AARHUS UNIVERSITET

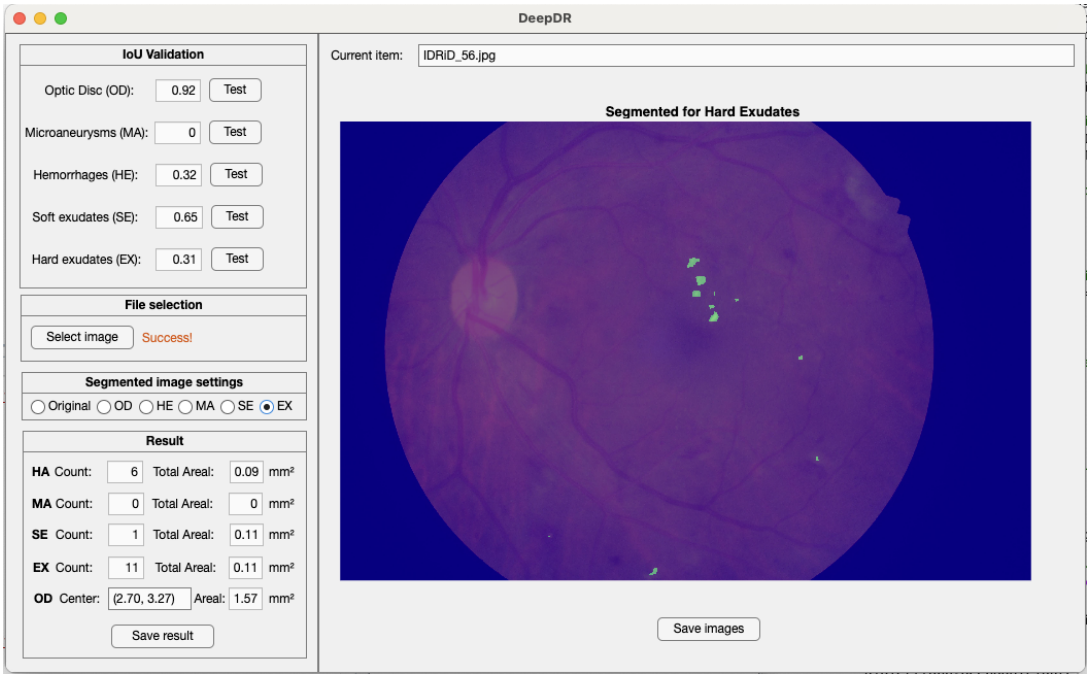
Institut for Elektro- og Computerteknologi
Foråret 2024

Eksamensopgave 3

Diabetisk retinopati

Søren Mehlsen - 202000071

Dato: 27/05-2024



STTBDP

Aarhus Universitet

Indhold

1 Indledning

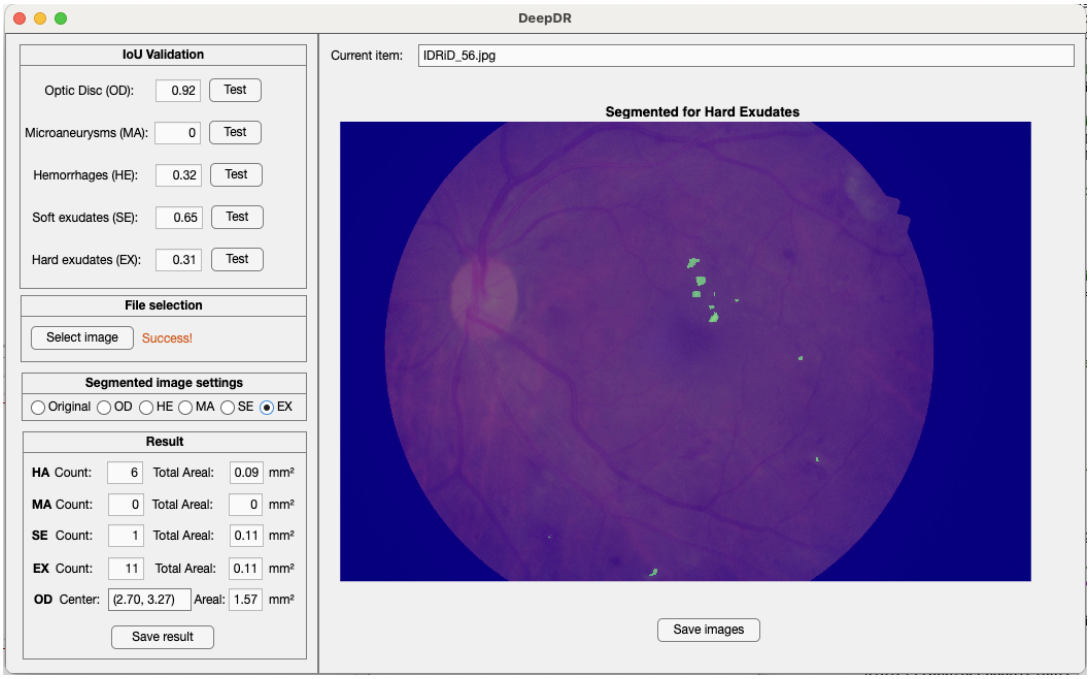
Denne rapport præsenterer udviklingen af et brugervenligt brugerinterface (GUI) i MATLAB og segmenteringsalgoritmer til detektion af læsioner forbundet med diabetisk retinopati, en alvorlig senkomplikation ved sukkersyge (diabetes). Diabetisk retinopati er den hyppigste årsag til forebyggelige synsnedsættelser, hvilket gør tidlig diagnostik afgørende for at forhindre efterfølgende synshandicaps.

Diabetes er i stigende grad en livsstilssygdom, der vokser hurtigt i udviklingslande som følge af øget velstand og livsstilsændringer. I disse regioner, hvor sundhedssektoren ofte har begrænsede ressourcer, kan automatiserede værktøjer til diagnosticering af diabetisk retinopati være særdeles vigtige for effektive screeningsprogrammer. Denne rapport fokuserer på at udvikle algoritmer, der kan detektere fire specifikke typer læsioner: mikroaneurismer (MA), hemorrhagier (HE), bløde eksudater (SE) og hårde eksudater (EX).

Formålet med segmenteringsalgoritmerne er at lave et pålideligt og effektivt værktøj, der kan bruges af sundhedspersonale i områder med begrænsede ressourcer. Ved at automatisere processen for detektion af læsioner minimeres risikoen for menneskelige fejl og forbedre diagnostikens nøjagtighed, hvilket muliggør hurtige og præcise vurderinger. Dette kan betydeligt forbedre mulighederne for tidlig intervention og behandling, hvilket er afgørende for at reducere risikoen for alvorlige synshandicaps hos patienter med diabetes.

