiä itsenäisesti. Tämä lähestymistapa on erityisen hyödyllinen silloin, kun aineistoa ei ole merkitty tai luokiteltu etukäteen. Tyypillisiä menetelmiä ovat esimerkiksi dimensionaalisuuden vähentäminen ja ryvästys (clustering), joita voidaan hyödyntää esimerkiksi videoiden suositusjärjestelmissä tai kuluttajien käyttäytymisen analysoinnissa. Tunnettuja ryvästysmenetelmiä ovat muun muassa k-means ja hierarkkinen ryvästys. On tärkeää osata tulkita ryvästystuloksia oikein ja soveltaa niitä käytännön ongelmiin (Ojanperä, 2023; Pietikäinen & Silvén, 2021)

2.7 Vahvistusoppiminen

Kone oppii kokeilemalla ja virheistä. Tässä oppimisessa kone saa positiivista palautetta oikeista valinnoista ja negatiivista palautetta virheistä, ja se oppii näiden palautteiden perusteella. Vahvistusoppimista käytetään tilanteissa, joissa ei ole valmiiksi luokiteltua dataa. Tavoitteena on oppia toimintastrategia, josta saadaan eniten positiivista palautetta. Yksi suosittu menetelmä on Q-oppiminen, joka perustuu arvioimaan odotettua palautetta eri toimintavaihtoehdoista. Vahvistusoppimista käytetään muun muassa peleissä, logistiikassa ja robotiikassa. (Ojanperä, 2023; Pietikäinen & Silvén, 2021)

3 MOBIILIKEHITYSYMPÄRISTÖT JA OHJELMOINTIKIELET

Toimeksiantajalla on käytössään erilaisia mobiililaitteita, joissa voi olla eri käyttöjärjestelmäversioita, erikokoiset näytöt ja muita ominaisuuksia. Sovelluksen on oltava skaalautuva ja toimittava saumattomasti jokaisella laitteella, jotta se näyttää järkevältä ja toimii asianmukaisesti. Yrityksellä ei kuitenkaan ole käytössä iOS-laitteita, eli Applen valmistamia iPhoneja tai iPadeja, eikä ainakaan tällä hetkellä ole tarvetta iOS-tuelle.

Nykyisin mobiiliohjelmoinnin ohjelmointikieliksi ja kehitysympäristöiksi on olemassa useita erilaisia vaihtoehtoja. Seuraavassa esitellään joitakin näistä keskeisimmistä vaihtoehdoista.

3.1 Flutter

Flutter on Googlen kehittämä avoimen lähdekoodin SDK (Software Development Kit), joka mahdollistaa sovellusten kehittämisen sekä Androidille että iOS:lle yhdellä koodipohjalla. Flutter käyttää Dart-ohjelmointikieltä ja tarjoaa oman käyttöliittymäkirjaston, jonka avulla sovelluksista voidaan toteuttaa visuaalisesti yhtenäisiä ja suorituskykyisiä. (Flutter, 2024)

3.2 Xamarin

Microsoftin kehitysympäristö, joka mahdollistaa C#-kielen käytön Android-sovellusten kehityksessä. Xamarinissa voit jakaa suuren osan koodista myös iOS-sovelluksen kanssa, ja mikäli sovellus halutaan myöhemmin tehdä myös iOS:lle, voidaan tuki lisätä melko helposti Xamarinin avulla. Huonoina puolina voidaan kuitenkin mainita se, että Googlen valmiita komponentteja löytyy vähän, tai ei ne kunnolla sovi yhteen. Myös Xamarinin dokumentaatio ei ole niin kattavaa, kun joissakin muissa kehitysympäristöissä. (Xamarin, 2024)

3.3 React Native ja Ionic

React Native on JavaScript-pohjainen kehys, jonka avulla voidaan rakentaa natiivin suorituskyvyn sovelluksia Androidille ja iOS:lle. Se hyödyntää natiivikomponentteja käyttöliittymässä, mikä tekee sovelluksista nopeita ja sulavia. React Native mahdollistaa myös pääsyn mobiililaitteen API:hin JavaScriptin kautta, mikä nopeuttaa sovelluskehitystä ja helpottaa koodin ylläpitoa. (React Native, 2024)

Ionic on JavaScriptiin perustuva kehys, joka käyttää web-teknologioita kuten HTML:ää, CSS:ää ja JavaScriptiä sovellusten rakentamiseen. Ionicin avulla sama koodipohja voidaan julkaista useille alustoille, mutta suorituskyky voi olla hieman heikompi kuin täysin natiivilla sovelluksella. Ionic tarjoaa pääsyn laitteiston ja käyttöjärjestelmän ominaisuuksiin JavaScriptin kautta, mikä tekee kehityksestä joustavaa ja nopeaa. (Ionic Framework, 2024)

3.4 Android Studio

Android Studio on virallinen ohjelmointiympäristö (IDE) Android -sovellusten kehittämiseen. Se perustuu IntelliJ IDEA:n kehittämään editoriin ja työkaluihin, ja se sisältää myös muun muassa emulaattorin, jolla voidaan mallintaa puhelinta tai muuta Android-sovellusta käyttävää laitetta. (Android Studio, 2024)

