Vesa Kotanen, Olli Rinta-Filppula ja Eeva-Leena Wallin-Niemelä

Tietotekniikan projektityö

OCR - Materiaalin tunnistus kuvasta

Projektityön raportti

Harjoitustyö   
Kevät 2025   
Tite22

**SISÄLTÖ**

[1 SOVELLUS 3](#_Toc190074697)

[2 DOKUMENTOINTI 4](#_Toc190074698)

[2.1 Sovelluksen kirjastot 4](#_Toc190074699)

[2.1.1 TensorFlow Hub 4](#_Toc190074700)

[2.1.2 TensorFlow 4](#_Toc190074701)

[2.1.3 Matplotlib.pyplot ja Matplotlib.image 5](#_Toc190074702)

[2.1.4 Pandas 5](#_Toc190074703)

[2.2 Sovelluksen vuokaavio 6](#_Toc190074704)

[3 TESTAUSSUUNNITELMA 7](#_Toc190074705)

[3.1 Testin tarkoitus 7](#_Toc190074706)

[3.2 Testin laajuus 7](#_Toc190074707)

[3.3 Testitulokset 7](#_Toc190074708)

[3.4 Tulosten analysointi 9](#_Toc190074709)

[3.5 Yhteenveto testituloksista 13](#_Toc190074710)

[4 RISKIENHALLINTA 14](#_Toc190074711)

[4.1 Riskit 14](#_Toc190074712)

[4.2 Riskienhallintastrategia 15](#_Toc190074713)

[4.3 Riskien seuranta ja arviointi 16](#_Toc190074714)

[5 YLLÄPITO JA LAAJENNETTAVUUS 17](#_Toc190074715)

[5.1 Järjestelmän ylläpitovaatimukset 17](#_Toc190074716)

[5.2 Laajennettavuuden suunnittelu 17](#_Toc190074717)

# SOVELLUS

Tämä sovellus liittyy materiaalien tunnistamiseen ja sen tarkoitus on analysoida ja tunnistaa erilaisia fyysisiä materiaaleja niiden ominaisuuksien, kuten tekstuurin, värin, muodon tai muun näkyvän tiedon perusteella. Sovellus hyödyntää kuvia tai skannattuja tiedostoja, joissa materiaalit esiintyvät, ja käyttää näitä visuaalisia tietoja niiden luokittelemiseen tai tunnistamiseen. Tavoitteena voi olla esimerkiksi selvittää materiaalin tyyppi, soveltuvuus tiettyihin tarkoituksiin tai tunnistaa kierrätettäviä materiaaleja.

Käytännössä sovellusta voidaan hyödyntää monilla eri aloilla. Esimerkiksi rakennus- ja tuotantoteollisuudessa sovellus voi auttaa varmistamaan materiaalien laadun tai tunnistamaan erilaisia komponentteja. Kierrätyksessä ja jätehuollossa sovellus voi tunnistaa, mitkä materiaalit ovat kierrätettäviä ja mihin jakeeseen ne kuuluvat. Kuluttajakäytössä se voi olla apuna vaikkapa rakennus- tai korjausprojekteissa, joissa materiaalien tunnistaminen on tärkeää oikean lopputuloksen saavuttamiseksi.

Teknologian ytimessä on tekoäly ja koneoppiminen, joiden avulla sovellus voi oppia jatkuvasti lisää erilaisista materiaaleista ja parantaa tunnistustarkkuuttaan. Sovelluksen käyttö voi tehostaa työnkulkua ja vähentää manuaalisen tunnistamisen tarvetta, tarjoten nopeutta, tarkkuutta ja helppokäyttöisyyttä käyttäjilleen.

# DOKUMENTOINTI

Tämä dokumentti kuvaa ohjelmiston rakennetta ja miten eri osat toimivat yhdessä. Se sisältää arkkitehtuurikuvaukset ja muita suunnittelutietoja.

## Sovelluksen kirjastot

Tässä koneoppimispohjaisessa kuvantunnistusprojektissa käytetään useita Python-kirjastoja, jotka tarjoavat valmiita toimintoja analyysiprosessin eri vaiheisiin. Näiden kirjastojen avulla voidaan toteuttaa muun muassa koneoppimismallien lataus, kuvan esikäsittely ja tulosten visualisointi. Seuraavassa kuvataan koodissa käytettyjä kirjastoja ja niiden rooleja projektin toteutuksessa.

