

Maksakuvien semanttinen segmentointi

(Liver Medical Image Semantic Segmentation)

Itä-Suomen yliopisto Luonnontieteiden ja metsätieteiden tiedekunta Tietojenkäsittelytieteen laitos

Denis Eskelinen ja Lauri Komulainen

Sisällys

[1 Projektin kuvaus ja ratkaisumenetelmät 3](#_Toc181731305)

[2 Toteutus 4](#_Toc181731306)

[2.1 Mallin koulutusstrategia ja tulokset 5](#_Toc181731307)

[2.2 Mallien suorituskyky 6](#_Toc181731308)

[3 Johtopäätökset 8](#_Toc181731309)

[4 Lainaukset ja viittaukset 9](#_Toc181731310)

[5 Itsearvoinnit 10](#_Toc181731311)

[Liitteet 12](#_Toc181731312)

# **1 Projektin kuvaus ja ratkaisumenetelmät**

Semanttinen segmentointi on kuvankäsittelytekniikka, jossa kuva jaetaan merkityksellisiin osiin siten, että jokaiselle pikselille määritellään oma luokka. Tämä mahdollistaa esimerkiksi maksan erottamisen muista elimistä ja kudoksista lääketieteellisissä kuvissa. Manuaalinen segmentointi on kuitenkin hidasta ja altis inhimillisille virheille, joten automaattiset segmentointimenetelmät tarjoavat mahdollisuuden vähentää ihmisten tekemiä virheitä ja nopeuttaa prosessia.

Tässä projektissa keskitytään kuvien segmentointiin, jossa tavoitteena on jakaa kuvat tarkasti eri alueisiin ja objekteihin pikselitasolla. Kuvasegmentointi on erityisen tärkeää lääketieteellisessä kuvantamisessa, jossa kudosten, elinten ja mahdollisten sairauksien tunnistaminen on olennaista sekä diagnoosin että hoidon suunnittelun kannalta. Tarkka segmentointi mahdollistaa kuvista saadun tiedon hyödyntämisen esimerkiksi automaattisessa analyysissä, jossa tietokoneavusteiset työkalut voivat tukea lääkäreiden työtä.

Syväoppiminen ja erityisesti U-Net-arkkitehtuuri ovat osoittautuneet erittäin tehokkaiksi lääketieteellisten kuvien segmentoinnissa. U-Net on konvoluutio-neuroverkko, joka on suunniteltu nimenomaan lääketieteellisten kuvien analysointiin. Sen rakenne yhdistää alas- ja ylöspäin suuntautuvat osat, mikä mahdollistaa sekä laajojen että yksityiskohtaisten piirteiden hyödyntämisen tarkassa segmentoinnissa. U-Net pystyy erottamaan kuvan alueet tarkasti pikselitasolla ja soveltuu erityisen hyvin tilanteisiin, joissa harjoitusdataa on saatavilla rajallisesti.

Projektin tavoitteena on rakentaa U-Net-arkkitehtuuriin perustuva neuroverkkomalli, joka kykenee saavuttamaan korkean tarkkuuden segmentointitulokset ja toimimaan luotettavasti myös pienellä datamäärällä.

# **2 Toteutus**

Mallimme on toteutettu Python-ohjelmointikielellä hyödyntäen PyTorch-koneoppimiskirjastoa. Mallin toteutus on jaettu kolmeen erilliseen tiedostoon:

1. **Main.py**: Tämä tiedosto käynnistää koulutusprosessin ja luo raportin.
2. **Model.py**: Tämä tiedosto sisältää mallin arkkitehtuurin ja logiikan mallin kouluttamiseksi erilaisilla hyperparametriyhdistelmillä.
3. **data\_handler.py**: Tämä tiedosto vastaa datan esikäsittelystä.