Android studiolla ohjelmoidaan Java-kielellä ja enenevässä määrin myös Kotlinilla. Kotlin on suhteellisen uusi ohjelmointikieli, joka on hyvin samankaltainen kuin Java, mutta tarjoaa paremman tuen

Androidin ominaisuuksille. Lisäksi C++-kieltä voidaan käyttää Android-sovellusten kehityksessä, erityisesti suorituskykyyn tai matalan tason toimintoihin liittyvissä osissa. C++-koodia voidaan yhdistää Java- tai Kotlin-koodin kanssa JNI-rajapinnan (Java Native Interface) kautta.

Android Studiossa on myös merkittävä etu siinä, että se on Googlen kehittämä. Tämän ansiosta siihen on saatavilla valmiita kirjastoja lähes mihin tahansa ominaisuuteen, kuten esimerkiksi PDF-tallennukseen tai valokuvien ottamiseen.

4 SOVELLUS

4.1 Käyttötarkoitus

Ensimmäinen palaveri pidettiin Teamsin välityksellä ohjaavan opettajan ja toimeksiantajan kanssa. Palaverissa käytiin läpi työn lähtökohtia, tavoitteita sekä alustavaa aikataulua. Tämän jälkeen kävin toimeksiantajan luona suunnittelemassa sovellusta ja haluttuja ominaisuuksia tarkemmin.

Toimeksiantajan ehdottama pelkkä optinen tekstintunnistus (OCR) kuvasta olisi kuitenkin liiankin helppo toteuttaa tätä opinnäytetyötä silmällä pitäen. Käytännössä myös teollisuuden olosuhteissa voi olla joskus, että laitteen koodit eivät ole näkyvillä, ne ovat kuluneita tai likaisia.

Sovellukseen haluttiin myös vaihtoehtoisia tapoja hakea laitteen tietoja: käyttäjä voisi syöttää laitteen koodin manuaalisesti tai selata laitteita laiteryhmittäin. Tämä mahdollistaa laitteen tunnistamisen tilanteissa, joissa kuvaaminen ei onnistu, esimerkiksi kameravian tai huonon näkyvyyden vuoksi. Sovellukselta, ainakaan alussa, ei vaadittu kovin monia ominaisuuksia ja se olisi ulkoasultaankin hyvin pelkistetty.

Kun kuvasta tunnistetaan tietty laite, avataan siihen laitteeseen liittyvät valikot. Näitä olisi ainakin asennustarkastuslista (sovellus, minkä teimme ryhmätyönä ennen tätä projektia), laitteen ohjekirjat ja kalibrointiohjeet

4.2 Käyttöympäristö ja laitteet

Toimeksiantajayritys toimii tyypillisesti teollisuuslaitoksissa, kuten paperitehtaat ja voimalaitokset. Yritys huoltaa teollisuuslaitoksien prosessilaitteistoja.

Automatisoiduissa tehtaissa ja prosesseissa käytetään hyvin monenlaisia laitteita, mitä yritys vaihtaa ja huoltaa. Tällaisia laitteita ovat esimerkiksi kytkimet, erityyppiset anturit tai ohjainlaitteet.

Kuvissa 1 ja 2 esitetään esimerkkejä teollisuuslaitoksissa käytettävistä laitteista, mitä tässä opinnäytetyössä suunnitellulla sovelluksella on tarkoitus tunnistaa.

Kuva 1: HART-lämpötilalähetin

Kuva 2: Rajakytkin

4.3 Valitut ratkaisut

Opinnäytetyön sovelluksen tekemiseen valittiin lopulta Android Studio -ympäristö. Valinta perustui siihen, että natiivikehityksellä voidaan hyödyntää Android-laitteiden kaikkia ominaisuuksia, kuten kameraa, sensoreita ja tiedostojärjestelmää, ilman rajoituksia. Lisäksi natiivisovellus tarjoaa parhaan suorituskyvyn, vakauden ja käyttäjäkokemuksen, mikä on tärkeää sovelluksen kuvantunnistus- ja laitetunnistustoiminnoissa.

Testilaitteena käytin Lenovo TB-7306X-tablettia, jossa on 7 tuuman näyttö. Toinen testilaite oli Samsung Galaxy A10 -puhelin. Molemmissa laitteissa oli Android versio 11.

4.4 Käyttöliittymä ja toiminnallisuus

Sovelluksen käyttöliittymä (Kuva 3) pyrittiin suunnittelemaan mahdollisimman selkeäksi ja käyttäjäystävälliseksi. Etusivulla käyttäjälle esitetään kaksi päätoimintoa: ensimmäisestä painikkeesta avautuu kamera-aktiviteetti, jonka avulla voidaan tunnistaa laite kuvan perusteella. Toisella painikkeella käyttäjä voi valita laitteen manuaalisesti esiin avautuvasta listasta.