### TensorFlow Hub

TensorFlow Hub on laajasti käytetty kirjasto, joka mahdollistaa valmiiden koneoppimismallien helpon käytön. Se sisältää esikoulutettuja malleja, joita voidaan hyödyntää esimerkiksi kuvantunnistukseen, tekstin analysointiin tai signaalinkäsittelyyn.

**Käyttötarkoitus tässä projektissa:**

TensorFlow Hubia käytetään erityisesti valmiiden kuvatunnistusmallien lataamiseen. Esimerkiksi **hub.load()**-funktiolla voidaan ladata Open Images -malli, joka on koulutettu laajaan kuvadatasettiin ja pystyy tunnistamaan monenlaisia esineitä ja materiaaleja.

Tämä säästää aikaa ja resursseja, koska kehittäjien ei tarvitse kouluttaa mallia alusta asti.

### TensorFlow

TensorFlow on yksi suosituimmista koneoppimiskirjastoista, jota käytetään laajasti erityisesti syväoppimisen (deep learning) sovelluksissa. Se mahdollistaa tehokkaan tensorilaskennan ja sisältää runsaasti työkaluja neuroverkkojen rakentamiseen ja käyttämiseen.

**Käyttötarkoitus tässä projektissa:**

TensorFlow:n avulla kuvat muunnetaan tensorimuotoon, jotta ne ovat yhteensopivia koneoppimismallien kanssa. Esimerkiksi **tf.convert\_to\_tensor()**-funktiolla voidaan muuntaa ladattu kuva tensoriksi.

TensorFlow suorittaa mallin laskennalliset operaatiot ja palauttaa analyysin tulokset.

### Matplotlib.pyplot ja Matplotlib.image

Matplotlib on suosittu kirjasto datan visualisointiin, ja sen **pyplot**-moduuli tarjoaa helpon tavan luoda kuvaajia ja näyttää kuvia. Samalla **matplotlib.image**-moduuli mahdollistaa kuvien lataamisen ja niiden esittämisen osana analyysiprosessia.

**Käyttötarkoitus tässä projektissa:**

**matplotlib.image** käytetään kuvan lataamiseen tiedostosta. Esimerkiksi **mpimg.imread()**-funktio voi lukea kuvan eri formaateissa, kuten PNG tai JPEG.

**pyplot** puolestaan näyttää analysoidut kuvat. Esimerkiksi **plt.imshow()**-funktiolla voidaan esittää kuva ja analyysitulokset käyttäjälle.

### Pandas

Pandas on tehokas kirjasto datan käsittelyyn ja analysointiin. Sitä käytetään erityisesti suurten datasetien hallintaan ja niiden käsittelemiseen DataFrame-rakenteessa, joka muistuttaa taulukkolaskentaohjelmien, kuten Excelin, rakennetta.

**Käyttötarkoitus tässä projektissa:**

Pandas-kirjastolla ladataan ja käsitellään CSV-tiedosto, joka sisältää analyysin kannalta oleellisia luokkakuvauksia. Esimerkiksi tiedostossa voi olla määritettynä, mitä luokkia (kuten "muovi", "metalli" tai "lasi") mallin tulee tunnistaa.

DataFrame-rakenne mahdollistaa helpon pääsyn ja muokkauksen näihin tietoihin.

## Sovelluksen vuokaavio

Kuva, joka sisältää kohteen teksti, diagrammi, luonnos, piirros

Kuvaus luotu automaattisesti

Kuva 1. Sovelluksen vuokaavio

# TESTAUSSUUNNITELMA

## Testin tarkoitus

Testin tavoitteena on arvioida ohjelman toimintaa erilaisten kuvien tunnistuksessa. Kuvat annetaan ohjelman tunnistettavaksi yksitellen, ja saatuja tuloksia verrataan silmällä tehtyihin havaintoihin. Tavoitteena on selvittää ohjelman tarkkuus eri materiaalien tunnistamisessa ja arvioida sen toimivuutta erilaisten taustojen ja olosuhteiden yhteydessä.

