A machine learning approach to improve analysis of mammograms

Anders Lindhardt Madsen 177609825

Bachelor Project in Software Engineering June 2023

With guidance from Uffe Kock Wiil, Professor, Ph.D.

The Maersk Mc-Kinney Moeller Institute
University of Southern Denmark

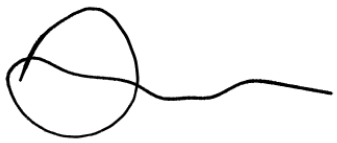
Titlepage

University of Southern Denmark

Faculty of Engineering

Maersk Mc-Kinney Moller Institute

Project period: 01. February 2023 - 01. June 2023



Anders Lindhardt Madsen
andem20@student.sdu.dk

Abstract

This project demonstrates how a software based mammogram analysis tool is build using machine learning. It proposes a pipeline consisting of four steps: preprocessing, region of interest detection, mass segmentation and pathology classification.
All relevant regions are extracted from the mammogram using the RetinaNet architecture which implements an object detection mechanism. The mass of each regions is segmented using a U-Net architecture with a parallel transformer block and lastly the pathology of the segmented region is classified using an ensemble network composed of three models taking individual inputs: shape, margin and the original region image.
The region detection model achieves an AP of 0.26, the segmentation model reaches an IoU of 81.12% and an AUC of 0.95 and the pathology classification model reaches an accuracy of 75.93%.
The full pipeline manages to correctly classify and detect 44% of all cases in the test set with a region confidence threshold of 0.3 which can be increase by lowering the threshold at cost of more false posittives.

Contents

1 Introduction

Each year 45,000 people are diagnosed with cancer where 4,800 cases are breast cancer (Gawron 2022). It is estimated that 1 out of 3 at some point will die due to the disease, despite of the high chance of being cured if the cancer is discovered in time (Knop 2021).

The breast is composed of three types of tissue:

- **Glandular tissue** - lobules produces milk and carries it to the nipple via channels called ducts.
- **Fibrous tissue** - connects different breast tissue and holds it in place.
- **Fatty tissue** - fills space between fibrous tissue and gives the breast its shape. (Disease Control et al. 2022)

Breast cancer primarily originates in the ducts (80% of the cases), with the lobules accounting for 10-15% of the cases and the rest being spread on other tissue (Knop 2021). These abnormalities are grouped into two categories: **Masses**, which is an abnormal area of tissue which shape and surrounding tissue stands out, and **Calcifications** which are small calcium deposits (Society 2022). Both abnormality types can be either malignant (cancerous) or benign (non-cancerous).

Women in Denmark aged 50-69 are offered a frequent breast screening each second year to increase the chances of discovering the cancer earlier resulting in earlier treatment and better chances of being cured (Bigaard et al. 2023b).
The examination is performed by capturing an x-ray of each breast to be analyzed by two radiologists who looks for potential malignant lumps in the breast. If any suspicious lumps are found, they will be further examined by performing a biopsy for the cells to be analyzed (ibid.).

Though having increased chances of detecting breast cancer in an early stage due to the frequent screenings, the amount of women affected by errors will increase proportionally with the amount of screenings in the form of misdiagnoses.
21.7% of misdiagnosis in Denmark are related to cancer where 27.3% of all cancer misdiagnoses being breast cancer. This means almost 6% of the total misdiagnoses are related to breast cancer (Patientsikkerhed 2019).
Misdiagnosis can either be in the form of a false positive or a false negative. A false positive is a case where a woman is diagnosed with breast cancer despite in reality not having cancer. This error can happen in the mammogram analysis-phase where a benign tumor is classified as malignant. This can result in an overtreatment which wastes resources and can be stressful for the patient (Bigaard et al. 2023a). 2.5% of women attending screenings gets a callback for further examination, where 74% of the callbacks gets the suspicion disproved and are diagnosed as healthy (ibid.).

A false negative is a case where a woman is diagnosed as healthy though in reality having a malignant tumor. This error can be caused by a radiologist wrongly classifying a malignant tumor as benign or the tumor being very small making it hard to detect. These errors can have fatal consequences as the treatment of the malignant tumors will be delayed decreasing the chances of being cured (Bigaard et al. 2023a).

From 2015 to 2017 a chief physician examined 3,600 but due to suspicion about errors in the examination procedure, all women were offered a secondary examination. 2,162 women accepted the offer where 21 were diagnosed with breast cancer despite initially being diagnosed as healthy by the chief physician - false negatives (Teglgård et al. 2017). In this case over half of the already examined women gets a second examination which cost a lot of resources and even worse some women turns out to be fatally misdiagnosed as false negatives. These mistakes could maybe have been avoided if additional tooling was used.

Artificial intelligence (AI) is beginning to be incorporated in multiple parts of the danish healthcare system and it has a big potential in assisting and solving complex tasks (Lindskow et al. 2020).
This project aims to develop a machine learning (ML) based assistant to be used by radiologists when analyzing mammograms. The final product will consist of a UI for the radiologist to upload a mammogram to be analyzed using ML, where the analyzed mammogram is returned showing the regions of interests (ROI) with a confidence score.

1.1 Problem statement

> How can machine learning be applied as assistance when analyzing mammograms, to improve the diagnosis quality and reducing resources spent on mammography screenings?

2 Project boundaries

To better be able to compare the results of the proposed model, only the mass abnormality cases are considered and therefore all calcification cases are left out.
The project should be seen as a demonstration of how machine learning techniques can be applied in mammogram analysis rather than a final production ready product.

2.1 Prerequisites

It is assumed that the reader has basic knowledge of the basic mechanics of machine learning as well as common techniques such as convolutional neural networks (CNN) and fully connected networks as well as an understanding of how loss functions and activation functions work.

3 Related work

Multiple articles have been published concerning mammogram analysis with machine learning.

3.1 An integrated framework for breast mass classification

Baccouche et al. have created a full framework for breast cancer detection. The proposed framework is composed of three parts (Baccouche, Garcia-Zapirain, and Elmaghraby 2022):

1. Region of interest (ROI) detection
2. ROI segmentation
3. ROI classification

The ROIs are detected using an object detection technique with the YOLO architecture (Redmon et al. 2015).
The detected regions are then further processed by a segmentation network creating an outline of the mass tumor (Baccouche, Garcia-Zapirain, Olea, et al. 2021). For this task a connected U-Net composed of two connected U-Nets in sequence is used yielding a result of 80.02% IoU and 0.79 AUC on the CBIS-DDSM dataset (ibid.).

To classify the pathology of each ROI, the original ROI cutout is masked with the segmentation. The pathology is a binary classification as either Benign or Malignant where an accuracy of 95.13% and an AUC of 0.95 is achieved.
Furthermore the BI-RADS and shape of the tumor is classified. The BI-RADS is a multi-class classification of 5 classes achieving a result of 83.84% accuracy and 0.94 AUC. The shape classification is as well a multi-class classification consisting of the top four present shape categories (Round, Ovel, Lobulated, Irregular) which achieves an accuracy of 90.02% and 0.98 AUC (Baccouche, Garcia-Zapirain, and Elmaghraby 2022).
The ROI classification is performed by a large model composed of three submodels each using the ResNet architecture: (1) ResNet50V2, (2) ResNet101V2, (3) ResNet152V2. Each submodel uses transferlearning by having preloaded the *imagenet* weights.