2 Arkitektur og design

2.1 GUI



Figur 2.1: Design af GUI: Foretaget en segmentering på billedet 'IDRID_56.jpg'

I figur 2.1 ses den grafiske brugergrænseflade(GUI), som er udarbejdet til at kunne detektere 4 forskellige læsioner samt den optiske disk. Her er en gennemgang af de forskellige funktioner på GUI'en:

IoU Validation: Formålet med IoU Validation er at kunne vise Intersection over Union (IoU) score for segmenteringen for hver læsion og den optiske disk, så brugeren hurtigt kan vurdere præstationen af segmenteringsmodellen. IoU er valgt, fordi det er en metric, der bruges til at evaluere præcisionen af objektdetektering. Opdelingen med test knapper for hver enkelt læsion i stedet for en, skyldes tiden det tager for at segmentere og beregne IoU ud fra datasættet, da dette tager tid. Fremtidige forbedringer ville være at optimere tiden for at implementere en enkelt test-knap.

File selection: Formålet med file selection er at give brugeren mulighed for at kunne uploade et billede til analyse ved tryk af select image knappen, for at gøre det nemt og simpelt for brugeren at komme i gang med segmenteringsprocessen.

Segmented image settings: Formålet med segmented image settings er at give brugeren

mulighed for at vælge hvilken type segmentering de ønsker at se. Dette er valgt for at give brugeren fleksibilitet til selv at kunne vælge og få visualiseret specifikke segmenterede områder, for at gøre det mere overskueligt, så brugeren lettere kan fokusere på hvad der er relevant for dem.

Result: Formålet med result er at kunne vise resultaterne af segmenteringen, som er antallet af hver type læsion og deres samlede størrelse. Dette giver en klar oversigt over segmenteringsanalysen, hvilket gør det nemmere for brugeren at forstå resultaterne uden selv at skulle analysere billedet.

Visualisering af segmenteringen: Dette består af at kunne vise billederne af de segmenterede læsioner baseret på den valgte type læsion, så brugeren har mulighed for at kunne se det orginale billede, som de har valgt og de segmenterede billeder. Brugeren har selv mulighed for at verificere nøjagtigheden af segmenteringen og bedre identificere placeringen af de enkelte læsioner og den optiske disk.

Gem resultater: Dette giver brugeren mulighed for at kunne gemme resultaterne og segmenteringsbillederne. Det giver brugeren mulighed for at kunne dokumentere resultaterne og lave en yderligere analyse til senere brug.

Brugervenlighed:

For at gøre GUI'en mere brugervenligt for sundhedspersonalet er designet lavet med enkle knapper og paneler, der gør det nemt for brugeren at navigere og udføre de nødvendige handlinger. Knapperne er gjort aktive/ikke aktive afhængig af hvor brugeren er i processen i programmet for at undgå fejl samt at kunne navigere brugeren til hvad den næste handling er i processen.

Ved at gruppere relaterede funktioner sammen (som IoU validation, filvalg og resultater) minimeres behovet for at skifte mellem forskellige dele af progrmmaet, hvilket gør det mere effektivt for brugeren at bruge.

Der er arbejdet på visuel feedback ved visning af de segmenterede billeder, hvilket giver en hurtig feedback på deres valg. Der gives øjeblikkelig feedback ved tryk på de forskellige knapper i form af status på processen og når det er udført. Dette er for at undgå fejl og forvirringer omkring om der er blevet trykket på knappen.

Der er arbejdet på at give fleksibilitet til brugeren, som gør det muligt, at de selv kan vælge mellem forskellige segmenteringer, gemme resultater og validering af programmets segmenteringsmodel. Dette giver brugeren kontrol over processen og mulighed for at tilpasse

analysen til deres specifikke behov, samtidig med fokus på at automatisere processen.

2.2 Sekvens diagram

Der er udarbejdet et sekvensdiagram, som giver et overblik over workflowet og interaktionerne i systemet "DeepDR". Diagrammet afdækker de sekventielle trin, som en bruger skal gennemgå for at udføre analysen, som ses i figur 2.2.

Beskrivelse af sekvensdiagrammet:
Sekvensdiagrammet beskriver to primære processer: validering af IoU (Intersection over Union) over hver enkelt læsion og segmentering ud fra eget billedvalg.

Validering af IoU
Processen starter, når brugeren klikker på en af testknapperne i GUI'en. Herefter bliver funktionen validationIoU() kaldt, og applikationen forsøger at indlæse det trænede netværk for den valgte læsion.

Hvis netværksfilen findes, returneres den til applikationen, som derefter segmenterer træningsbillederne og deres tilhørende ground truth billeder samt beregner IoU. Resultatet af denne beregning sendes tilbage til GUI'en, hvor det vises til brugeren.