Käyttöliittymä toteutettiin perinteisellä Activity-pohjaisella rakenteella, ja näkymien määrittelyssä käytettiin XML-kuvauksia.

Kuvantunnistuksen jälkeen sovellus siirtyy laitteen tietonäkymään, jossa esitetään laitteen perustiedot. Lisäksi näkymässä on painikkeet, joiden kautta käyttäjä pääsee tarkastelemaan esimerkiksi laitteen dokumentaatiota, huoltohistoriaa tai muita siihen liittyviä tietoja.

Kuva 3: Ensimmäinen versio käyttöliittymästä

Kun käyttäjä valitsee päävalikosta vaihtoehdon **"Recognize device"**, sovellus avaa kameran. Kuvan ottamisen jälkeen näytölle avautuu ponnahdusikkuna, jossa esitetään otettu kuva sekä laitteen nimi, mikäli sovellus onnistui tunnistamaan laitteen. Mikäli tunnistuksen varmuus on heikko, voidaan valita uudelleenkuvaus tai palata päävalikkoon.

4.5 Kuvantunnistuksen testaaminen ja menetelmien selvitys

Aluksi aloin tutkia, millä tekniikoilla kuvantunnistuksen voisi toteuttaa. Etsin vaihtoehtoja sekä Googlen että ChatGPT:n avulla, jonka avulla pääsinkin alkuun. Tutkin myös paljon hieman samanlaisista aiheista tehtyjä opinnäytetöitä ja katsoin YouTubesta opetusvideoita, joissa näytettiin koodin toteutusta vaiheittain.

4.5.1 Ensimmäiset testit ja koodin tunnistus kuvasta

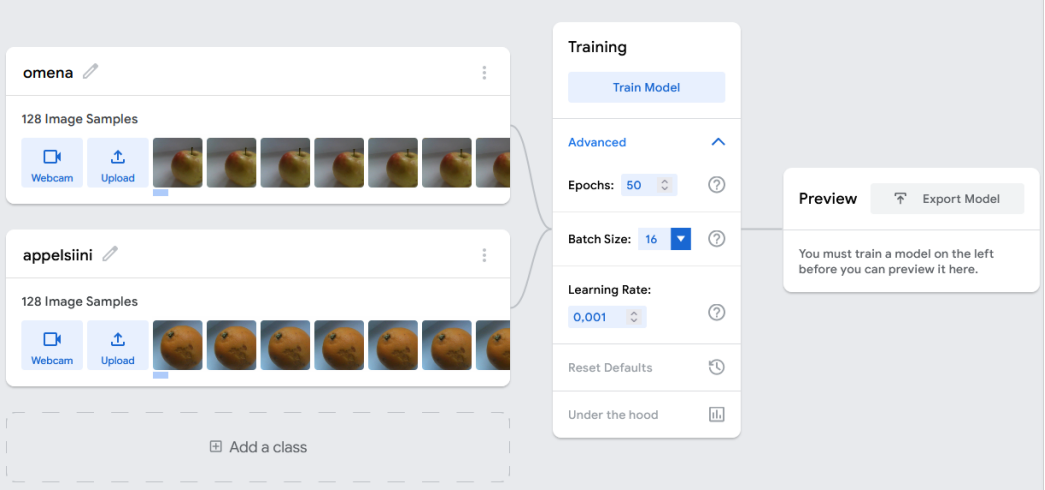
Aluksi tein testiprojektin, jossa kokeilin tunnistaa kuvasta tietyn koodin ja avata sen perusteella seuraavan näkymän. Tämä toimi yksinkertaisena tapana testata peruskonseptia ja varmistaa, että kuvantunnistus oli mahdollista toteuttaa juuri kyseisellä menetelmällä.

Tekstin tunnistus kuvasta onnistui melko helposti Googlen ML Kit -kirjaston Vision API:n avulla. Aluksi tunnistus toimi vain, kun kohdetta zoomattiin lähemmäksi, ja muutenkin se oli epävarmaa. Käyttämäni esimerkkikoodi oli vanhentunutta, sillä siinä käytettiin vanhaa Camera-API:a, eikä se ollut optimaalinen uudempien kamerasovellusratkaisujen kanssa.

Päivitin projektin käyttämään uudempaa Camera2-API:a, ja tämän jälkeen tekstintunnistus toimi huomattavasti paremmin ja luotettavammin. Tämä paransi merkittävästi sovelluksen luotettavuutta ja tunnistuksen tarkkuutta. Huomasin, että uudempien kamerasovellusratkaisujen käyttö, kuten Camera2, oli välttämätöntä, jotta sovelluksen toiminta olisi luotettavaa ja nopeaa.