3.2 Weakly-supervised High-resolution Segmentation

To solve the problem of sparse amounts of available data within the medical domain, a weakly-supervised learning (WSL) approach is proposed for detecting regions of interests in a full size mammogram only by a single label (benign or malignant) (Liu et al. 2021). This is achieved by introducing Global-local Activation Maps (GLAM), where the original mammogram is fed into a global network outputting multiple patches by cropping the

most prominent features in the coarse saliency map. A segmentation is then created for each patch by a more fine-grained saliency map resulting in a dice score of 39% and 0.82 AUC on the NYU Breast Cancer Screening Dataset v1.0 dataset.

3.3 Multi-scale adversarial networks for segmentation

Adversarial network with three discriminators and one generator. The generator is a network with the U-Net architecture which creates a segmentation of the input image. The result of the generator is fed to the three discriminator networks each receiving the segmentation image in different sizes: original, half and quarter respectively. This way each discriminator has a different receptive field, making them able to catching both details and more global context.
The discriminators are trained to differentiate between ground truth and synthetic images, resulting in a well performing segmentation generator, when the discriminators cannot differentiate between synthetic and ground truth(Juan Chen et al. 2020).
This approach yields a dice score of 82.16% and 0.99 AUC on the CBIS-DDSM dataset.

3.4 AUNet for breast mass segmentation

To address the issue of information loss when upsamling in the decoder of the U-Net, the AUNet architecture is introduced. The AUNet is composed of an asymmetric encoder and decoder by introducing attention-guided upsampling block (AU block) in the decoder where both high and low level features are combined. Firstly the high level low-resolution features are upsampled and then combined with the low-level higher-resolution features. The proposed model yields a dice score of 81.4% on the CBIS-DDSM dataset (Sun et al. 2018).

3.5 YOLO-LOGO transformer-based image segmentation

Utilizes the YoloV5L6 architecture which is a version of the original YOLO architecture (Redmon et al. 2015) for fast object detection. This approach detect all regions of interests in a full mammogram. This is combined with a transformer based architecture known as medical transformer MedT (Valanarasu et al. 2021) to create a segmentaiton of the ROI. The MedT has a self-attention mechanism enabling the ability to catch global information (Su et al. 2022).
The proposed architecture yields an IoU of 64.04% on the CBIS-DDSM dataset.

3.6 TransUNet

The TransUNet tries to address the issue of lacking global context awareness in regular U-Nets. It extends the original UNet architecture with a hybrid of CNN and transformers, having a transformer layer in the last layer of the encoder (Jieneng Chen et al. 2021). The proposed architecture achieved an average dice score of 77.48% on the Synapse multi-organ segmentation dataset, which is a multi class dataset.

4 Method

The project utilizes an agile methodology where design, analysis, experiments/prototyping and implementation are done iteratively and incrementally.
First the dataset is analyzed for which a coarse design is constructed. Hereafter prototypes are created and tested to confirm or disprove the proposed design. Experiments are then performed with the prototypes to increase the performance. Each experiment is documented for comparison and to avoid performing the same experiments twice.

The articles in section 3 have been gathered by searching the web for articles including the CBIS-DDSM dataset or machine learning for medical image analysis. The articles have been selected based on their relevance where the inclusion criteria was whether the approach could be used as inspiration or the results could be used for comparison.

5 Analysis

To assist and support the radiologist best possible, the system will try to imitate how the radiologist analyzes mammograms. This means that the system should be able to identify masses in a mammogram and classify them as either malignant or benign.

The typical size of a mammogram is approximately 12 megapixels or 4000x3000 pixels which is relatively large with a pixel size of $70\mu m$ (Huda et al. 2015). Tumors ranges in size up to $5cm$ making them relatively small compared to the size of the breast (Clinic 2023). A $2cm$ tumor correspond to about 260x260 pixels making up around 0.5% of the entire mammogram. This means that the system must be able to detect rather small objects in a large image. To accommodate this, the system should approach each mammogram by first getting an overview of its contents resulting in multiple detected regions of interest (ROI) and then further investigate and analyze each ROI.

To better visualize the reasoning about the analyzed mammogram, the final analyzed result should be showed as an overlay on top of the original mammogram together with a confidence score of each analyzed ROI. This will enable the radiologist to further analyze the result based on the detected regions and the provided confidence scores.

The analysis of the mammogram will be performed by supervised machine learning requiring an annotated dataset containing the original mammograms with masks of the tumors and a pathology annotation of each tumor. This should make it possible for the system to learn patterns corresponding to malignant or benign tumors and that way acquire the same "knowledge" as a trained radiologist may possess.

5.1 Descriptive data analysis

The following describes the chosen dataset containing annotations. The dataset is chosen due to it being the standard for benchmarking machine learning models in this particular field. Though other similar datasets such as MIAS (SOCIETY 2017) and INbreast (Moreira et al. 2012) exists, it should be noted that it could be necessary to expand the chosen dataset with these.

5.1.1 CBIS-DDSM dataset

The CBIS-DDSM (*Curated Breast Imaging Subset of Digital Database for Screening Mammography*) dataset contains 3568 grayscale mammogram images divided between 1566 patients (Sawyer-Lee et al. 2016). The images in the dataset comes as DICOM (.dcm) files which is a lossless 16 bit grayscale format and an international standard for medical images (*About DICOM: Overview* 2023).

Figure 5.1 depicts a sample from the dataset showing from the left, the original mammogram, a mask for each abnormality found and on the far right an example of how the masks fit the original image.

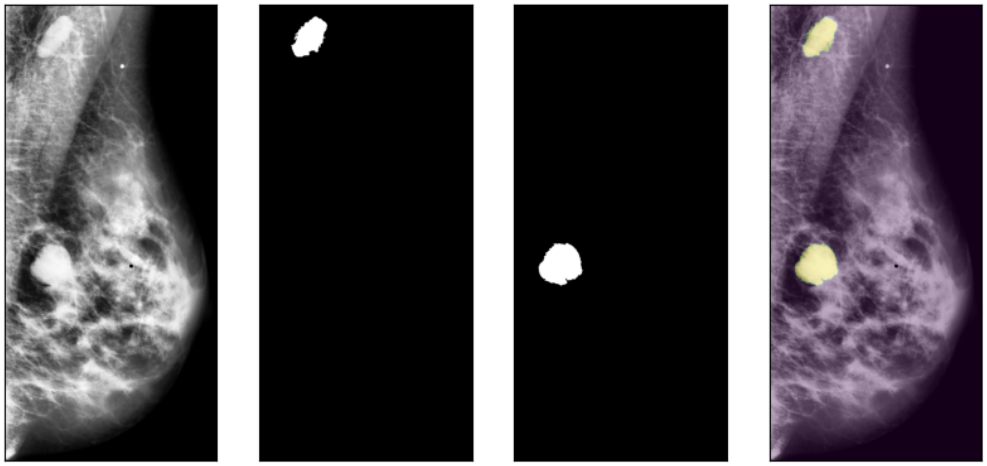


Figure 5.1: Examples of mammogram and masks

The resolution ranges from 3000x4000 to 4000x5000 pixels making them relatively large. Figure 5.2 depicts the distribution of the mass sizes. Here there is a notable cluster in the lower range dimensions with a width and height of 150 to 500 pixels.