## Testin laajuus

Testissä käytetyt kuvat kuuluvat neljään eri materiaalikategoriaan: lasi, metalli, muovi ja puu. Kuvasettejä on neljä kappaletta, joista jokaisessa on yksi kuva jokaisesta materiaalikategoriasta (yhteensä 16 kuvaa):

Setti 1 ja 2: Sisältävät selkeitä ja yksiselitteisiä kuvia eri materiaaleista.

Setti 3: Kuvat sisältävät päämateriaalin lisäksi muita objekteja, jotka voivat vaikuttaa tunnistukseen.

Setti 4: Materiaali esiintyy muodottomana ja monimutkaista taustaa vasten, testaten ohjelman kykyä tunnistaa materiaali haastavammissa olosuhteissa.

## Testitulokset

Alla olevassa taulukossa esitetään testin tulokset, mukaan lukien kuvan polku, tunnistuksen onnistuminen sekä ohjelman antama tunnistustulos:

**Kuva, joka sisältää kohteen teksti, numero, kuvakaappaus, ruokalista

Kuvaus luotu automaattisesti**

**Taulukko 1.** Testitulokset

## Tulosten analysointi

Testitulokset osoittavat, että ohjelma toimii luotettavasti selkeissä ja yksiselitteisissä kuvissa. Kahden ensimmäisen testisarjan kuvat tunnistettiin oikein kaikissa tapauksissa. Kolmannessa testisarjassa, jossa kuvat olivat monimutkaisempia, ohjelma tunnisti kohteet osittain oikein, mutta teki virheitä, kun kuvassa esiintyi useita objekteja. Tällöin ohjelma antoi useita mahdollisia tuloksia.

Neljännessä testisarjassa ohjelma ei pystynyt tunnistamaan materiaaleja luotettavasti. Virhetunnistukset johtuivat siitä, että ohjelma keskittyi taustalla näkyviin elementteihin eikä itse materiaaliin. Esimerkiksi kuvassa "kuppi\_kasi.jpeg" ohjelma ei tunnistanut materiaaliksi lasia, mutta havaitsi viinilasin, drinkin ja ihmiskäden. Kuva "ikkuna.jpeg" puolestaan tunnistettiin lasiksi, vaikka taustalla näkyi puita.

Alla olevassa kuvassa (Kuva1) on käytetty kuvaa ”kuppi\_kasi.jpeg” josta ohjelma ei tunnistanut materiaaliksi lasia. Ohjelma on kuitenkin tunnistanut juomalasin ensisijaisesti viinilasiksi (Wine glass). Ohjelma on tunnistanut myös Drink, Coctail ja Human hand.

A hand holding a glass of water

AI-generated content may be incorrect.

Kuva 2

Kuvan ikkunasta ”ikkuna.jpeg” ohjelma tunnisti oikein lasiksi ikkunasta näkyvistä puista huolimatta (Kuva2). Mikäli ikkunaa, ja lasia ei olisi tunnistettu, olisi materiaaliksi valikoitunut puu, joka näkyy myös material mapping taulukossa (Kuva3).

A screenshot of a window

AI-generated content may be incorrect.

Kuva 3 Kuvatunnistus

A graph with numbers and text

AI-generated content may be incorrect.

Kuva 4 Mapping

Viimeisen setin kuvat materiaaleista osoittautuivat ohjelmalle hankalaksi tunnistaa. image\_0001\_w.jpg on puuta epämäärisessä muodossa (Kuva 4).

A screenshot of a computer

AI-generated content may be incorrect.

Kuva 5

## Yhteenveto testituloksista

Testitulosten perusteella kuvatunnistusohjelma suoriutuu hyvin selkeistä kuvista, joissa materiaali on yksiselitteisesti erotettavissa. Kahdessa ensimmäisessä testisarjassa ohjelma tunnisti oikein kaikki annetut materiaalit, mikä osoittaa sen toimivan luotettavasti yksinkertaisissa olosuhteissa.

Kolmannessa testisarjassa, jossa kuviin sisältyi häiriötekijöitä, kuten muita materiaaleja tai objekteja, ohjelman tunnistustarkkuus heikkeni. Se pystyi tunnistamaan osan materiaaleista oikein, mutta virhetunnistuksia ilmeni erityisesti silloin, kun kuvassa esiintyi samanaikaisesti useita materiaaleja. Näissä tapauksissa ohjelma antoi todennäköisyydet eri materiaaleille, mutta ensisijainen tunnistus ei aina vastannut odotettua lopputulosta.