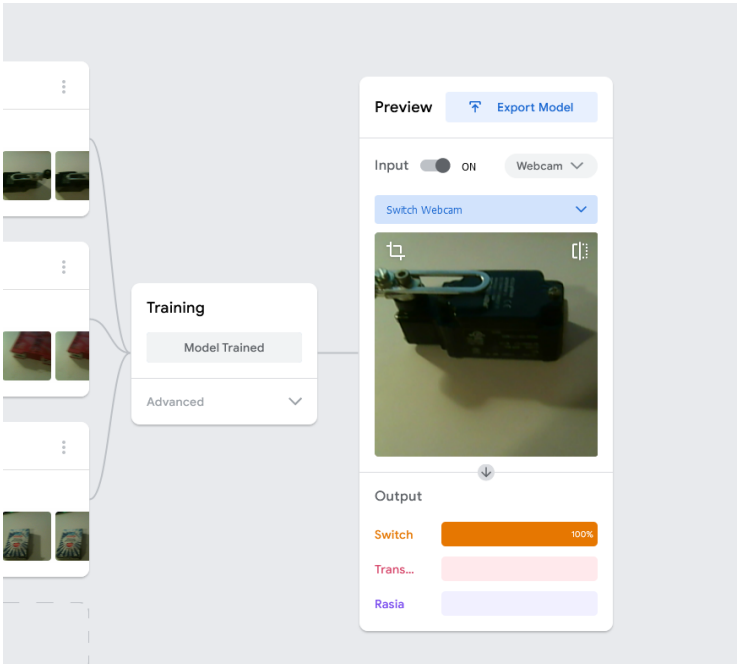
4.5.2 Teachable Machine

Seuraavaksi aloin perehtyä mallien kouluttamiseen ja siihen, kuinka valmis malli liitetään osaksi Android-projektia. Ratkaisuksi löytyi Googlen tarjoama Teachable Machine -palvelu, joka osoittautui helppokäyttöiseksi vaihtoehdoksi erityisesti alkuvaiheen kokeiluihin. Palvelussa voidaan esimerkiksi ottaa web-kameralla sarja kuvia halutusta kohteesta ja käyttää niitä suoraan mallin opetukseen (Kuva 4).



Kuva 4: Mallin koulutus Teachable Machinessa

Mallin luominen palvelussa on yksinkertaista: sovellukseen lisätään halutut kuvat, minkä jälkeen valitaan Train Model. Kun koulutus on valmis, malli voidaan viedä (export) ja ladata TensorFlow Lite -tiedostona omalle tietokoneelle jatkokäyttöä varten. Koulutetun mallin toimivuutta voidaan testata suoraan palvelussa, jossa näkyy, millä varmuudella malli tunnistaa sille syötetyn kuvan (Kuva 5).



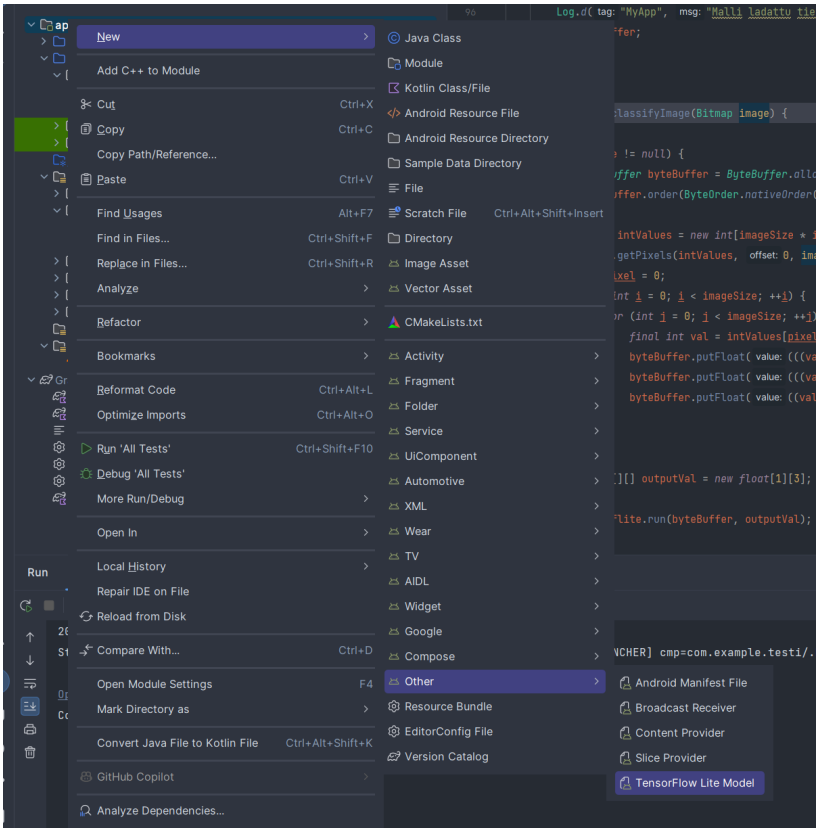
Kuva 5: Teachable Machinen näkymä, jossa testataan mallin tunnistustarkkuutta. Kuvassa näkyy mallin ennustama luokka ja sen todennäköisyysprosentti (“varmuus”) kullekin testatulle kuvalle.