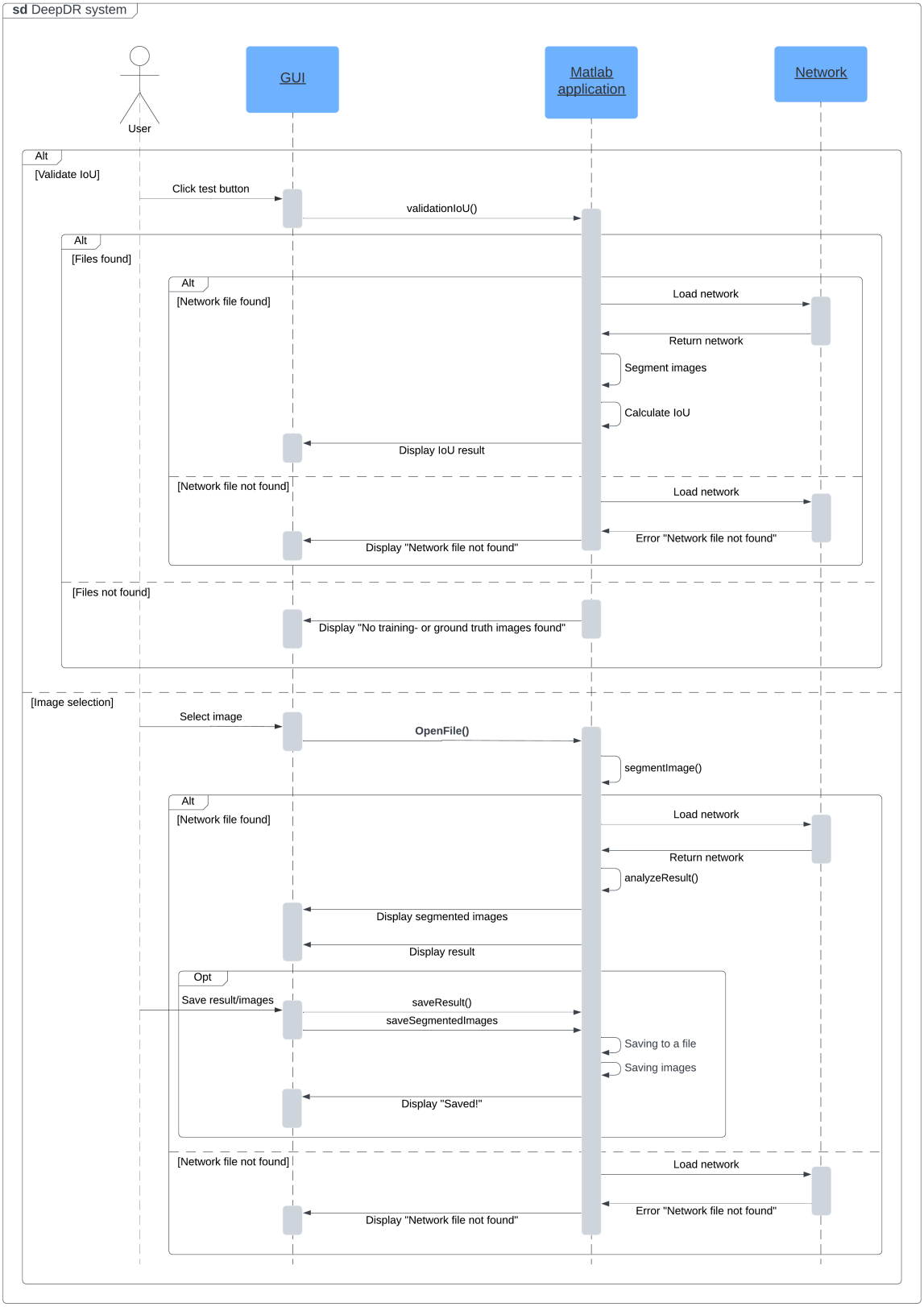
Hvis netværksfilen ikke kan findes, returneres en fejlmeddelelse, som vises i GUI'en med beskeden "Network file not found". Hvis trænings- eller ground truth-billederne ikke findes, vises beskeden "No training- or ground truth images found" i GUI'en.

Billedvalg
Denne proces begynder, når brugeren klikker på 'Select image' knappen i GUI'en. Herefter bliver funktionen OpenFile() kaldt og applikationen forsøger at indlæse de trænede netværk for hver læsion.

Hvis netværksfilen findes, returnerer den til applikationen, som segmenterer det valgte billede og analyserer resultatet. De segmenterede billeder og analyseresultatet sendes derefter tilbage til GUI'en, hvor de vises til brugeren.

Brugeren har også mulighed for at gemme resultatet og de segmenterede billeder. Hvis denne mulighed vælges, bliver funktionerne saveResult() eller saveSegmentedImages() kaldt. Applikationen gemmer derefter resultaterne og billederne, og en besked "Saved!"

vises i GUI'en.



Figur 2.2: Sekvens diagram af DeepDR-systemet

3 Segmentering ved brug af Deep Learning

Til segmentering af de forskellige typer læsioner og den optiske disk, er der blevet anvendt deep learning. Koden blev lavet i en seperat MATLAB script og benytter DeepLabv3+ modellen med en ResNet-50 backbone til at segmentere billederne.

DeepLabv3+ blev valgt som model til segmentering af læsioner i nethinden i forbindelse med diabetisk retinopati. Forskning viser, at DeepLabv3+ kan præcist segmentere forskellige typer retinale læsioner som mikroaneurismer, blødninger og eksudater. Modellen udnytter dilaterede konvolutioner til at bevare rumlige detaljer, og dens multiskala feature extraction forbedrer segmenteringsnøjagtigheden. Studier har valideret dens effektivitet på åbne datasæt som DIARETDB1 og IDRiD, hvor resultaterne har vist bedre præstationer sammenlignet med andre metoder.[1]

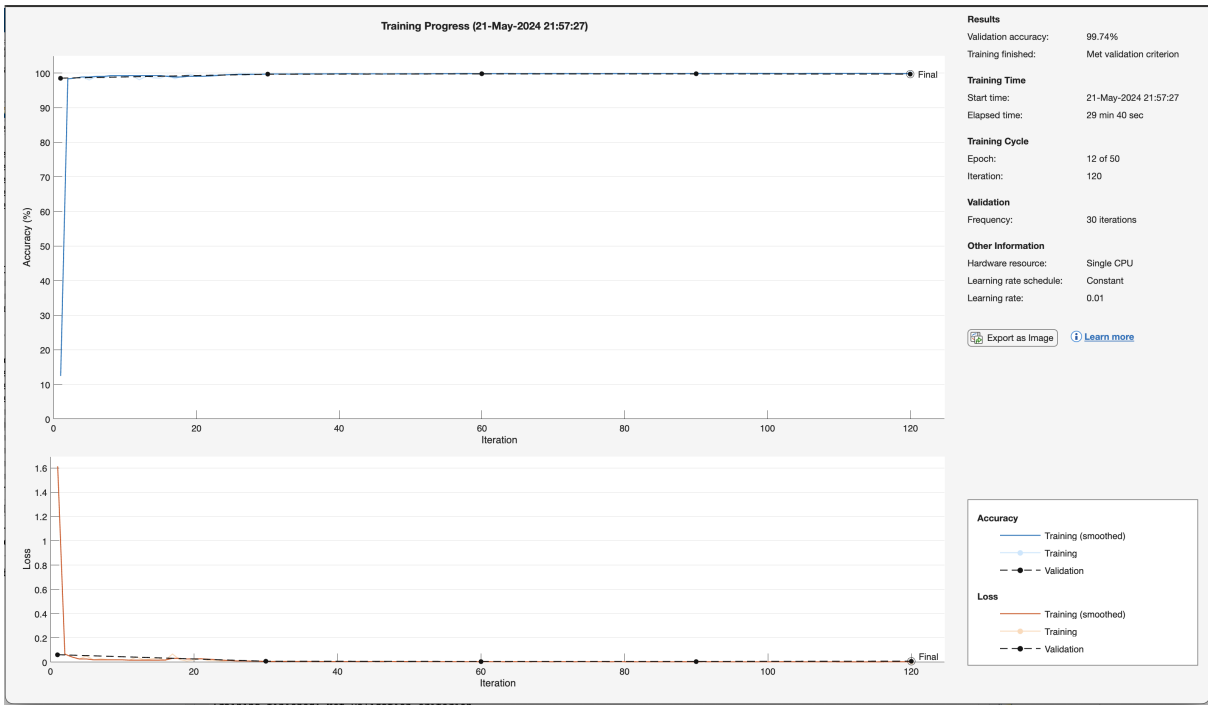
Et udsnit af MATLAB scriptet i figur 3.1 er blevet udarbejdet ud fra Matlab dokumentationen for DeepLabv3+[2]. Her ses hvilke indstillinger der blev brugt for at træne modellen til at kunne detektere den optiske disk. Denne fremgangmåde blev brugt for hver type læsion i stedet for at træne modellen med alle læsionerne på en gang, da der opstod fejl. Træning af modellen bestod af de udleverede træningsbilleder sammen med deres ground truth billeder.

```
lgraph = deeplabv3plusLayers(imageSize, numClasses, 'resnet50');

% Træningsindstillinger
opts = trainingOptions("sgdm", ...
    'MiniBatchSize', 4, ...
    'MaxEpochs', 50, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', tdsVal, ... % Tilføjer valideringsdata
    'ValidationFrequency', 30, ...
    'ValidationPatience', 2,...
    'Plots', 'training-progress');

% Træn netværket
net = trainNetwork(tdsTrain, lgraph, opts);
```