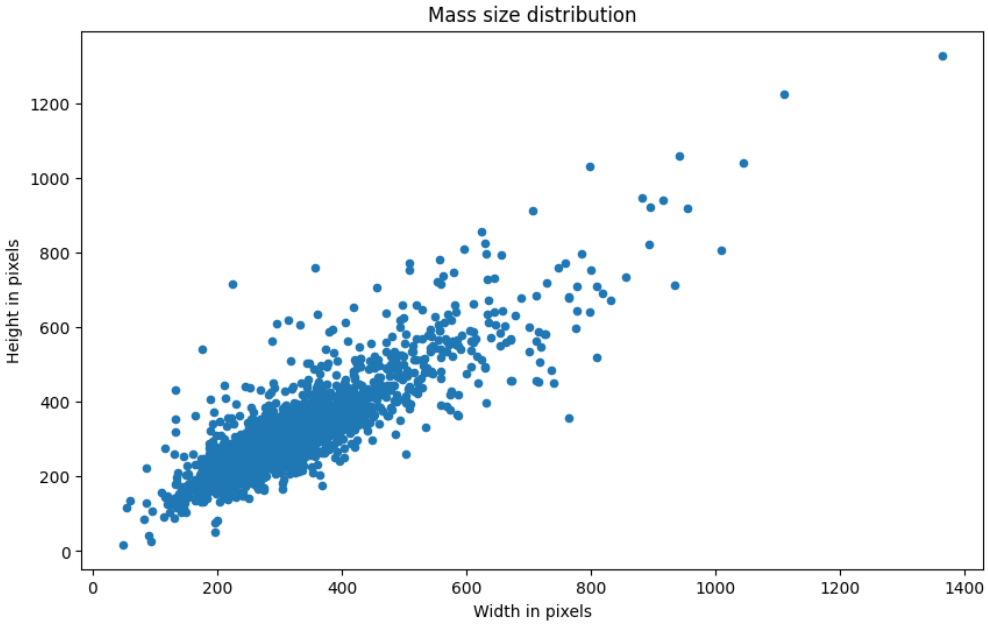


Figure 5.2: Mass size distribution

Each image is annotated with several labels:

Patient id

Breast density

1. The breasts are almost entirely fatty (about 10% of women).
2. A few areas of dense tissue are scattered through the breasts (about 40% of women).
3. The breasts are evenly dense throughout (about 40% of women).
4. The breasts are extremely dense (about 10% of women).
   (Cancer Prevention et al. 2022)

Left or right - Specifies left or right breast

Image view - CC (Craniocaudal view) or MLO (Mediolateral oblique view)

Abnormality id - Id of specific segmentation mask for the same patient

Abnormality type

- Calcification (Depots of calcium)
- Mass (Area of abnormal breast tissue)

Assessment - BI-RADS assessment. Actions to be taken based on the analysis.

0. Incomplete - Additional imaging is needed
1. Negative - Symmetrical and no masses
2. Benign - 0% probability of malignancy
3. Probably benign - <2% probability of malignancy
4. Suspicious for malignancy - 2-94% probability of malignancy
5. Highly suggestive of malignancy - >95% probability of malignancy
   (Niknejad 2022)

Pathology

- Benign (Noninvasive noncancerous tumors)
- Benign without callback (A benign tumor worth tracking)
- Malignant (Invasive cancerous tumors)

Subtlety - Radiologists' rating of difficulty in viewing the abnormality in the image

Mass shape and margin - Shape of the mass and the surrounding tissue

Calcification type and distribution

For each abnormality a segmentation mask image is provided. Each mammography can have multiple abnormalities of different type (*mass* and *calcification*) and therefore multiple segmentation mask images can be present for a single mammography.
The dataset was originally split by the abnormality types (*Mass* or *Calcification*) and then split into a training (80%) and test set (20%) based on the BI-RADS category resulting in the datasets being somewhat equally balanced (Sawyer-Lee et al. 2016).

Due to the boundaries of this project, only the mass cases are considered and therefore only those are referenced when referring to *the dataset*. This results in a training and test set with a mass abnormality consisting of 1318 and 378 samples respectively.
Before training a model the training set is split into a training (80%) and validation set (20%) where the validation set is used for hyperparameter tuning where the test set is used for the final evaluation of the models.

5.1.1.1 Pathology

Three pathological types are present in the dataset: **Benign** - a noninvasive noncancerous lump, **Benign without callback** (BWOC) - a noncancerous lump but which is more suspecious than a completely benign case and therefore worth tracking, **Malignant** - an invasive cancerous tumor for which actions should be taken.

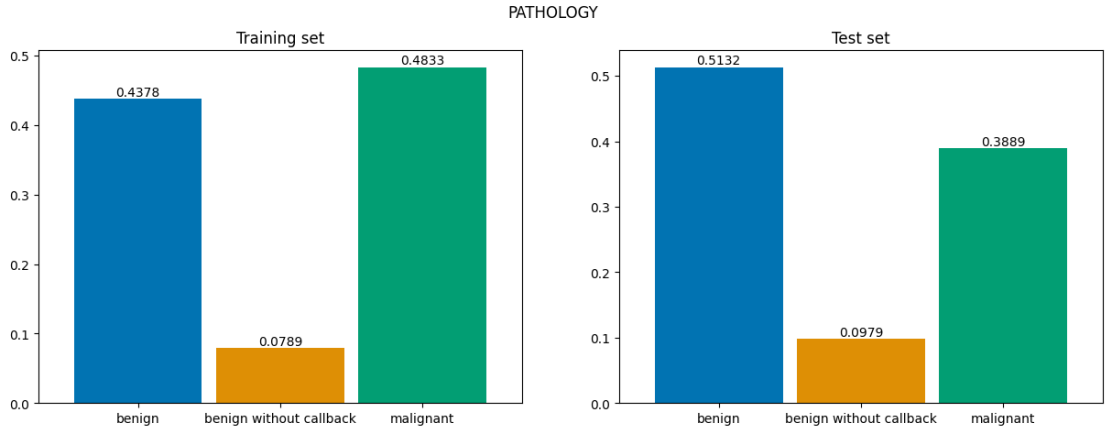


Figure 5.3: Distribution of pathology on training and test set

The distribution of the pathologies are clearly not balanced in the datasets with BWOC only making up 8-9% of in both the training and test dataset. This can lead to the classifier overfitting to the most represented classes and therefore not performing well when having to classify the underrepresented class. Though there are methods for handling this case of class imbalance (with an oversampling technique or class weights), the BWOC class is merged with the Bening class as these are closely related being both noncancerous. This results in all the BWOC cases being labelled as Benign cases which makes the training set balanced but still leaves the test set imbalanced as depicted in figure 5.4.