4.5.3 Luodun mallin testaaminen

Mallin koulutuksen jälkeen seuraava vaihe oli testisovelluksen toteuttaminen ja koulutetun mallin liittäminen siihen (Kuva 6). Käytössä oleva malli on TensorFlow Lite -malli, joka soveltuu hyvin mobiilisovelluksiin kevyen rakenteensa ja nopean suorituskykynsä ansiosta. TensorFlow Lite -tiedosto (.tflite) kopioitiin Android-projektin res/raw -hakemistoon, joka sisältää sovelluksen staattisia resursseja. Hakemisto luodaan projektiin tarvittaessa.

Tämän jälkeen malli lisättiin projektiin Android Studion valikoiden kautta:

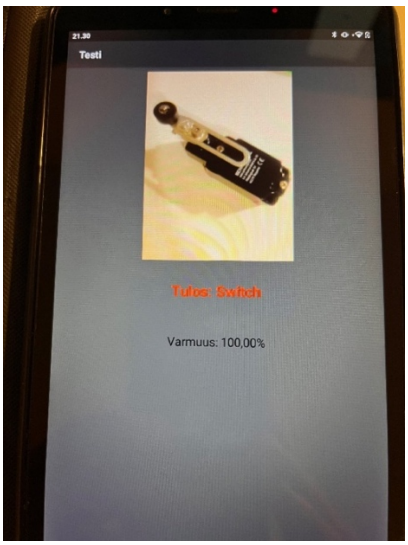
New → Other → TensorFlow Lite Model



Kuva 6: Mallin lisääminen projektiin

Kun malli oli liitetty ja tarvittava koodi kirjoitettu, siirryttiin testaamiseen. Testausta varten sovellus käynnistettiin emulaattorin sijaan fyysisellä Android-laitteella. Laitteen voi yhdistää Android Studioon joko USB-kaapelilla tai Wi-Fi-yhteyden avulla.

Fyysisen laitteen käyttöönotto onnistuu Android Studiossa myös QR-koodin avulla, kunhan tietokone ja mobiililaite ovat samassa Wi-Fi-verkossa. Testisovellus tunnisti testikuvista kaksi eri laitetta odotetusti (Kuva 7).



Kuva 7: Tunnistettu kuva mobiililaitteen näytöllä

Teachable Machinessa koulutettu malli tuntui toimi kokeilussa varsin hyvin, vaikka sen testaaminen erilaisissa ympäristöissä, valaistuksissa ja olosuhteissa jäi rajalliseksi. Käytännön sovelluksiin Teachable Machine ei kuitenkaan sovellu, sillä mallin ylläpito ja erityisesti uusien laitteiden lisääminen olisi työlästä. Palvelu toimii parhaiten prototypointiin ja alustavaan kokeiluun, mutta ei sellaisenaan tuotantokäyttöön.

4.5.4 Kuvien ottaminen

Mallin koulutukseen käytettävien kuvien määrä vaihtelee sen mukaan, kuinka tarkaksi tunnistus halutaan saada. Yleisesti noin 100–500 kuvaa per luokka riittää kohtuulliseen tarkkuuteen, mutta suurempaan luotettavuuteen pyrittäessä voidaan käyttää jopa tuhansia kuvia. Koulutuskuvien tulee olla mahdollisimman monipuolisia, eli ne olisi hyvä ottaa erilaisissa valaistuksissa, eri kulmista ja vaihtelevilta etäisyyksiltä, jotta malli oppii tunnistamaan laitteen todellisissa olosuhteissa. Suuren kuvamäärän kerääminen onnistuu käytännössä helpoiten kuvaamalla lyhyt video ja muuntamalla se kuviksi siihen tarkoitetulla työkalulla. Tässä projektissa käytin komentorivipohjaista ffmpeg-ohjelmaa, jolla videosta voidaan poimia haluttu määrä ruutuja koulutuskuviksi.

Ensin laitetta varten luotiin kansiot (esimerkiksi WL260, G9SB, F4SP), joihin kuvat tallennettiin. Yksittäiset videot voi muuntaa kuviksi erikseen, mutta useiden videoiden käsittelyä varten on kätevää tehdä erillinen .bat-tiedosto, jonka avulla kaikki videot voidaan prosessoida kerralla. ffmpeg-ohjelman komento noudattaa seuraavaa syntaksia:

C:\Users\Kone\Downloads>ffmpeg -i img_3363.mp4 -vf "fps=2" kuvat/C60N/frame_%03d.jpg

Tässä esimerkissä ohjelma poimii videosta yhden kuvan joka toisen ruudun välein (fps=2) ja tallentaa ne C60N-kansioon tiedostonimillä *C60N_001.jpg, C60N_002.jpg* jne. Näin saadaan nopeasti suuri määrä tasalaatuisia kuvia mallin koulutusta varten.

4.6 Mallin luominen

Malli koulutettiin käyttämällä Python-ohjelmointikieltä sekä Keras- ja TensorFlow-kirjastoja. Tässä työssä hyödynnettiin valmiiksi koulutettua MobileNetV2-konvoluutioneuroverkkoa (CNN) transfer learning -menetelmällä. MobileNetV2 on kevyt ja tehokas malli, joka on alun perin koulutettu suurella ja monipuolisella kuvadatasetilla, ja sen avulla malli on jo oppinut yleisiä visuaalisia piirteitä, kuten ääriviivoja, muotoja ja tekstuureja.