Figur 3.1: MATLAB-snippet: Udsnit af Matlab script til træning af netværket for den optiske disk

Forklaring til valget af træningsindstillinger: En minibatch på 4, blev valgt for at give modellen en lille del af træningsdataene for at få hyppigere opdateringer, hvilke kunne hjælpe modellen med at finde en hurtigere løsning. En MaxEpochs angiver det maksimale antal epoker, som modellen skal trænes i, hvor en epoke er en fuld gennemgang af hele træningsdatasættet. Valget på 50 epoker ville give modellen tiltrækkelig tid til at lære fra dataene. Der blev lavet en validationData for modellen, som blev brugt til at evaluere modellens ydeevne efter hver epoke. Dette var for at kunne tjekke modellens generaliseringsevne og for at forhindre overfitting (overtræning). ValidationFrequency og ValidationPatience blev benyttet for at kunne få evalueringer for hver 30. iteration og få opdateringer på modellens ydeevne. En validationPatience på 2 ville gøre, at træningen stoppede hvis valideringstabet ikke blev forbedret efter 2 omgange. Dette var igen for at undgå overfitting og unødvendig træning.

Ud fra figur 3.2 ses en graf over træningens fremgang for den optiske disk. Her kan det tydeligt ses, at modellen havde nemt ved at lære at detektere den optiske disk. Dette ses ud fra den blå graf, som viser modellens nøjagtighed, som er hurtigt stigende i starten og forbliver høj. Ud fra den røde graf, som viser modellens tab (loss), falder den også hurtigt og forbliver lav, hvilket understøtter modellens evne til at detektere den optiske disk. Ved at implementere en validering, stoppede træningen af sig selv efter 12 træningsomgange.



Figur 3.2: En graf over træningens fremgang på OD, herunder tab og nøjagtighed over tid.

3.1 Beregning af IoU

Intersection over Union (IoU) er en måling, der anvendes til at evaluere præstationen af et segmenteringsnetværk ved at sammenligne netværkets forudsigelser med de faktiske ground truth labels. IoU er defineret som forholdet mellem mængden af overlap (intersection) mellem forudsigelsen og ground truth og mængden af union (foreningen) af disse to sæt. IoU værdi kan variere fra 0 til 1, hvor 1 angiver perfekt overlap mellem forudsigelsen og ground truth.

Formlen for IoU:

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$

Til beregning af IoU skulle hvert billede anvendes til det trænede netværk til at forudsige segmenteringsklassen for hver pixel. Derefter kunne forudsigelserne sammenlignes med de faktiske ground truth billeder, som angiver den korrekte klasse for hver pixel. Her kunne MATLABs jaccard-funktion anvendes som IoU, da den måler ligheden mellem to sæt.[3]

4 Funktioner

4.1 ValidationIoU

Funktionen i figur 4.1 og 4.2 fungere ved at validere nøjagtigheden af et trænet neuralt netværk for hver læsion ved at sammenligne dets resultater med kendte sandhedsværdier for diabetisk retinopati. Den sikre at det trænede netværk præcist kan identificere og klassificere læsioner ud fra et billede ved at beregne Intersection over Union (IoU) for hver klasse, hvilket giver en målestok for modellens præstation.

```
function validationIoU(app, regionType)
    % Initialiser en waitbar for at informere brugeren om status
    wb = waitbar(0, 'Loading data...');
    waitbar(0.1, wb, 'Checking for trainingfiles and ground truth files...');

    % Mappestier for træningsbilleder og deres ground truth-filer
    trainingImagesDir = 'DiabeticRetinopathyDataset/a. Training Set';
    GroundTruthImagesDir = 'DiabeticRetinopathyDataset/a. Training Set Groundtruths';

    % Liste over træningsbilleder og ground truth-filer
    trainingImagesFiles = dir(fullfile(trainingImagesDir, '*.jpg'));
    groundTruthPattern = ['*_' regionType '.tif'];
    groundTruthImagesFiles = dir(fullfile(GroundTruthImagesDir, groundTruthPattern));

    % Opretter lister over filnavne uden stier og uden filudvidelser
    imageFileNames = erase({trainingImagesFiles.name}, '.jpg');
    groundTruthPattern_ = ['_' regionType '.tif'];
    annotationFileNames = erase({groundTruthImagesFiles.name}, groundTruthPattern_);

    % Finder de fælles filnavne mellem billeder og annotationsfiler
    [~, imageIdx, annotationIdx] = intersect(imageFileNames, annotationFileNames);

    % Filtrer imageFiles og annotationFiles så de kun indeholder de fælles filer
    trainingImagesFiles =  trainingImagesFiles(imageIdx);
    groundTruthImagesFiles = groundTruthImagesFiles(annotationIdx);

    % Tjekker for træningsbilleder og ground truth-filer
    if isempty(trainingImagesFiles) || isempty(groundTruthImagesFiles)
        uialert(app.DeepDRUIFigure, error.message, 'No training images or ground truth files found. Check path
            and filenames.');  % Show a error message
    end

    % Tjekker antallet af træningsbilleder passer med antallet af ground truth-filer
    if numel(trainingImagesFiles) ~= numel(groundTruthImagesFiles)
        uialert(app.DeepDRUIFigure, error.message, 'The number of training images does not match the number of
            ground truth files.');
    end

    % Definer klassernes navne og labels
    if strcmp(regionType, 'EX')
        classNames = ["Background", 'Exudates'];
    else
        classNames = ["Background", regionType];
    end
    labelIDs = [0 1]; % 0 for baggrund, 1 for læsion typen
    numClasses = numel(classNames); % Antal af klasser

    % Initialiser metrikker
    iouValues = zeros(numel(trainingImagesFiles), numClasses);

    % Indlæs det trænede netværk for den valgte læsion
    networkFile = ['trainedNetwork_' regionType '.mat'];
    try
        loadNetwork = load(networkFile);
        net = loadNetwork.net;
    catch
        uialert(app.DeepDRUIFigure, error.message, ['The network file "' networkFile '" was not found. ' ...
            'Ensure that the file is in the correct folder and try again.']);
    end
```