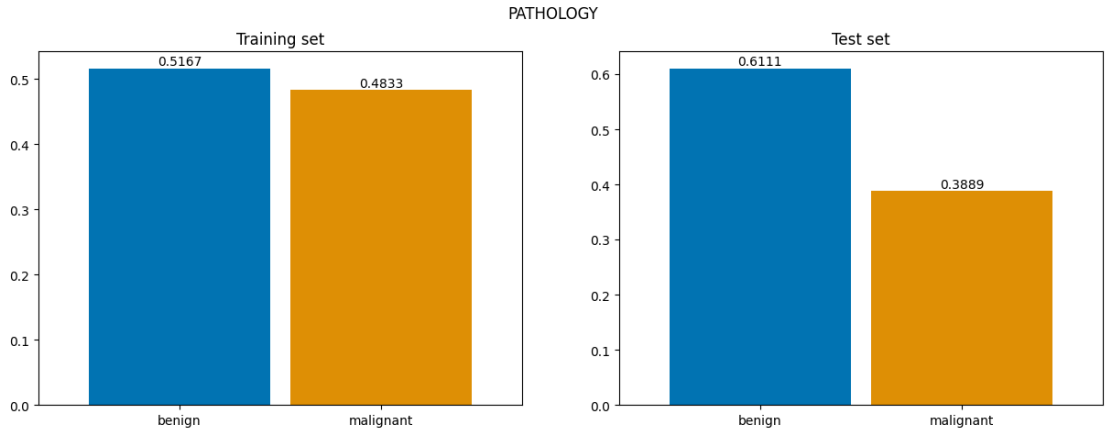


Figure 5.4: Distribution of pathology with merging benign and benign without callback

The imbalance of the test set is no issue as it will not be used in training but only for

evaluation. Though it can be valuable to know how the test set is balanced when evaluating the trained model to better interpret the results.

5.1.1.2 Assessment

The Assessment label defines the Breast Imaging-Reporting and Data System (BI-RADS) rank of the particular sample. The rank ranges from 0-6 and describes the severity of the finding introducing a standardized way for radiologists to report their findings (Niknejad 2022). The dataset contains assessments with rank 0-5 with a major class imbalance with rank 1 being almost non-existing, which could be due to rank 1 describing a negative case not containing any mass.

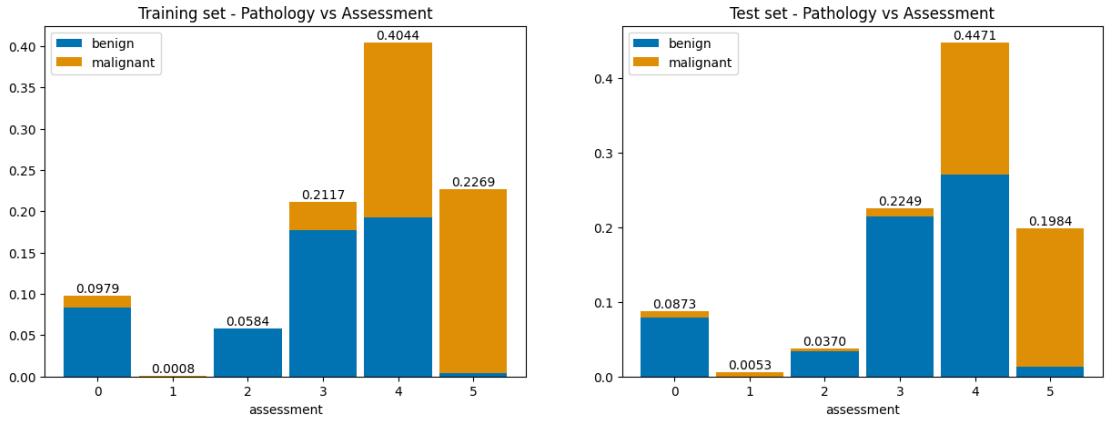


Figure 5.5: Distribution of assessment and pathology on training and test set

The stacked bar chart on figure 5.5 depicts the assessment distribution and the pathology distribution for each assessment rank. The chart clearly shows the relation between pathology and assessment where rank 5 almost exclusively contains malignant cases, rank 4 having almost 50% of both benign and malignant cases. This confirms that a correlation between the assessment rank and pathology is present which means that the assessment rank should be used when prediction the pathology of a particular region of interest.

The distribution of the assessments are imbalanced which can lead to overfitting to cases with assessment rank 4. This should be kept in mind when training a model for predicting the assessment, where oversampling the minority classes or using class weights could be a beneficial tool to compensate for the imbalance.

5.1.1.3 Breast Density

The breast density rank describes the breast tissue composition. Though a higher breast density should have an impact on the probability of getting cancer (Cancer Prevention et al. 2022), it is not reflected in the dataset on figure 5.6, where each breast density rank has almost the same pathological distribution about 50% for both malignant and benign cases.

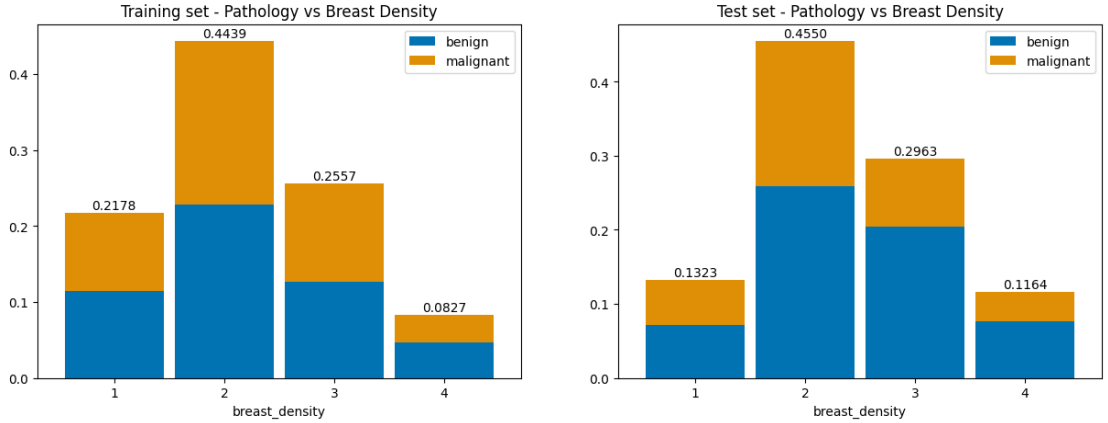


Figure 5.6: Distribution of breast density and pathology on training and test set

This means that for the particular task of classifying the pathology of a given mass, the breast density label will probably not provide any meaningful patterns for the model to recognize.

5.1.1.4 Mass shape and margin

The shape of the mass is described by 8 atomic shapes: (1) Irregular, (2) Oval, (3) Lobulated, (4) Round, (5) Architectural distortion, (6) Lymph node, (7) Focal asymmetric density, (8) asymmetric breast tissue. These are also combined in groups up to three such as *Round-Irregular-Architectural distortion*, though the most common cases are shapes with an atomic description.
Figure 5.7 depicts the top 5 most present shape types in relation to the pathology.