Transfer learning -lähestymistavassa valmiin verkon alkuosat pidettiin muuttumattomina, ja ainoastaan viimeisiä kerroksia muokattiin uutta luokittelutehtävää varten. Tämä mahdollisti sen, että malli saavutti hyvän tunnistustarkkuuden suhteellisen pienellä määrällä harjoituskuvia, eikä se mukautunut liiallisesti opetusdataan. Käytännön testeissä malli saavutti tunnistustarkkuuden noin 80–90 prosenttia, mikä teki siitä käyttökelpoisen todellisissa olosuhteissa.

Kuvassa 8 on havainnollistava pätkä Python-koodista, jossa valmiin MobileNetV2-mallin viimeisiä kerroksia muokataan ja koulutetaan uudelleen projektin kuvadatalle. Lopuksi koulutettu malli muunnettiin TensorFlow Lite -muotoon, jotta se voitiin integroida Android-sovellukseen. Mallin luokkien nimet tallennettiin erilliseen tekstitiedostoon, mikä mahdollisti luokkien helpon käytön sovelluksessa.

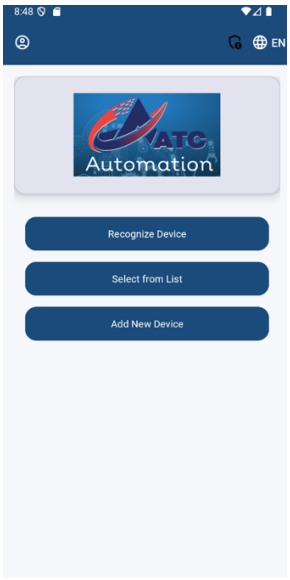
```python
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Ladataan valmiiksi opetettu perusmalli ilman yläkerroksia
base_model = MobileNetV2(input_shape=(*img_size, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Aluksi jäädytetään painot

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(train_data.num_classes, activation='softmax')
])
```

Kuva 8: MobileNetV2 koulutusta Pythonilla

5 LOPULLINEN SOVELLUS

5.1 Alkuruutu

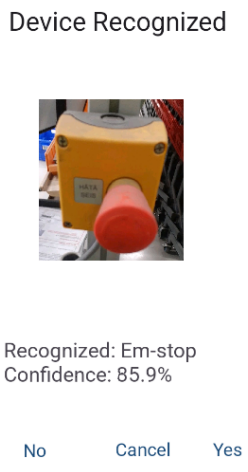
Sovelluksen käynnistyessä avautuu aloitussivu (Kuva 9), missä päästään hakemaan laitteita luettelosta, tai tunnistamaan laite kuvan perusteella, mikä tässä työssä olikin itse tarkoitus. Lisäksi sovelluksessa on mahdollista lisätä uusi laite järjestelmään, ottamalla kuvat ja antamalla tiedot.



Kuva 9: Sovelluksen päävalikko

5.2 Tunnistus kuvasta

Kuvasta tunnistuksella otetaan kuva laitteesta, minkä tiedot halutaan hakea. Kamera käynnistyy ja kuvauksen jälkeen näytetään ponnahdusikkuna, missä näytetään tunnistettu laite (Kuva 10), sekä millä tarkkuudella prosenteissa se on tunnistettu.



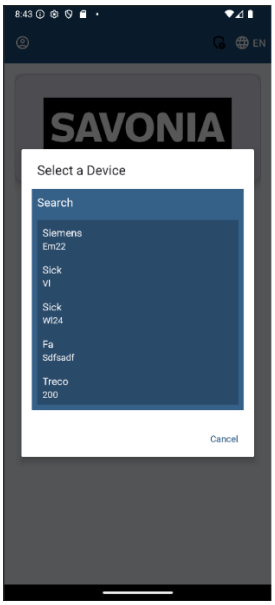
Kuva 10: Tunnistettu laite

Ponnahdusikkunassa voidaan valita myös uudelleenkuvaus, mikäli laite ei vaikuta oikealta. Valittaessa ”kyllä”, avautuu tunnistetun laitteen laitekortti (Kuva 11).

Kuva 11: Laitekortti

5.3 Laitteen valinta luettelosta

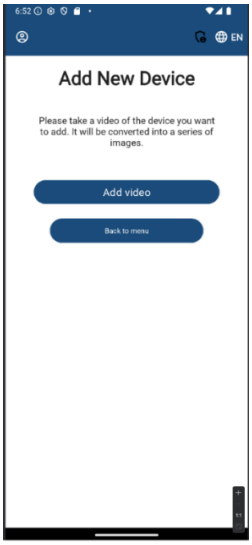
Laitteet voidaan kuvasta tunnistuksen sijaan valita myös luettelosta (Kuva 12). Hakukenttään voi alkaa kirjoittaa haetun laitteen valmistajaa tai nimeä, niin se sitä mukaan löytää vastaavia laitteita. Tämä on kätevää siinä vaiheessa, jos laitteita olisi kymmeniä tai jopa satoja erilaisia.