Figur 4.1: MATLAB-snippet: ValidationIoU funktion

4.2 ValidationIoU

```
% Opdater waitbar status
   waitbar(0.5, wb, 'Segment trainingsfiles...');

   % Finder antallet af træningsbilleder
   numFiles = numel(trainingImagesFiles);

   % Iterer gennem træningsbillederne
   for idx = 1:numel(trainingImagesFiles)
       % Indlæser træningsbillede
       trainingImage = imread(fullfile(trainingImagesFiles(idx).folder, trainingImagesFiles(idx).name));
       % Juster størrelsen på træningsbilledet til kravet for
       % netværket
       trainingImageResized = imresize(trainingImage, app.imageSize(1:2));
       % Indlæser tilhørende ground truth billede
       groundTruth = imread(fullfile(groundTruthImagesFiles(idx).folder, groundTruthImagesFiles(idx).name));
       % Juster størrelsen på ground truth billedet til kravet for
       % netværket
       groundTruthResized = imresize(groundTruth, app.imageSize(1:2), 'nearest');
       % Kategoriser ground truth med den ændrede billede
       % størrelse, label IDs og klassenavne
       groundTruthCategorical = categorical(groundTruthResized, labelIDs, classNames);
       % Segmenter træningsbilledet med netværket
       C = semanticseg(trainingImageResized, net);
       % Beregner IoU ud fra det segmenteret træningsbillede og
       % tilhørende ground truth billede
       iou = jaccard(C, groundTruthCategorical);
       iouValues(idx, :) = iou;
       % Opdater waitbar status på antallet af billeder tilbage
       waitbar(0.5 + 0.4 * (idx / numFiles), wb, sprintf('Segment %d of %d images...', idx, numFiles));
   end

   % Opdater waitbar status
   waitbar(0.9, wb, ['Calculating IoU for ' regionType '...']);

   % Beregner gennemsnitlig IoU for hver klasse
   meanIoU = mean(iouValues, 1);

   % Angiver gennemsnitlig IoU til the rigtige felt
   switch regionType
       case 'OD'
           app.ODIoUField.Value = round(meanIoU(2), 2);
       case 'SE'
           app.SEIoUField.Value = round(meanIoU(2), 2);
       case 'EX'
           app.EXIoUField.Value = round(meanIoU(2), 2);
       case 'HE'
           app.HEIoUField.Value = round(meanIoU(2), 2);
       case 'MA'
           app.MAIoUField.Value = round(meanIoU(2), 2);
       otherwise
           error('Unknown region type.');
   end

   % Indstiller waitbar til fuldført
   waitbar(1, wb, 'Complete!');
   pause(0.5); % Pauser waitbar så brugeren kan se, at det er fuldført
   close(wb); % Lukker waitbar
end
```

Figur 4.2: MATLAB-snippet: ValidationIoU part 2 funktion

4.3 OpenFile

Funktionen i figur 4.3 fungere ved at give brugeren mulighed for at vælge og indlæse et billede, som bliver aktiveret ved tryk af select image knappen.

```
function openFile(app)
     [app.file,app.path] = uigetfile('*.jpg');
    if isequal(app.file,0)
        return
    else
        % Opdatere feltet for the nuværende billede
        app.CurrentImageField.Value = app.file;
        try
            % Indlæser og viser the valgte billede
            showInputImage(app);
        catch e
            % Giver en feedback til brugeren om at der er
               sket en
            % fejl
            uialert(app.DeepDRUIFigure, 'Failed to load the
               image. Please try another image.', 'Image Load
                Error', 'Icon', 'error');
        end
    end
end
```

Figur 4.3: MATLAB-snippet: OpenFile funktion

4.4 ShowInputImage

Funktionen i figur 4.4 fungere ved at indlæse og vise det valgt billede i GUI'en, som bliver aktiveret efterfulgt af openFile funktionen.

```
function showInputImage(app)
    % Indlæser the valgte billede
    app.inputImage = imread(fullfile(app.path,app.file));
    % Viser the valgte billede i GUI
    imshow(app.inputImage,'Parent',app.UIAxes);
    app.UIAxes.Title.String = "Original Image";
end
```

Figur 4.4: MATLAB-snippet: ShowInputImage funktion

4.5 SelectedImage

Funktionen i figur 4.5 fungere ved at opdatere det viste billede i GUI'en baseret på brugerens valg af segmenteringsindstillinger.

```
function selectedImage(app)
    % Henter den seneste valgte knap
    selectedButton = app.SegmentedimagesettingsButtonGroup.SelectedObject;

    % Baseret på den valgte knap opdateres billedet i GUI
    switch(selectedButton.Text)
        case "Original"
            imshow(app.inputImage,'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Original Image";
        case "SE"
            imshow(app.labeloverlaySEImage, 'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Segmented for Soft Exudates";
        case "OD"
            imshow(app.labeloverlayODImage, 'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Segmented for Optic Disc";
        case "HE"
            imshow(app.labeloverlayHEImage, 'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Segmented for Hermorrhages";
        case "MA"
            imshow(app.labeloverlayMAImage, 'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Segmented for Microaneurysms";
        case "EX"
            imshow(app.labeloverlayEXImage, 'Parent',app.UIAxes);
            app.UIAxes.Title.String = "Segmented for Hard Exudates";
    end
end
```