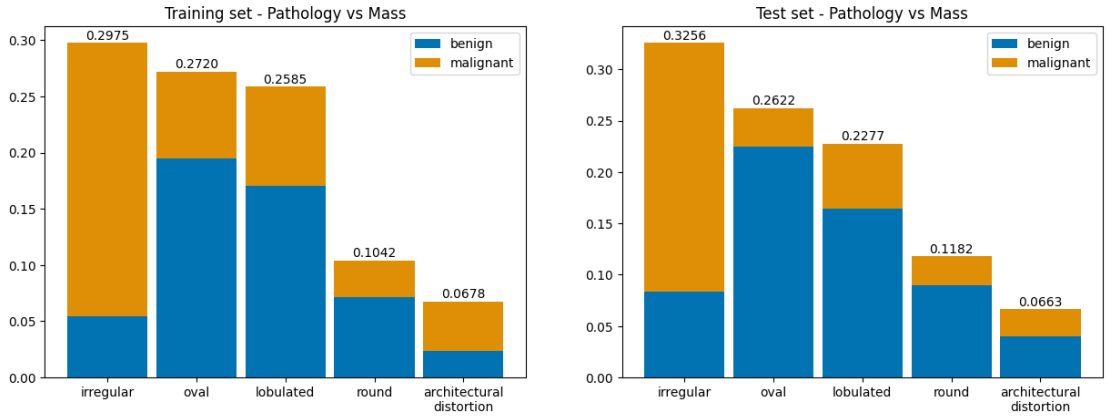


Figure 5.7: Distribution of shape and pathology on training and test set

The figure clearly shows that the *irregular* shape type has a overrepresentation of the malignant pathology type, whereas the other categories consists mostly of benign cases. This means that a model for predicting pathology might benefit from using the shape description, as there might exist some pattern, but it should be noted that only the most present categories should be included due to the heavy class imbalance.

The margin label describes the tissue surrounding the mass. It is composed in the same way as the shape category with five atomic categories: (1)Circumscribed, (2) Spiculated, (3) Ill defined, (4) Obscured, (5) Microlobulated. These categories are also combined in multiple ways in groups up to three.
Figure 6.2 depict the margin distribution in relation to pathology. The *spiculated* category has a strong correlation with the malignant cases and whereas the *circumscribed* category has mostly benign cases.

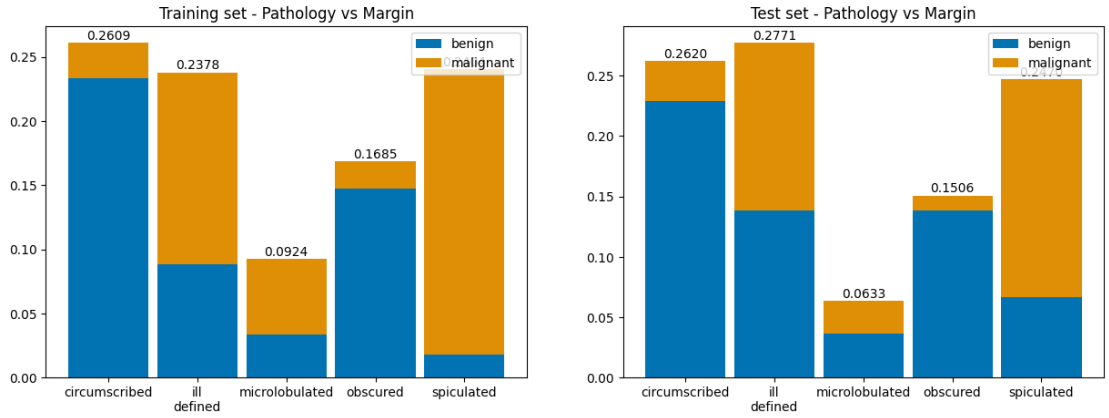


Figure 5.8: Distribution of margin and pathology on training and test set

Though both the shape and margin distributions are heavily imbalanced if all categories are considered, they clearly represents some patterns regarding the pathology and should both be considered when classifying the pathology.

5.2 Evaluation metrics

To ensure that the models are reliable and as robust as possible, multiple evaluation metrics are used depending on the particular task for the model to accomplish.

5.2.1 Accuracy metric

The accuracy metric is one of the most common used metrics and measures the ratio of correct predictions out of all actual cases:

$$Accuracy = \frac{\text{Correct predictions}}{\text{Total cases}} = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP = True positives, TN = True negatives, FP = False positives, FN = False negatives.
This metric works well when the dataset is balanced.

5.2.2 Precision & Recall

The precision and recall metrics are useful when dealing with imbalanced data or wanting to have a more detailed description than the accuracy metric can provide of a given model.

The precision metric measures how often the model is correct when predicting a certain class:

$$Precision = \frac{TP}{TP + FP}$$

The recall metric measures how often the model predicts the correct class when a sample with a given label is evaluated:

$$Recall = \frac{TP}{TP + FN}$$

A high recall score with a low precision indicates that the model correctly classifies most of the classes in the dataset, but is overpredicting for that particular class resulting in many false positives. This is a sign that the model is not able to differentiate between the two classes (ex. benign and malignant).
Inversely, a high precision with a low recall indicates that the model can differentiate between the two classes, but has not learned all the necessary features to generalize across the dataset. This results in many false negatives.

5.2.3 Intersection over union (IoU)

The intersection over union (IoU) metric, also in some cases referred to as the Jaccard index, measures the overlap of two objects (Baeldung 2023). The IoU metric is calculated the following way:

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} = \frac{A \cap B}{A \cup B} =$$

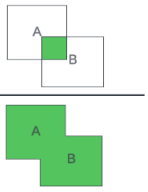


Figure 5.9: Intersection over union

The IoU metric solves the problem of measuring how precise the overlap of the prediction objects are which cannot be achieved by using the regular accuracy metric as this will not treat the potential class imbalance well.
For instance if the pixels of an image is occupied of 97% background (class 1) and 3% of an object (class 2) leading to a class imbalance between class 1 and 2, the model could just predict that all pixels in the image belongs to class 1 and that way achieve an accuracy of 97%. Though using the IoU metric the IoU score would be calculated for each class and then averaged also known as the mean IoU (mIoU). In the particular example, the IoU score of the background (class 1) would be 97% and for the object (class 2) it would be 3%:

$$mIoU = \frac{IoU(\text{class 1}) + IoU(\text{class 2})}{2} = \frac{0.97 + 0.03}{2} = 0.5$$

Intuitively this yields a far more realistic measurement of the performance of the model and can be used as well as a loss function, for the model to be punished for not classifying smaller objects.

It should be noted that another metric called *Dice score* also exists but is proportional to the IoU score:

$$Dice = \frac{2IoU}{IoU + 1}$$

5.2.4 Area under the ROC curve (AUC)

The reciever operating characteristic (ROC) curve describes the relation between the true positive rate (TPR) (the same as recall) and false positive rate (FPR) at various classification thresholds (Developers 2022).