Kuva 12: Laitteen valinta luettelosta

5.4 Uuden laitteen lisäys

Laitteen lisääminen (Kuva 13) aloitetaan videon kuvauksella. Videon kuvat muutetaan kuviksi ja tallennetaan pilveen, uuteen kansioon. Videon pituudeksi on valittuna korkeintaan 20 sekuntia, eli tämän pituisella videolla tulisi noin 40 kuvaa.



Kuva 13: Lisää uusi laite

Seuraavaksi voidaan valita oletuskuva, mikä tulee laiteen valikon kuvaksi. Tämän jälkeen syötetään laiteen tiedot: merkki, malli, tyyppi ja annetaan linkit pdf-muotoisiin ohjeisiin.

Kun kaikki tarvittavat kentät on täytetty, tallennetaan tiedot tietokantaan. Tämän jälkeen alkaa prosessi, missä palvelimella pyörivällä koodilla tehdään kaikista kuvakansioista uusi kuvantunnistusmalli.

6 KÄYTETTYJÄ TEKNOLOGIOITA

6.1 Google Drive

Google Drive on pilvipalvelu, jonka tarjoaa Google. Se mahdollistaa käyttäjille tiedostojen tallentamisen, jakamisen ja hallinnoinnin verkossa. Google Driveen voi tallentaa erilaisia tiedostoja, kuten kuvia, videoita, dokumentteja ja muita tiedostoja. Tiedostot sijaitsevat verkkopohjaisessa pilvessä eli palvelimilla, joihin käyttäjät voivat kirjautua sisään mistä tahansa Internetin kautta. (Google Drive, 2024)

Google Drive tarjoaa myös mahdollisuuden tiedostojen jakamiseen muiden käyttäjien kanssa ja yhteistyöhön reaaliajassa. Käyttäjät voivat jakaa tiedostoja linkin avulla tai kutsua muita käyttäjiä osallistumaan yhteisölliseen tiedostojen muokkaamiseen. Tämä tekee yhteistyön helpoksi ja mahdollistaa monien ihmisten työskentelyn saman tiedoston parissa samanaikaisesti. (Google Drive, 2024)

6.2 ML Kit

ML Kit on Googlen kehittämä mobiilikehityksen työkalupaketti, joka sisältää valmiita ratkaisuja koneoppimisen integroimiseksi Android- ja iOS -sovelluksiin. Se tarjoaa helppokäyttöisiä rajapintoja ja toiminnallisuuksia, jotka mahdollistavat esimerkiksi kuvantunnistuksen, tekstin analysoinnin ja muiden koneoppimistehtävien toteuttamisen sovelluksissa ilman syvällistä koneoppimisosaamista. (Google ML Kit, 2024)

6.3 TensorFlow

TensorFlow on avoimen lähdekoodin koneoppimis- ja syväoppimisalusta, joka tarjoaa kattavan valikoiman työkaluja ja kirjastoja koneoppimisen ja syväoppimisen mallien rakentamiseen, kouluttamiseen ja käyttöön erilaisissa sovelluksissa. TensorFlow hyödyntää tietojen käsittelyssä niin kutsuttua graafipohjaista mallia. Tiedot kulkevat mallin läpi tietovirtoina, ja eri laskennalliset toiminnot on esitetty solmuina, jotka on yhdistetty reunoilla. Tämä rakenne mahdollistaa tietojen tehokkaan käsittelyn ja mallien skaalautuvuuden, mikä tekee TensorFlow’sta soveltuvan monenlaisiin koneoppimistehtäviin, kuten kuvien, puheen ja tekstin tunnistamiseen. TensorFlow Lite on TensorFlown kevytversio, joka on suunniteltu erityisesti mobiili- ja sulautetuille laitteille, jolloin koneoppimismalleja voidaan ajaa suoraan laitteella nopeasti ja energiatehokkaasti (TensorFlow, 2024)

7 YHTEENVETO JA POHDINTA

Sovellus toimisi kyllä sillä tavalla, kun alun perin haluttiin, eli laite voidaan teollisuusympäristössä hakea luettelosta kätevästi vain kuva ottamalla. Käytännössä toteutus olisi kuitenkin hyvin hidasta ja hankalaa, jos jokainen laite pitäisi erikseen kuvata ja malli aina uudelleen kouluttaa, koska laitteita tulisi olemaan kymmeniä, jopa satoja kappaleita. Alussa ensimmäistä mallia tehtäessä, jossa on paljon laitteita, kuvat olisi viisainta ottaa manuaalisesti ja tallentaa kansioihin, minkä jälkeen malli voidaan kouluttaa. Yksittäisten laitteiden lisääminen tällä tavalla onnistuu kuitenkin suhteellisen helposti.