Figur 4.5: MATLAB-snippet: SelectedImage funktion

4.6 SegmentImage

Funktionen i figur 4.6 fungere ved at segmentere et indlæst billede for forskellige typer læsioner ved af hjælp af de trænede neurale netværk.

```
function segmentImage(app)
    % Gemmer den oprindelige størrelse på det valgte billede
    originalSize = size(app.inputImage);

    % Ændre størrelsen på billedet ud fra netværkskravet
    imageResized = imresize(app.inputImage, app.imageSize(1:2));

    % Definer Filnavne på det trænede netværk samt labels
    networkFiles = {'trainedNetwork_SE.mat', 'trainedNetwork_OD.mat', 'trainedNetwork_HE.mat', '
        trainedNetwork_MA.mat', 'trainedNetwork_EX.mat'};
    app.segmentLabels = {'SE', 'OD', 'HE', 'MA', 'Exudates'};

    % Initialiser en celle array til at holde de segmenterede billeder
    app.segmentedImages = cell(1, length(networkFiles));

    % Giver en feedback på segmenteringen til brugeren
    app.SelectImageStatusLabel.Text = 'The segment is starting...';
    drawnow; % Sørger for at opdatere GUI

    % Looper gennem hvert netværk og udfør segmentering
    for i = 1:length(networkFiles)

        % Viser en status på segmenteringen til brugeren
        app.SelectImageStatusLabel.Text = ['Segment ', app.segmentLabels{i} '...'];
        drawnow;

        % Indlæser det gemte netværk
        try
            loadNetwork = load(networkFiles{i});
            net = loadNetwork.net;
        catch
            uialert(app.DeepDRUIFigure, error.message, ['The network file "' networkFiles{i} '" was not found.
                ' ...
              'Ensure that the file is in the correct folder and try again.']);
        end

        % Segmenter det justerede billede med netværket
        app.segmentedImages{i} = semanticseg(imageResized, net);
    end

    % Ændre det segmentede billede størrelse til den oprindelige
    % størrelse
    for i = 1:length(app.segmentedImages)
        app.segmentedImages{i} = imresize(app.segmentedImages{i}, originalSize(1:2), 'nearest');
    end

    % Kombiner det valgte billede med det segmenterede billede for
    % hver læsion
    app.labeloverlaySEImage = labeloverlay(app.inputImage, app.segmentedImages{1}, 'Transparency', 0.5);
    app.labeloverlayODImage = labeloverlay(app.inputImage, app.segmentedImages{2}, 'Transparency', 0.5);
    app.labeloverlayHEImage = labeloverlay(app.inputImage, app.segmentedImages{3}, 'Transparency', 0.5);
    app.labeloverlayMAImage = labeloverlay(app.inputImage, app.segmentedImages{4}, 'Transparency', 0.5);
    app.labeloverlayEXImage = labeloverlay(app.inputImage, app.segmentedImages{5}, 'Transparency', 0.5);
end
```

Figur 4.6: MATLAB-snippet: SegmentImage funktion

4.7 SaveSegmentedImages

Funktionen i figur 4.7 fungere ved at give brugeren mulighed for at gemme segmenterede billeder i en valgt filplacering og filformat.

```
function saveSegmentedImages(app)
    % Lad brugeren vælge en filplacering og navn
    [app.file, app.path] = uiputfile({'*.tif', 'TIFF image';
       '*.mat', 'MAT-file'}, 'Save Segmented Image');
    if isequal(app.file, 0) || isequal(app.path, 0)
        return;
    end

    % Kombiner sti og filnavn
    fullPath = fullfile(app.path, app.file);

    % Gem hver segmenteret billed-label som en TIFF-fil eller
        MAT-fil
    if endsWith(app.file, '.tif')
        [~, name, ext] = fileparts(fullPath);
        imwrite(app.labeloverlaySEImage, fullfile(app.path, [
           name '_SE' ext]));
        imwrite(app.labeloverlayODImage, fullfile(app.path, [
           name '_OD' ext]));
        imwrite(app.labeloverlayHEImage, fullfile(app.path, [
           name '_HE' ext]));
        imwrite(app.labeloverlayMAImage, fullfile(app.path, [
           name '_MA' ext]));
        imwrite(app.labeloverlayEXImage, fullfile(app.path, [
           name '_EX' ext]));

        % Feedback til brugeren om billederne blev gemt
        app.SaveImagesStatus.Text = 'Saved!';
    elseif endsWith(app.file, '.mat')
        % Gem segmenteringsresultater som MAT-fil
        save(fullPath, 'app.segmentedSEImage', 'app.
           segmentedODImage', 'app.segmentedHEImage', 'app.
           segmentedMAImage', 'app.segmentedEXImage');

        % Feedback til brugeren om billederne blev gemt
        app.SaveImagesStatus.Text = 'Saved!';
    end
end
```

Figur 4.7: MATLAB-snippet: SaveSegmentedImages funktion

4.8 AnalyzeResult

Funktionen i figur 4.8 fungere ved at analysere segmenteringsresultaterne fra et billede ved at beregne det samlede areal og antallet af de forskellige typer læsioner samt centrummet og arealet af den optiske disk.

```
function analyzeResult(app)
    pixelSize = 0.003; % Har antaget en værdi, men skal opdates med den rigtige opløsning (i millimeter pr. pixel)
    pixelArea = pixelSize^2; % Beregner arealet af en pixel i kvadratmillimeter

    % Giver feedback til brugeren om, at resultaterne bliver beregnet
    app.SelectImageStatusLabel.Text = 'Calculate the result...';
    drawnow;

    % Initialiser en struktur til at gemme segmenteringsresultaterne
    app.segmentationResults = struct();


    % Loop gennem hver type af segmenteringslabels
    for i = 1:length(app.segmentLabels)
        label = app.segmentLabels{i}; % Henter den aktuelle label
        segmentedImage = app.segmentedImages{i}; % Henter det segmenterede billede for den aktuelle label
        BW = segmentedImage == label; % Opreter en binær maske for den aktuelle label

        if strcmp(label, 'OD') % Hvis label er 'OD'
            % Beregn centrum og areal for OD
            stats = regionprops(BW, 'Centroid', 'Area');
            if ~isempty(stats)
                OD_Centroid = stats.Centroid * pixelSize; % Konverter centrum til millimeter
                OD_Area = stats.Area * pixelArea; % Konverter arealet til kvadratmillimeter
                app.segmentationResults.OD = struct('Centroid', OD_Centroid, 'Area', OD_Area);
            else
                app.segmentationResults.OD = struct('Centroid', [], 'Area', 0); % Hvis ingen OD fundet, sæt default værdien
            end
        else
            % Beregner antal og samlet areal for de andre læsioner
            stats = regionprops(BW, 'Area');
            count = numel(stats); % Antallet for den givne læsion
            totalArea = sum([stats.Area]) * pixelArea; % Samlet areal for den givne læsion konverteret til kvadratmillimeter
            app.segmentationResults.(label) = struct('Count', count, 'TotalArea', totalArea);
        end
    end

    % Viser resultaterne for hver læsion i GUI
    app.ODArealField.Value = round(app.segmentationResults.OD.Area, 2);
    app.ODCenterField.Value = sprintf('(%.2f, %.2f)', app.segmentationResults.OD.Centroid(1), app.segmentationResults.OD.Centroid(2));

    app.SETotalArealField.Value = round(app.segmentationResults.SE.TotalArea, 2);
    app.SECountField.Value = app.segmentationResults.SE.Count;

    app.HETotalArealField.Value = round(app.segmentationResults.HE.TotalArea, 2);
    app.HECountField.Value = app.segmentationResults.HE.Count;

    app.MATotalArealField.Value = round(app.segmentationResults.MA.TotalArea, 2);
    app.MACountField.Value = app.segmentationResults.MA.Count;

    app.EXTotalArealField.Value = round(app.segmentationResults.Exudates.TotalArea, 2);
    app.EXCountField.Value = app.segmentationResults.Exudates.Count;

    % Feedback på segmenteringen til brugeren
    app.SelectImageStatusLabel.Text = 'Success!';
end
```