Neljäs testisarja osoitti ohjelman suurimmat haasteet: kun materiaali ei ollut selkeästi erottuva ja kuvan tausta oli monimutkainen, ohjelma epäonnistui tunnistuksessa. Virhetunnistukset johtuivat usein siitä, että ohjelma tulkitsi taustan elementit ensisijaiseksi kohteeksi, jolloin tunnistus perustui väärään materiaaliin.

Mahdollisia kehityssuuntia ohjelman parantamiseksi:

**Material Mapping -taulukon laajentaminen**: Parantamalla ohjelman tuntemia materiaaleja ja niiden kuvantunnistusmalleja voidaan lisätä sen tarkkuutta erityisesti haastavammissa kuvissa.

**Objektien erottelun kehittäminen**: Kuvien esikäsittelyvaiheessa voidaan parantaa ohjelman kykyä suodattaa ylimääräiset objektit pois ja keskittyä olennaiseen materiaaliin.

**Monimutkaisten kuvien analysointi:** Algoritmia voidaan kehittää siten, että se tunnistaa materiaalit paremmin myös silloin, kun niiden muoto on rikkonainen tai epämääräinen.

**Kontekstitietoisuuden parantaminen:** Kehittämällä ohjelmalle kykyä ymmärtää materiaalien tyypillisiä käyttöympäristöjä (esimerkiksi juomalasit tunnistettaisiin useammin lasimateriaaliksi, vaikka ne esiintyisivät kädessä pidettynä) voitaisiin vähentää virhetunnistuksia.

Näillä parannuksilla ohjelman luotettavuus ja käyttökelpoisuus voidaan merkittävästi kohentaa, erityisesti silloin, kun kuvien taustat ja materiaalien muodot eivät ole yksiselitteisiä.

# RISKIENHALLINTA

## Riskit

Materiaalien tunnistukseen tarkoitetulla sovelluksella, kuten muillakin tekoäly- ja OCR-t**ek**nologiaan perustuvilla sovelluksilla, on joitakin tunnettuja riskejä ja haasteita.

* Tunnistuksen epätarkkuus: Sovellus ei välttämättä tunnista materiaaleja oikein, jos kuvassa on heikko valaistus, epäselvä resoluutio tai materiaalit ovat kuluneita, likaantuneita tai päällekkäin. Tämä voi johtaa virheellisiin päätelmiin ai vääriin toimintoihin, kuten kierrätyksessä materiaalin ohjaamiseen väärään käsittelyyn.
* Rajoitettu aineistopohja: Jos sovelluksen koneoppimismalli on koulutettu rajallisella materiaalidatalla, se ei välttämättä tunnista harvinaisempia materiaaleja tai uusia muotoja. Tämä voi estää sen tehokkaan käytön tietyissä tilanteissa.
* Tietoturva- ja yksityisyysriskit: Jos sovellus käsittelee kuvia, joissa on arkaluonteista tietoa (esim. asiakirjoja, joissa materiaalit esiintyvät osana taustaa), datan väärinkäytön tai tietovuotojen riski voi kasvaa.
* Ympäristötekijöiden vaikutus: Sovellus voi olla altis ulkoisille häiriöille, kuten liialle valolle, varjoille tai muille ympäristöolosuhteille, jotka vääristävät tunnistusta.
* Käyttäjän virheet: Käyttäjä voi syöttää sovellukselle vääriä tai epäselviä kuvia, mikä johtaa virheellisiin tuloksiin. Tämä riski korostuu, jos käyttäjä ei ole perehtynyt sovelluksen käyttöön.
* Väärät johtopäätökset ja päätökset: Sovelluksen tekemät virheelliset tunnistukset voivat johtaa vääriin päätöksiin tuotannon, rakentamisen tai kierrätyksen kaltaisilla aloilla, aiheuttaen taloudellisia tai ympäristövaikutuksia.

Näiden riskien minimoimiseksi tarvitaan laadukasta koulutusaineistoa, käyttöympäristön optimointia, selkeitä käyttöohjeita ja tietoturvallista suunnittelua. Lisäksi sovelluksen käytön yhteydessä on hyvä varmistaa, että kriittiset päätökset tarkastetaan tarvittaessa manuaalisesti.