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

The area under the ROC curve (AUC) is a number between 0 and 1, where 1 indicates a perfect model, predicting all cases correctly.

5.2.5 Mean Average Precision (mAP)

The mean average precision metric is a standard metric used when evaluating object detection models. It measures the accuracy of the model in relation to the precision and recall score at various IoU thresholds (Cord Technologies 2023):

$$AP = \sum_{n} (R_n - R_{n-1}) P_n$$

$$mAP = \frac{1}{k} \sum_{i=1}^{k} AP_i$$

The average precision metric calculates a weighted sum of all precision scores at n IoU thresholds, resulting in a single number ranging from 0-1 describing how well the model can predict the position and class of objects.
For instance the Common Object in Context (COCO) consortium, which provides a dataset of 330,000 images containing 80 different classes, uses the mAP with a IoU threshold at 0.5 to 0.95 with steps of 0.05 (COCO 2023).

5.3 Requirements

The following requirements were identified from the analysis and from other related studies described in chapter 3 (related work).

5.3.1 Functional requirements

F1. The system must be able to analyze mammograms.

F1.1. The system must be able to open the image file of the mammogram and return the results of the analyses as an overlay of the original image.

F1.1.1. The outline of the detected mass tumors should be visualized.

F1.2. The radiologist must be able to see the preprocessed mammogram to better understand predictions of the model.

F1.3. The system must be able to handle the DICOM format as it is the standard format of x-ray images.

F1.4. The system must be able to handle various image dimensions.

F1.5. The system must be able to detect small masses in a high resolution mammogram.

5.3.2 Non-functional requirements

NF1. The mammogram analysis must be performed by machine learning models.

NF1.1. The machine learning models must be trained on the CBIS-DDSM dataset to better compare the results with other studies.

NF1.2. The models must have good performance.

NF1.2.1. The overall model should be able to detect >=80% of all mass cases and classify them correctly with a IoU of >=50%.

NF1.2.2. The ROI detection model must have an mAP of >=40% with an IoU threshold of >=50%.

NF1.2.3. The segmentation model must have a mean IoU >=80% and an AUC score >=0.79.

NF1.2.4. The pathology classification model must have an accuracy of >=90% and an AUC of >= 0.9.

NF2. The system should be able to run offline.

NF3. The system should have a modular architecture to increase modifiability and be integrable with other systems.

NF4. The analysis time for a mammogram should be <=30 seconds.

6 Design

The following describes the design of the overall model as well of all the incorporated submodels necessary to achieve a final solution which can detect mass tumors in mammograms, generate a segmentation of each and finally classify them as either malignant or benign.
The main idea of the overall model is for it to first gain a global overview of the content of the entire input image and then attend to the most interesting regions which are further detailed and classified.

6.1 Architecture

The overall model is composed of a pipe and filter architecture due to the sequential process of the data being piped through three filters (submodels) each processing the data for the next filter to consume:

1. Prepare the data for the model.
2. Find region of interests (ROIs).
3. Create a segmentation of each ROI.
4. Classify the pathology of each segmentation.

Furthermore a modular architecture is desired as it improves the maintainability of the overall model by allowing submodels to be swapped out without having to retrain the full model which can be very time consuming and resource demanding.

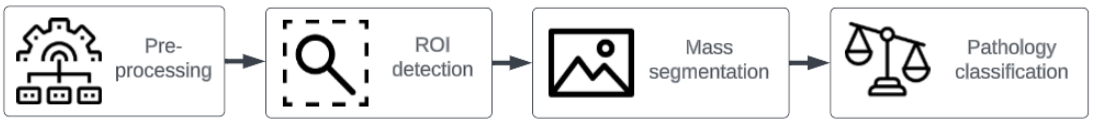


Figure 6.1: Overall model architecture

Figure 6.1 depicts the modular architecture consisting of a preprocessing filter preparing the data and three machine learning models having different responsibilities, fulfilling requirement **NF1.** (*analysis must be performed by machine learning models*).
Each filter can be developed independently enabling parallel development lowering the development time. The independence of each filter facilitates the modular architecture, fulfilling requirement **NF3.** (*the system should have a modular architecture*), and this way introduces the possibility of swapping out a filter or integrate it with another systems as long as the input and output interface is satisfied.

6.2 Preprocessing

To increase the performance of the overall model, the data input is transformed in a preprocessing filter which crops the image and enhances its contrast.

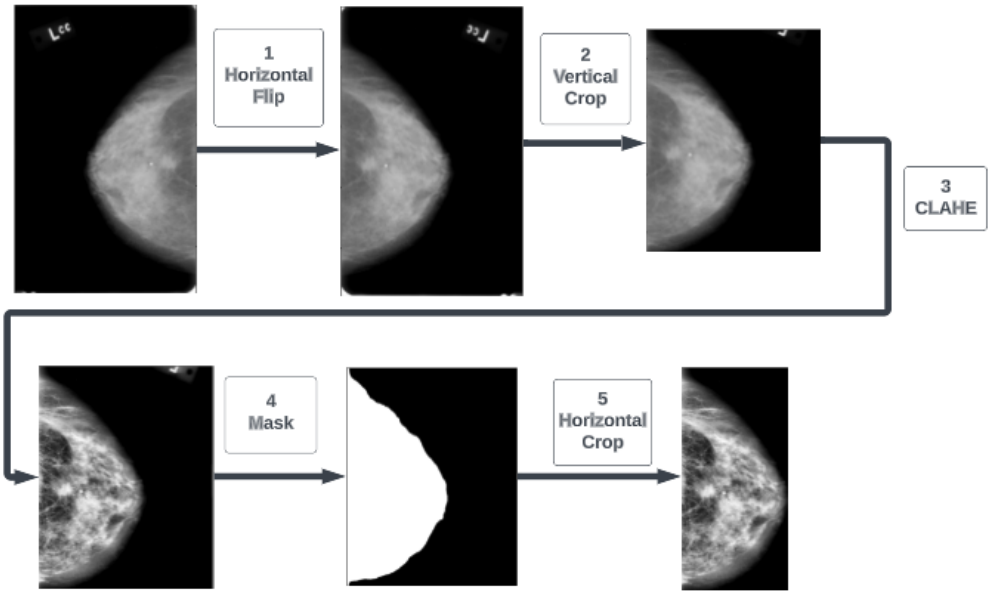


Figure 6.2: Image preprocessing pipeline

Figure 6.2 depicts the image operations of an input image in the preprocessing filter. It is worth noting that in this filter the file format is converted from DICOM to a compressed format such as PNG or JPG which reduces the size multiple times at the cost of some lost detail. This fulfills requirement **F1.3.** (*the system should handle the DICOM format*). The input image is processed by 5 operations:

1. **Horizontal flip** - flips the image conditionally based on its original orientations. This ensures that all images have the same orientation which the mask (4) operation is dependent on.

2. **Vertical crop** - crops the image vertically removing excessive background.

3. **CLAHE** (Contrast Limited Adaptive Histogram Equalization) - enhances the contrast of the image reducing excessive noise, which can improve the training of the subsequent models.

4. **Mask** - removes all other artifacts except of the breast.