Malli perustuu U-Net-arkkitehtuuriin, ja sen rakenne koostuu seuraavista osista:

1. **Konvoluutiolohkot**: ***conv\_block*-** **funktio luo konvoluutiolohkoja, joissa on kaksi peräkkäistä konvoluutiokerrosta kernelikoolla (Kernel size) 3x3 ja reunuksella (Padding) 1. Jokaisen konvoluutiokerroksen jälkeen seuraa erän normalisointi (Batch Normalization) ja ReLU-aktivaatio. Ylikouluttamisen estämiseksi dropout-kerros on lisätty ensimmäisen aktivointikerroksen jälkeen.**
2. **Alasajovaihe (kontraktio):** Kerrokset *conv1–conv4* muodostavat alasajovaiheen. Jokainen kerros sisältää konvoluutiolohkon, jota seuraa maksimipoolauskerros *pool1–pool4*, jossa kernelikoko (Kernel size) on 2x2 ja askellus (Stride) 2. Tämä vaihe pienentää kuvan spatiaalista resoluutiota asteittain samalla, kun piirteiden (kanavien) määrä kasvaa. Tämä mahdollistaa sen, että verkko oppii korkeamman tason piirteitä.
3. **Pullonkaulakerros (bottleneck):** ***conv5* toimii pullonkaulana alasajo- ja ylösajovaiheiden välillä. Se koostuu konvoluutiolohkosta, joka sisältää kaksi konvoluutiokerrosta, erän normalisoinin (Batch Normalization), ReLU-aktivoinnin ja dropout-kerroksen.**
4. **Ylösajovaihe (ekspansio):** Transponoidut konvoluutiokerrokset *up6–up9* ja konvoluutiolohkot *conv6–conv9* muodostavat ylösajovaiheen. Tässä vaiheessa kuvan spatiaalinen resoluutio palautetaan asteittain alkuperäiseen kokoonsa. Lisäksi ylösajovaiheessa hyödynnetään skip connection -yhteyksiä, joissa alasajovaiheen ominaisuuskartat yhdistetään ylösnäytettyihin ominaisuuskarttoihin. Tämä mahdollistaa hienojakoisempien piirteiden säilyttämisen ja parantaa segmentointituloksia.
5. **Lähtökerros:** Lopuksi kerros *conv10* on 1x1 konvoluutio, joka muuntaa ominaisuuskartan haluttuun lähtökanavien määrään, tässä tapauksessa yhteen.

Mallin lähdekoodi on saatavilla julkisessa GitHub-repositorissa osoitteessa: <https://github.com/LauriKomulainen/Liver_Medical_Image_Semantic_Segmentation>.

## **2.1 Mallin koulutusstrategia ja tulokset**

Mallin koulutuksessa pyrimme löytämään optimaalisen hyperparametriyhdistelmän, joka maksimoisi suorituskyvyn ja parantaisi yleistymiskykyä. Yksittäisen koulutuskierroksen kesto vaihteli hyperparametrivalinnoista riippuen noin 1–4 tuntia, joten laajan grid search -haun käyttö ei ollut aikataulullisesti mahdollista. Tämän vuoksi hyödynsimme *main.py*-tiedostossa toteutettua satunnaista hyperparametrien valintaa.

Teimme yhteensä 15 koulutusta erilaisilla hyperparametriyhdistelmillä: 10 niistä suoritettiin Adam-optimointialgoritmilla ja 5 SGD-optimointialgoritmilla. Lisäsimme myös *model.py*-tiedostoon early stopping-mekanismin, joka keskeytti koulutuksen automaattisesti, jos validaatiohäviö ei parantunut kolmen peräkkäisen koulutuskierroksen (epochin) aikana. Tämä mahdollisti ylikoulutuksen havaitsemisen ja koulutuksen lopettamisen ennen suorituskyvyn heikkenemistä.

Koulutus suoritettiin kahdella tehokkaalla tietokoneella, joissa molemmissa oli suorituskykyinen prosessori ja näytönohjain. Satunnaisen haun avulla valitsimme hyperparametriyhdistelmiä seuraavista arvoista:

* Eräkoko (batch size): [4, 8, 16]
* Oppimisnopeus (learning rate): [1e-3, 1e-4, 1e-5]
* Dropout-määrä: [0.1, 0.2, 0.3, 0.4, 0.5]
* Optimoijatyyppi: [Adam, SGD]
* Epoch-määrä: [10, 15, 20]

## **2.2 Mallien suorituskyky**

Jokaisesta mallista on koottu oma raportti liitteisiin. Raportit löytyvät myös ’*model reports’*-kansiosta, joka sijaitsee lähdekoodin juuressa. Ne sisältävät keskeiset suorituskykymittarit, kuten mIoU- ja mPA-arvot, käytetyt hyperparametrit, harjoitus- ja validointitappiot sekä segmentointikuvat. Tulosten perusteella Malli 10 suoriutui parhaiten, ja sen hyperparametrit olivat:

* **Eräkoko**: 4
* **Oppimisnopeus**: 0.001
* **Dropout-arvo**: 0.1
* **Optimoija**: Adam
* **Kierrosten määrä (Epochs)**: 15

Tällä parametriyhdistelmällä saavutettiin mIoU-arvo **0.9749** ja mPA-arvo **0.9978**, mikä osoittaa mallin erinomaista segmentointitarkkuutta ja yleistymiskykyä validaatiodatan kanssa. Lopullinen malli tallennettiin *unet\_model.pth* -tiedostoon, mikä mahdollistaa sen suoran lataamisen ja käyttöönoton ilman uudelleenkoulutusta. Tallennettu *unet\_model.pth* malli on saatavilla lähdekoodin juuressa.

Alla ’Malli 10’ esimerkkikuvat: vasemmalla näkyy alkuperäinen kuva, keskellä todellinen segmentointi (label), ja oikealla mallin ennustama segmentointitulos.

Kuva 1, segmentation\_result\_012

Kuva, joka sisältää kohteen teksti, kuvakaappaus, musta, mustavalkoinen

Kuvaus luotu automaattisesti

Kuva 2, segmentation\_result\_015

Kuva, joka sisältää kohteen teksti, kuvakaappaus, musta, mustavalkoinen

Kuvaus luotu automaattisesti

Kuva 3, segmentation\_result\_016

Kuva, joka sisältää kohteen teksti, kuvakaappaus, musta, mustavalkoinen

Kuvaus luotu automaattisesti

# **3 Johtopäätökset**

Hyperparametrikokeilujen tavoitteena oli löytää optimaalisin yhdistelmä, joka maksimoi mallin suorituskyvyn ja yleistymiskyvyn segmentointitehtävässä. Jokaiselle mallille mitattiin keskeiset suorituskykymittarit, kuten mIoU- ja mPA-arvot, sekä visualisoitiin harjoitus- ja validointihävion kehitys.

Tulosten perusteella **Malli 10** osoittautui kokeilun parhaaksi malliksi, sillä se saavutti mIoU-arvon **0,9749** ja mPA-arvon **0,9978**, jotka olivat hieman parempia kuin Mallin 8 sekä Mallin 9 arvot. Mallin 10 hyvä suorituskyky vaikuttaisi olevan yhteydessä sen pieneen eräkokoon, keskiarvoiseen oppimisnopeuteen, matalaan dropout-arvoon sekä Adam-algoritmiin. Mallin häviökäyrät osoittavat myös selkeää laskua ilman merkkejä ylikoulutuksesta, mikä viittaa sen vahvaan yleistymiskykyyn.

Analysoimalla muita malleja huomasimme, että SGD-optimointialgoritmi oli yhteydessä heikkoon mIoU- sekä mPA-arvoon. Tämän lisäksi suurempi eräkoko sekä korkeampi dropout-arvo heikensivät mallien suorituskykyä mikä viittaa siihen, että pienempi eräkoko sekä matalampi dropout-arvo tukevat paremmin mallin kykyä oppia segmentoinnin hienovaraisia piirteitä.

Mallia koulutettiin varsin pienellä data määrällä. Jatkossa mallin kouluttaminen laajemmalla ja monipuolisemmalla datalla voisi merkittävästi parantaa mallin suorituskykyä. Suurempi datamäärä mahdollistaisi mallin oppia paremmin segmentoinnin hienovaraisia piirteitä. Suorituskykyä voitaisiin myös parantaa kokeilemalla dynaamisesti säätyvää oppimisnopeutta, jossa oppimisnopeus mukautuu koulutuksen aikana mallin suorituskyvyn mukaan. Tämä lähestymistapa voisi tehostaa koulutusta ja auttaa saavuttamaan optimaalisen tuloksen nopeammin, sillä oppimisnopeuden säätyminen koulutuksen edetessä voi auttaa mallia konvergoitumaan tehokkaammin ilman liiallista optimointia. Lisäksi muiden optimointialgoritmien, kuten AdamW ja Rangerin, kokeileminen voisi tuoda lisää suorituskykyparannuksia erityisesti suuremmilla aineistoilla.