## Riskienhallintastrategia

* Tunnistuksen epätarkkuus
  + Parannetaan kuvankäsittelyä sovelluksessa (esim. kuvan tarkkuuden parantaminen ja valaistuksen optimointi).
  + Laajennetaan koneoppimismallin koulutusaineistoa kattamaan erilaisia valaistusolosuhteita, kulumia ja likaisia materiaaleja.
  + Otetaan käyttöön varmistusmekanismi, jossa epävarmat tunnistukset liputetaan tarkistettavaksi.
* Rajoitettu aineistopohja
  + Jatkuva koneoppimismallin päivitys uusilla materiaalidatoilla.
  + Yhteistyö käyttäjien kanssa keräämällä heidän tuottamaansa palautetta ja esimerkkidataa harvinaisista materiaaleista.
  + Integrointi laajempiin avoimiin materiaalidatoihin, jos mahdollista.
* Tietoturva- ja yksityisyysriskit
  + Varmistetaan kuvien anonymisointi ja salaaminen, jotta arkaluonteinen tieto ei pääse vuotamaan.
  + Käytetään turvallista tiedon siirtoa (esim. HTTPS, TLS) ja datan tallennusta (salauslevyt tai tietokannat).
  + Koulutetaan käyttäjiä tietoturvan perusperiaatteista ja sovelluksen turvallisesta käytöstä.
* Ympäristötekijöiden vaikutus
  + Suositellaan käyttäjille sovelluksen käyttöä standardoiduissa olosuhteissa, kuten riittävässä valaistuksessa ja tasaisella pinnalla.
  + Kehitetään algoritmeja, jotka korjaavat ympäristöstä johtuvia virheitä (esim. varjojen poisto ja automaattinen kontrastin säätö).
* Käyttäjän virheet
  + Luodaan palautemekanismi, jonka avulla käyttäjät voivat raportoida virheellisistä tunnistuksista.
  + Tarjotaan koulutusmateriaaleja sovelluksen oikeaoppiseen käyttöön.
* Väärät johtopäätökset ja päätökset
  + Implementoidaan kaksivaiheinen tunnistusjärjestelmä, jossa epävarmat tulokset tarkistetaan manuaalisesti ennen lopullisia päätöksiä.
  + Lisätään varoitusjärjestelmä, joka ilmoittaa käyttäjälle, jos tunnistuksen tarkkuus ei täytä tiettyä kynnysarvoa.
  + Dokumentoidaan sovelluksen rajoitukset ja suositellaan sen käyttöä vain tietyissä tehtävissä tai olosuhteissa.

## Riskien seuranta ja arviointi

* Seurantatyökalut: Käytetään analytiikkaa ja raportointityökaluja sovelluksen suorituskyvyn, käyttäjäpalautteen ja virhetilanteiden seuraamiseen.
* Auditoinnit: Suoritetaan säännöllisesti sisäisiä ja ulkoisia auditointeja tietoturvan ja tarkkuuden varmistamiseksi.
* Päivitykset: Julkaistaan säännöllisesti sovelluspäivityksiä, jotka parantavat tunnistuksen tarkkuutta ja vähentävät havaittuja riskejä.

Strategian avulla voidaan vähentää tunnettuja riskejä ja varmistaa, että sovelluksen käyttö on turvallista, tarkkaa ja tehokasta kaikissa tarkoituksissaan.

# YLLÄPITO JA LAAJENNETTAVUUS

## Järjestelmän ylläpitovaatimukset

Järjestelmän ylläpitoon liittyvät vaatimukset varmistavat, että sovellus toimii tehokkaasti pitkällä aikavälillä, on helppo päivittää ja vastaa käyttäjien muuttuviin tarpeisiin. Sovelluksen lähdekoodi, käyttöohjeet ja tekniset spesifikaatiot tulee dokumentoida selkeästi, jotta kehittäjät ja ylläpitäjät voivat helposti ymmärtää järjestelmän rakenteen ja toiminnallisuudet. Lisäksi on tärkeää kirjata päivitysten ja muutosten historia, jotta kehitystyötä voidaan seurata ja muutosten vaikutuksia arvioida. Versionhallintajärjestelmä, kuten Git, on olennainen työkalu muutosten seuraamiseen ja sovelluksen eri versioiden hallintaan.