5. **Horizontal crop** - crops the image to the edge of the generated mask, reducing the image size.

The sum of all these processing steps should result in an image with a significantly reduced size as most excessive background and noise is removed. This will improve the training time for the ML models as they will not have to waste time analyzing the full image where 50% might contain irrelevant black background pixels.

6.3 Models

The following describes the architecture and design of each of the three ml models.

6.3.1 ROI detection

Due to the high resolution of the mammograms an object detection approach is chosen dividing the large image into smaller parts containing interesting regions to be further analyzed. This fulfills the requirement **F1.5.** (*The system must be able to detect small masses in a high resolution mammogram*)
Figure 6.3 depicts the essential task of the ROI detection model.

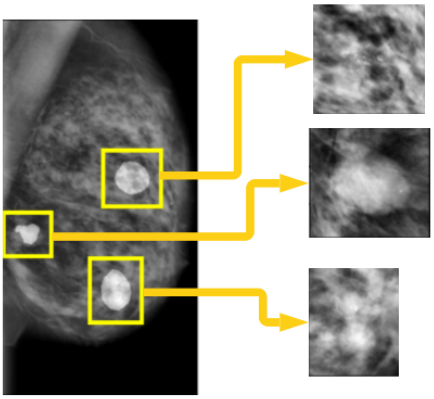


Figure 6.3: ROI detection diagram

The ROI detection uses the RetinaNet architecture which uses a focal loss function to mitigate the class imbalance problem between objects and background (Lin, Goyal, et al. 2018). RetinaNet uses a feature pyramid network (FPN) backbone which constructs a multi-scale feature pyramid where each level can be used to detect objects at different scale. Additionally multiple anchor boxes are generated for which two parallel networks performs class predictions and bounding box regression for each feature map (ibid.).

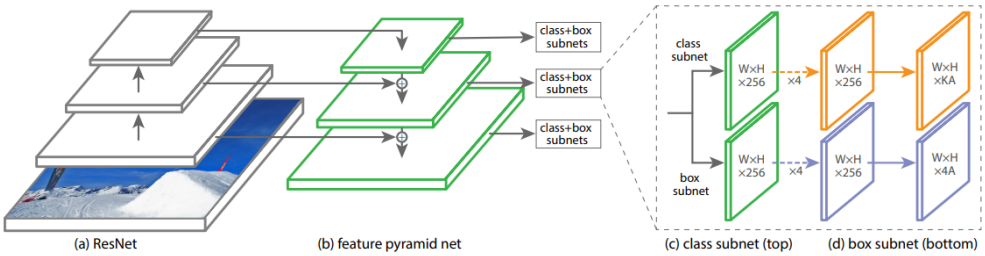


Figure 6.4: RetinaNet architecture (Lin, Goyal, et al. 2018)

The FPN is composed of a convolutional network backbone which encodes the image input and a decoder which upsamples the encoded image and combines them with the original image encodings generating stronger feature maps (Lin, Dollár, et al. 2017).
This approach is independent of image size satisfying requirement **F1.4.** (*The system must be able to handle various image dimensions*).

6.3.2 Mass segmentation

To fulfill requirement **F1.1.1.** (*should be able to visualize the outline of a mass tumor*), a semantic segmentation technique is used. This makes it possible for a network to take an image as an input and then output a mask image of the desired object(s) to be detected. This is done by classifying all pixels in a given image.
For this particular task, only binary classification is needed meaning the model only has to classify whether a pixel belongs to the background or a mass tumor.

The U-Net architecture is chosen as the backbone for this task as it was initially developed for medical images and has good performance for smaller datasets due to its residual connections (Ronneberger et al. 2015).

6.3.2.1 U-Net

The U-Net architecture consists of an encoder and decoder (very similar to the architecture of an FPN). The encoder is a convolutional neural network (CNN) which captures features of the input image from simple to complex shapes. The decoder is responsible for generating the actual segmentations, where the final layer of the encoder is upsampled. This is done in combination with residual connections between the layers in the encoder and decoder, which improves the performance of the model, as the upsample layers has more "knowledge" about the original input.

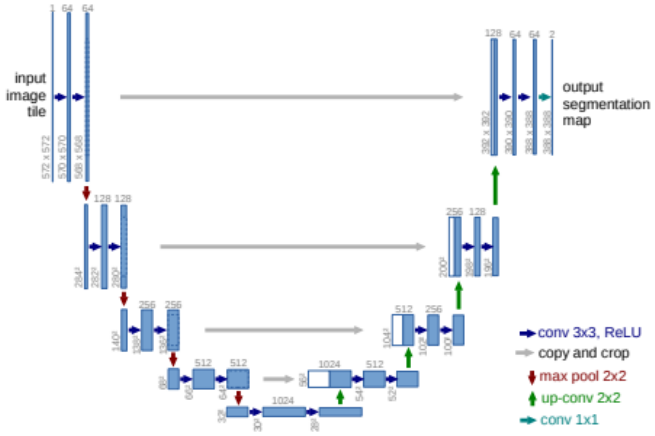


Figure 6.5: U-Net architecture (Ronneberger et al. 2015)

Figure 6.5 depicts the initial U-Net architecture from 2015. Here the encoder has 5 layers and the decoder having 4.
This architecture is modified and combined with a transformer in the final mass segmentation architecture.

6.3.2.2 Transformers

Transformers is a relatively new concept initially created for language models to solve the problem of context awareness (Vaswani et al. 2017). This concept is also adopted in computer vision as vision transformer (ViT) which can benefit from the positional information achieved with transformers.

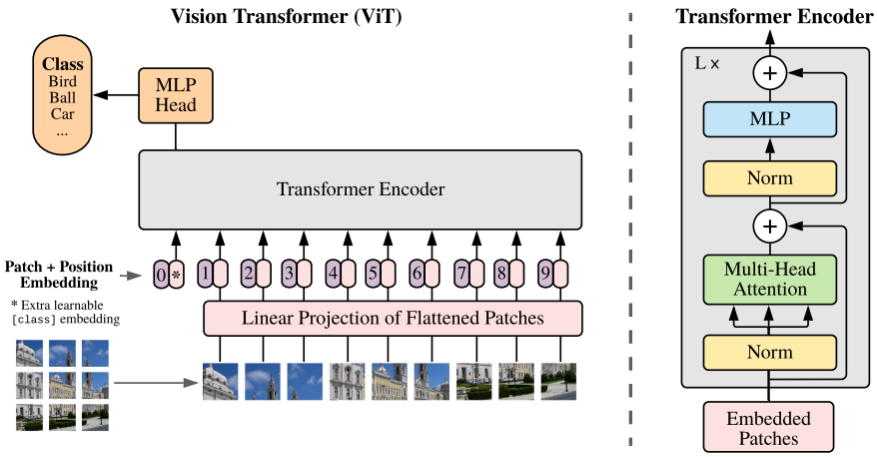


Figure 6.6: Vision Transformer (Dosovitskiy et al. 2020)

Figure 6.6 depict the design of a Vision Transformer. The input image is split up into multiple patches which is parsed to the transformer encoder together with their positional index. The Transformer Encoder outputs an attention matrix which describes the relation between the image patches and can be further used for predictions.
Transformers has been implemented in a U-Net architecture as the TransUNet (Jieneng Chen et al. 2021), and will be the main inspiration for the final architecture.