Käytännössä tällainen sovellus voisi olla helpompi toteuttaa myös siten, että joka laitteessa olisi QR-koodi, jonka lukemalla päästäisiin nopeasti laitteen tietoihin käsiksi. Tämä ratkaisu vaatisi kuitenkin alkuvaiheessa paljon työtä esimerkiksi tarrojen suunnittelun ja kiinnittämisen muodossa.

Vaikka projekti eteni välillä hitaasti, se oli silti erittäin opettavainen. Työn aikana opin paljon siitä, miten kuvantunnistus toimii käytännössä ja mitä kaikkea sen toteuttaminen vaatii. Samalla ymmärrys Android-ohjelmoinnista kasvoi. Projekti oli melko pitkä, ja kehityksen aikana jotkin käytetyt teknologiat ehtivät jo vanhentua tai niiden tilalle ilmestyi uusia, tehokkaampia ratkaisuja. Tämä ilmiö on tyypillistä Android-kehityksessä, sillä alustojen, kirjastojen ja kehysten päivitystahti on nopeaa.

Erilaisista apuvälineistä huolimatta ohjelmointi oli ajoittain hidasta, ja alkuun pääseminen saattoi olla vaikeaa. Lisäksi sovellusta testattaessa se saattoi kaatua monista eri syistä, ja vianetsintään kului usein paljon aikaa.

Mikäli sovellus otettaisiin myöhemmin tuotantokäyttöön yrityksessä, siihen olisi tarpeen lisätä useita uusia ominaisuuksia. Sovellukseen voitaisiin esimerkiksi integroida aiemmin kouluprojektina toteutettu tarkistuslistasovellus, jota käytetään sähköisten laitteiden mittauspöytäkirjoina tai huoltotöiden yhteydessä. Lisäksi sovellukseen tulisi toteuttaa käyttäjien kirjautuminen, jossa esihenkilöille ja asentajille olisi omat tunnuksensa sekä rajatut oikeudet laitteiden lisäämiseen.

Jatkokehityksessä tulisi huomioida myös ulkoasuun ja käytettävyyteen liittyvät tekijät, kuten sovelluksen skaalautuminen eri kokoisille näytöille ja näytön kääntämisen tuki. Sovelluksen testaaminen todellisissa käyttötilanteissa toisi todennäköisesti esiin uusia kehityskohteita sekä ideoita jatkokehitysominaisuuksille.

Kokonaisuutena projekti antoi hyvän kuvan siitä, miten kuvat, koneoppimismalli ja sovelluksen käyttöliittymä liittyvät toisiinsa. Työn aikana saatu oppimiskokemus antaa hyvän pohjan vastaaville projekteille tulevaisuudessa sekä mahdollisuuden jatkaa sovelluksen kehittämistä eteenpäin, mikäli tarve sitä edellyttää.

LÄHTEET

Tekoälyä on käytetty tekstin muokkaamiseen, lyhyiden kappaleiden yhdistämiseen ja kielelliseen viimeistelyyn: ChatGPT (2025) GPT-5.1. OpenAI. Käyttöajankohta: 2025. Saatavilla: https://chat.openai.com (haettu 12.12.2025).

Amazon Web Services. Haettu 31.1.2024 osoitteesta https://aws.amazon.com/devops/source-control/

Flutter. Haettu 15.5.2024 osoitteesta https://docs.flutter.dev/resources/faq

Google Drive. Haettu 31.1.2024 osoitteesta https://drive.google.com/

Google ML Kit. Haettu 21.2.2024 osoitteesta https://developers.google.com/ml-kit

IBM (2025a). Haettu 18.12.2025 osoitteesta https://www.ibm.com/topics/deep-learning

IBM (2025b). Haettu 18.12.2025 osoitteesta https://www.ibm.com/think/topics/convolutional-neural-networks

Ionic Framework. Haettu 14.4.2024 osoitteesta https://ionicframework.com/

Xamarin. Haettu 10.4.2024 osoitteesta https://dotnet.microsoft.com/en-us/apps/xamarin

Ojanperä, Tero 2023. Tekoälyn vallankumous: käsikirja. E-kirja. Helsinki: Alma Talent. Viitattu 13.4.2024.

Pietikäinen, M. ja Silvén, O.: Tekoälyn haasteet: Koneoppimisesta ja konenäöstä tunnetekoälyyn (päivitetty toinen painos). Oulun yliopisto 2021. Viitattu 14.4.2024. Saatavilla: https://oulu-repo.oulu.fi/bitstream/handle/10024/36376/isbn978-952-62-3202-7.pdf?sequence=1&isAllowed=y

React Native. Haettu 14.4.2024 osoitteesta https://reactnative.dev/

Russell, Stuart J. & Norvig, Peter 2022. Artificial Intelligence: A Modern Approach, 4. painos. E-kirja. Harlow: Pearson Education Limited. Viitattu 16.12.2025.

TensorFlow. Haettu 14.4.2024 osoitteesta https://www.tensorflow.org/

YouTube. Haettu 21.2.2024 osoitteesta https://www.youtube.com/howyoutubeworks/