Figur 4.8: MATLAB-snippet: AnalyzeResult funktion

4.9 SaveResult

Funktionen i figur 4.9 fungere ved at gemme analyseresultaterne fra segmenteringen som en tekstfil.

```
function saveResult(app)
            % Gemmer resultatet som en enkelt tekst fil
            [app.file,app.path] = uiputfile({'*.txt','TXT-files'},'Save result as
                ');
            if app.file ~= 0
                output{1} = app.file;
                output{2} = round(app.segmentationResults.OD.Area, 2);
                output{3} = sprintf('(%.2f, %.2f)', app.segmentationResults.OD.
                    Centroid(1), app.segmentationResults.OD.Centroid(2));
                output{4} = round(app.segmentationResults.SE.TotalArea, 2);
                output{5} = app.segmentationResults.SE.Count;
                output{6} = round(app.segmentationResults.HE.TotalArea, 2);
                output{7} = app.segmentationResults.HE.Count;
                output{8} = round(app.segmentationResults.MA.TotalArea, 2);
                output{9} = app.segmentationResults.MA.Count;
                output{10} = round(app.segmentationResults.Exudates.TotalArea, 2)
                    ;
                output{11} = app.segmentationResults.Exudates.Count;

                % Opretter en tabel fra celle arrayen med navngivne kolonner
                outputTable = cell2table(output,'VariableNames',{'Filename','OD
                    Area','OD Center', 'SE Total Area', 'SE Count', 'HE Total
                    Area', 'HE Count', 'MA Total Area', 'MA Count', 'EX Total
                    Area', 'EX Count'});

                % Gemmer tabellen til en CSV-fil med semikolon som delimiter
                writetable(outputTable,fullfile(app.path,app.file),'Delimiter',';
                    ');

                % Feedback til brugeren om resultatet blev gemt
                app.SaveResultStatus.Text = 'Saved!';
            end
        end
    end
```

Figur 4.9: MATLAB-snippet: SaveResult funktion

4.10 Callbacks funktioner

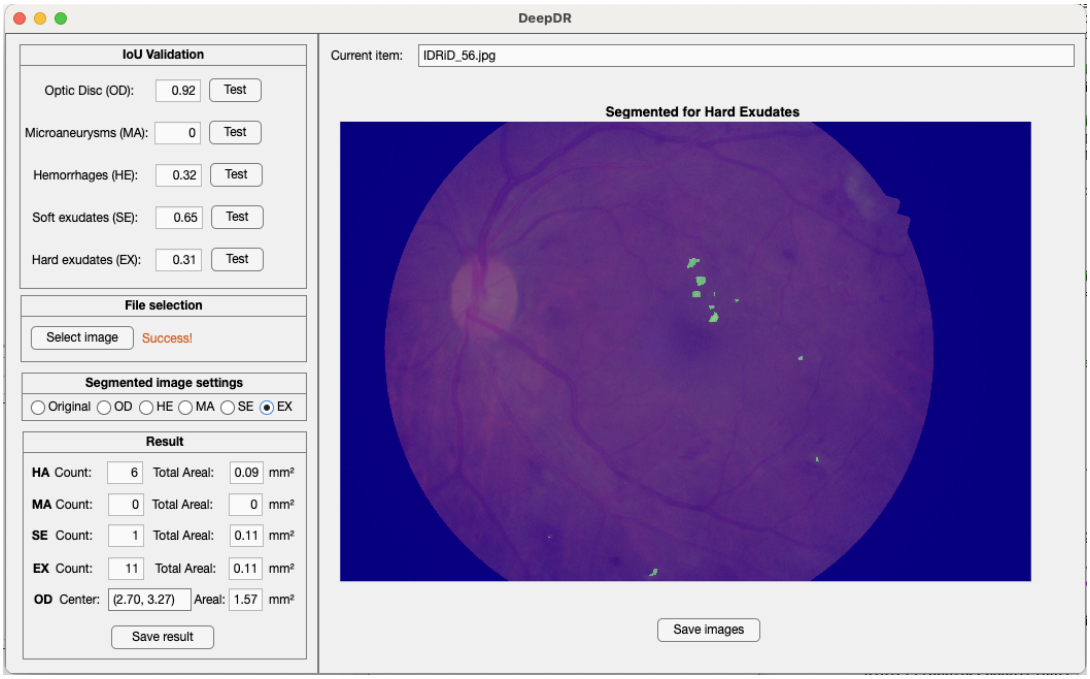
I figur 4.10 ses de forskellige callbacks funktioner for de forskellige knapper i GUI'en, hvor man kan se hvilke funktioner der bliver kaldt, når en knap bliver trykket på.

```
    % Callbacks that handle component events
    methods (Access = private)
        % Code that executes after component creation
        % Button pushed function: TestODButton
        function TestODButtonPushed(app, event)
            validationIoU(app, 'OD');
        end

        % Button pushed function: TestSEButton
        function TestSEButtonPushed(app, event)
            validationIoU(app, 'SE');
        end

        % Button pushed function: SelectimageButton
        function SelectimageButtonPushed(app, event)
            % Deaktiver UI elements
            app.OriginalButton.Enable = "off";
            app.ODButton.Enable = "off";
            app.MAButton.Enable = "off";
            app.HEButton.Enable = "off";
            app.SEButton.Enable = "off";
            app.EXButton.Enable = "off";
            app.SaveresultButton.Enable = "off";
            app.SaveimagesButton.Enable = "off";

            % Nulstiller UI labels
            app.SaveResultStatus.Text = "";
            app.SaveImagesStatus.Text = "";
            app.SelectImageStatusLabel.Text = "";

            openFile(app);
            segmentImage(app);
            analyzeResult(app);

            % Enable UI elements
            app.OriginalButton.Enable = "on";
            app.ODButton.Enable = "on";
            app.MAButton.Enable = "on";
            app.HEButton.Enable = "on";
            app.SEButton.Enable = "on";
            app.EXButton.Enable = "on";
            app.SaveresultButton.Enable = "on";
            app.SaveimagesButton.Enable = "on";
        end

        % Selection changed function: SegmentedimagesettingsButtonGroup
        function SegmentedimagesettingsButtonGroupSelectionChanged(app, event)
            selectedImage(app);
        end

        % Button pushed function: TestHEButton
        function TestHEButtonPushed(app, event)
            validationIoU(app, 'HE');
        end

        % Button pushed function: TestMAButton
        function TestMAButtonPushed(app, event)
            validationIoU(app, 'MA');
        end

        % Button pushed function: TestEXButton
        function TestEXButtonPushed(app, event)
            validationIoU(app, 'EX');
        end

        % Button pushed function: SaveresultButton
        function SaveresultButtonPushed(app, event)
             saveResult(app);
        end

        % Button pushed function: SaveimagesButton
        function SaveimagesButtonPushed(app, event)
            saveSegmentedImages(app);
        end
    end
```