# **4 Lainaukset ja viittaukset**

Mallin kehitysprosessissa ja suunnittelussa on hyödynnetty ChatGPT 4.0- ja o1-preview-malleja erityisesti koodin luomisessa ja teknisten ongelmien ratkaisemisessa. Tämä on nopeuttanut kehitystyötä ja auttanut syventämään ymmärrystä teknisistä haasteista. Mallin raportointiin liittyvät ratkaisut ja päätökset, kuten analyysien toteutustapa ja tulosten esitystapa, ovat meidän suunnittelemia ja toteuttamia.

Raportin kirjoittamisessa on käytetty ChatGPT:tä apuna kirjoitusvirheiden korjaamisessa ja oikoluvussa, mutta varsinainen teksti on kirjoitettu kokonaan meidän toimesta

# **5 Itsearvoinnit**

Ryhmätyö jaettiin tasaisesti molemmille. Kumpikin osallistui mallin koodaamiseen sekä raportin kirjoittamiseen. Alla molempien itsearvionti.

**Denis Eskelisen itsearvointi**:

Osallistuin aktiivisesti projektin toteuttamiseen alusta alkaen. Projektin alkuvaiheessa suunnittelimme koodin rakenteen ja pohdimme parhaita mahdollisia lähestymistapoja. Tämän jälkeen aloitin algoritmin pohjan toteuttamisen, jota kehitettiin edelleen yhdessä Laurin kanssa projektin loppuvaiheeseen asti. Projektin keskivaiheessa osallistuin mallin koulutukseen ja optimointiin, jotta saavutimme parhaan mahdollisen suorituskyvyn. Loppuvaiheessa olin mukana laatimassa loppuraporttia. Koko projekti toteutettiin Discordin kautta, ja käytimme ChatGPT:tä apuna tiedonhakuun, neuvojen saamiseen ja koodin kirjoittamiseen.

Suurimmat haasteet liittyivät mallin toteutukseen ja suunnitteluun. Prosessin aikana olen saanut syvällisemmän ymmärryksen mallin toiminnasta ja parametrien merkityksestä.

**Lauri Komulaisen itsearviointi**:

Osallistuin aktiivisesti projektin suunnitteluun ja toteutukseen alusta alkaen. Projektin alkuvaiheessa työskentelimme tiiviisti yhdessä Discordin välityksellä, mikä auttoi hahmottamaan projektin kokonaiskuvaa ja löytämään yhteisen suunnan mallin kehittämiselle. Osallistuin jokaiseen projektin vaiheeseen; mallin kodaamiseen, optimoimiseen sekä raportin kirjoittamiseen. Koodin luonnissa käytin apuna ChatGPT:tä, mikä nopeutti ongelmanratkaisua ja auttoi kokeilemaan eri lähestymistapoja tehokkaasti.

Kokonaisuutena haastavinta projektissa oli mallin arkkitehtuurin suunnittelu, erityisesti kerrosrakenteen ja hyperparametrien optimoinnin osalta. Parhaan optimointiyhdistelmän löytäminen vaati paljon kokeilua ja tarkkaa analysointia, sillä pienetkin muutokset saattoivat vaikuttaa suorituskykyyn merkittävästi. Tässä prosessissa sain syvemmän ymmärryksen mallin optimoinnista ja hyperparametrien merkityksestä.

Opin myös analysoimaan mallin tuloksia ja tulkitsemaan koulutuskäyriä, mikä auttoi arvioimaan, miten eri muutokset vaikuttivat mallin suorituskykyyn. Tämä projekti antoi erinomaisen käsityksen mallin kehittämisestä ideasta toteutukseen ja tarjosi arvokasta kokemusta siitä, mitä osa-alueita on tärkeä tarkastella mallin tehokkuuden varmistamiseksi.

# **Liitteet**

Malli 1,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Kuvaus luotu automaattisesti

Malli 2,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli 3,

Kuva, joka sisältää kohteen teksti, diagrammi, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Malli 4,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli 5,

Kuva, joka sisältää kohteen teksti, diagrammi, kuvakaappaus, Tontti

Kuvaus luotu automaattisesti

Malli 6,

Kuva, joka sisältää kohteen teksti, diagrammi, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Malli 7,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Kuvaus luotu automaattisesti

Malli 8,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Kuvaus luotu automaattisesti

Malli 9,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli 10,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli, 11

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli 12,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti, diagrammi

Kuvaus luotu automaattisesti

Malli, 13

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti

Malli 14,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Kuvaus luotu automaattisesti

Malli 15,

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, Fontti

Kuvaus luotu automaattisesti