Järjestelmän tiedot ja data tulee varmuuskopioida säännöllisesti tietojen säilyvyyden takaamiseksi. Varmuuskopiointi tulisi automatisoida, jotta se tapahtuu luotettavasti ilman manuaalista väliintuloa. Lisäksi on tärkeää testata säännöllisesti palautustoiminnallisuus, jotta voidaan varmistaa, että tiedot voidaan tarvittaessa palauttaa ongelmatilanteissa.

Vianhallinnan osalta järjestelmän toiminnan seuraamiseksi voi ottaa käyttöön virheiden ja häiriöiden raportointityökalu. Työkalun avulla voidaan seurata järjestelmän tilaa reaaliaikaisesti ja kerätä tietoa mahdollisista ongelmista. Lisäksi sovelluksen logitiedostot olisi hyvä tallentaa järjestelmällisesti ja analysoida säännöllisesti, jotta piilevät ongelmat voidaan havaita ja ratkaista nopeasti.

Resurssienhallinnan osalta järjestelmän suorituskykyä tulee seurata jatkuvasti, jotta voidaan estää resurssien ylikuormitus. Mahdolliset kapasiteettiongelmat ratkaistaan skaalaamalla järjestelmää tarpeen mukaan, esimerkiksi lisäämällä palvelinresursseja tai optimoimalla järjestelmän suorituskykyä. Näiden ylläpitovaatimusten avulla varmistetaan, että sovellus toimii luotettavasti ja käyttäjäkokemus säilyy korkealaatuisena.

## Laajennettavuuden suunnittelu

Laajennettavuus varmistaa, että järjestelmä pystyy mukautumaan uusiin vaatimuksiin ja teknologioihin ilman merkittäviä muutoksia perustason toimintoihin. Sovelluksen rakenne voidaan suunnitella modulaariseksi, mikä tarkoittaa, että uusia ominaisuuksia voidaan lisätä erillisinä moduleina ilman, että olemassa oleva toiminnallisuus häiriintyy. Esimerkiksi materiaalien tunnistuskategoriat tai analyysialgoritmit voidaan lisätä järjestelmään modulaarisina laajennuksina.

Konfiguroitavuus on keskeinen osa laajennettavuutta. Järjestelmässä tulee olla mahdollisuus muuttaa toimintoja ja asetuksia helposti konfiguraatiotiedostojen avulla ilman koodimuutoksia. Esimerkiksi uusien materiaalikategorioiden lisääminen voidaan tehdä suoraan asetustiedostoista, mikä yksinkertaistaa järjestelmän ylläpitoa ja päivityksiä.

Laajennusten testausvalmius varmistetaan kattavilla testikehyksillä, jotka tukevat yksikkö-, integraatio- ja järjestelmätestejä. Testiautomaation avulla voidaan varmistaa, että uudet ominaisuudet eivät aiheuta yhteensopivuusongelmia järjestelmän olemassa olevissa toiminnoissa.

Sovelluksen voisi pakata konttisovellukseksi esimerkiksi Dockerin avulla. Tätä kokeiltiin, mutta toistaiseksi ei ole tarkoituksenmukaista laittaa Colab-koodia Dockeriin. Konttiteknologia tekisi sovelluksesta helposti siirrettävän, asennettavan ja käynnistettävän missä tahansa ympäristössä, joka tukee konttialustoja. Tämä mahdollistaa sovelluksen nopean käyttöönoton, skaalautuvuuden ja ylläpidon. Google Colab ei suoraan kuitenkaan tue Dockeria, mutta sitä voi käyttää etänä GCP:n, AWS:n tai oman palvelimen kautta.

Lopuksi, käyttäjäpalaute on tärkeä osa laajennettavuutta. Järjestelmään voidaan sisällyttää palautejärjestelmä, jonka avulla käyttäjät voivat ehdottaa uusia ominaisuuksia tai raportoida laajennustarpeita. Tämä tekee järjestelmästä käyttäjälähtöisen ja varmistaa, että se kehittyy vastaamaan todellisia tarpeita.