Figur 4.10: MATLAB-snippet: Callbacks funktioner

5 Resultater

Resultaterne af segmenteringen for filen "IDRID_56.jpg" vises i figur 5.1. Ud fra figuren ses det, at programmet 'DeepDR' får lavet en analyse af et nethinde-billede, hvor den får beregnet IoU, antallet og samlet areal for hver læsion og den opstiske disk samt viser et billede af segmenteringen for hårde eksudater.



Figur 5.1: Segmentering af billedet "IDRID_56.jpg".

IoU validation

IoU er en måleenhed til at evaluere nøjagtigheden af segmenteringsalgoritmer. En højere IoU-værdier indikerer en bedre nøjagtighed. Her er IoU-værdierne for hver læsion og den optiske disk ud fra figur 5.1:

- Optic Disc (OD): 0.92
- Microaneurysms (MA): 0
- Hemorrhages (HE): 0.32
- Soft exudates (SE): 0.65
- Hard exudates (EX): 0.31

Disse værdier angiver, at programmet har en høj præcision for segmentering af den optiske

disk, en god præcision for SE og mindre præcis for læsionerne HE og EX. Dog kan den ikke identificere for MA, da IoU er 0, hvilket indikere at modellen ikke er trænet godt nok til at genkende disse små læsioner. Der mangler derfor stadig at få finjusteret modellen.

Resultater

For resultaterne beregner programmet antallet for den enkelte type læsion samt den samlede størrelse, og for den optiske disk beregner programmet centrum i form af et koordinat og arealet. Her er resultaterne for hver læsion og den optiske disk ud fra figur 5.1:

- OD: Centrum i (2.70, 3.27) med et areal på 1,57 mm^2
- MA: Antal på 0 med et samlet areal på 0 mm^2
- HE: Antal på 6 med et samlet areal på 0.09 mm^2
- SE: Antal på 1 med et samlet areal på 0.11 mm^2
- EX: Antal på 11 med et samlet areal på 0.11 mm^2

Der er fra opgaven ikke angivet, hvad pixels er i mm, så derfor er en pixelværdi på 300 pixel per mm valgt. Denne værdi, ville kunne tilpasses til den rigtige.

For OD er værdierne præcise, da den har en IoU på 0.92, som indikerer at den segmenterer den optiske disk præcist. For SE fik modellen identificeret 1 blød eksudat, hvilket må være relativ præcist, da den har en IoU på 0,65. For HE fik modellen identificeret seks blødninger, men da den har en nøjagtighed på 32% ud fra dens IoU kan disse betyde falske positiver eller falske negativer. Det samme gælder for EX. Selvom modellen har identificeret 11 hårde eksudater, er nøjagtigheden af disse kun omkring 31%, hvilket betyder at der kan være falske positiver eller falske negativer. For MA får modellen ikke identificeret nogen mikroaneurismer, hvilket er forventeligt, da den har en IoU på 0 og kan derfor ikke identificere nogen.

Segmenteret billede

Ud fra det segmenteret billede af hårde eksudater i figur 5.1 vises et snit af nethinden med hårde eksudater, som er markeret med grønt.

Ud fra resultaterne kan det konkluderes, at segmenteringsmodellen har fungeret bedst

for OD og SE, men mindre godt for de resterende læsioner. Dette betyder, at yderligere forbedring af modellen kræves, i form af en øget mængde, variation af træningsdata samt finjustering af modellens parametre, for at sikre en bedre generalisering og præcision.

6 Konklusion

Der er blevet præsenteret udviklingen af et brugervenligt GUI i Matlab og segmenteringsalgoritmer til detektion af læsioner ved diabetisk retinopati. Denne opgave har til formål at skabe en pålideligt og automatiseret værktøj til sundhedspersonale for at forbedre diagnosticering og reducere risikoen for synshadicaps hos diabetespatienter.

Resultaterne i denne opgave viste en høj præcision for segmentering af den optiske disk med en IoU på 0,92 og en god præcision for de bløde eksudater med en IoU på 0,65. Dog var der en lavere præcision for hemorrhagier og hårde eksudater med en IoU på 0,32 og 0,31 samt ingen detektion af mikroaneurismer med en IoU på 0.

For at forbedre segmenteringsmodellens resultater ville et fremtidigt arbejde være at fokusere på at øge træningsdata, for at give en bedre generalisering og præcision, samt finjusterer modelparametrene for at give modellen en bedre ydeevne.

Samlet set har opgaven demonstreret potentialet for at udvikle en effektiv segmenteringsalgoritme for diabetisk retinopati, men at det stadig kræver yderligere arbejde for at forbedre nøjagtigheden og pålideligheden for alle typer læsioner.

Reference Liste

[1] Natasha Shaukat. "Three-Dimensional Semantic Segmentation of Diabetic Retinopathy Lesions and Grading Using Transfer Learning". I: *Journal of Personalized Medicine* (sep. 2022). URL: https://www.mdpi.com/2075-4426/12/9/1454.

[2] MATLAB. "Create DeepLab v3+ convolutional neural network for semantic image segmentation". I: (). URL: https://se.mathworks.com/help/vision/ref/deeplabv3pluslayers.html.

[3] MATLAB. "Jaccard similarity coefficient for image segmentation". I: (). URL: https://se.mathworks.com/help/images/ref/jaccard.html.