6.3.2.3 Final mass segmentation architecture

The final architecture for mass segmentation is a U-Net with an attention mechanism in the form of a ViT in the top layer, encoding the original input image.
The architecture is 5 layers deep having residual connections both between the encoder and decoder as well as residual connections between the convolutional layers in each layer of the encoder.
The output of the transformer encoded is added to the deepest layer in the U-Net having a shape of 16x16x512. This block contains all complex features present in the input image where the addition of the transformer should be able to add context awareness making the model more robust but also increase learning speed.

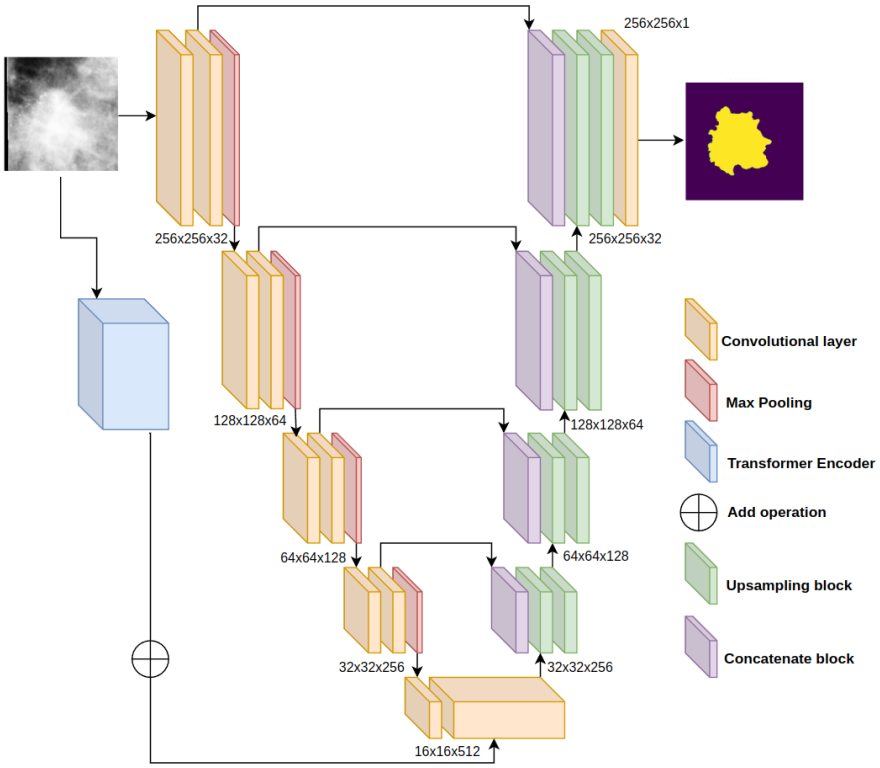


Figure 6.7: Final mass segmentation architecture

Figure 6.7 depicts the final segmentation architecture built with inspiration from the TransUNet having a transformer in last layer of the encoder. The output of this model will be used as input for the last model classifying the pathology of the segmented mass.

6.3.3 Pathology classification

The pathology classification network is composed of four networks as an ensemble network with three networks using the ResNet architecture and the last one being a fully connected network.
The ResNet architecture was developed to solve a general problem of training deep networks due vanishing gradients (He et al. 2015). This is achieved by adding residual connections mapping the input of a node to its output.

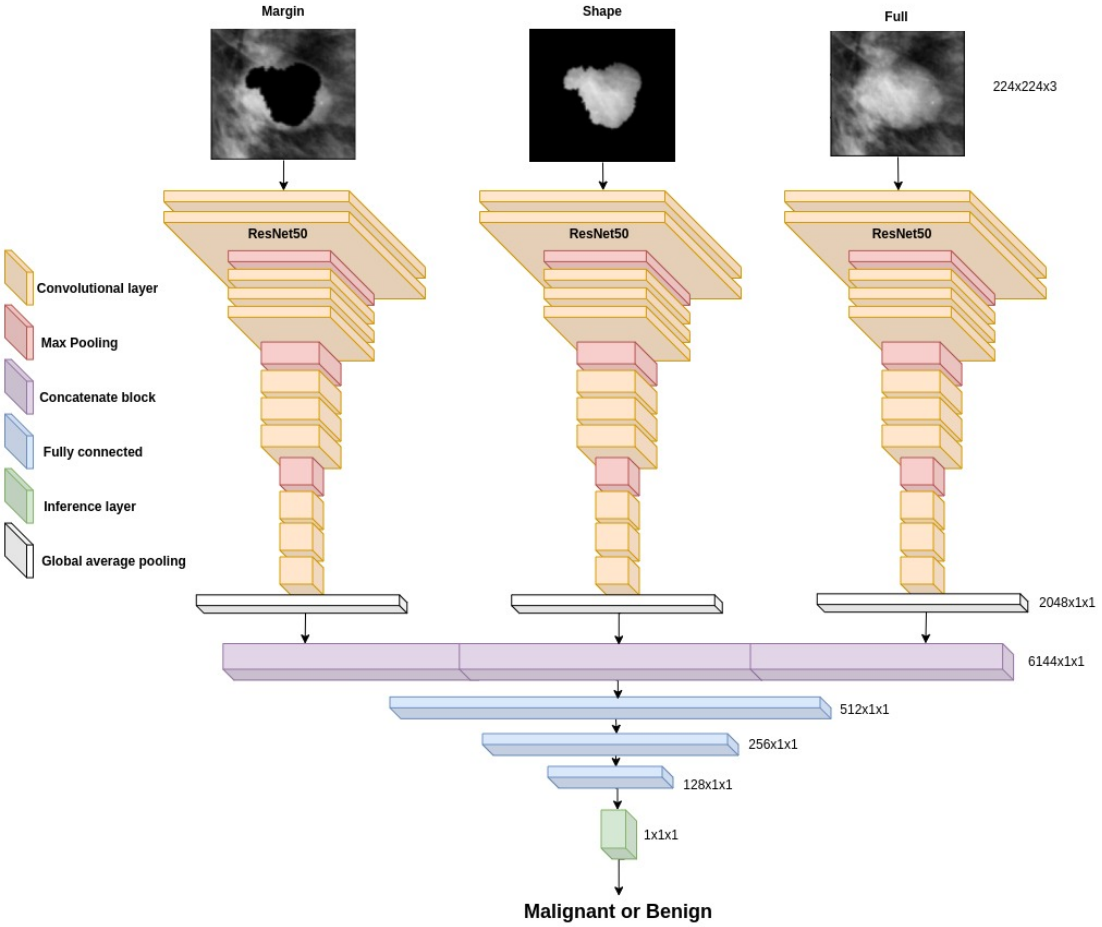


Figure 6.8: Pathology classification architecture

Figure 6.8 depict the architecture of the pathology classification. The model takes three inputs:

1. **Margin** - ROI filtered with the inverse of the segmentation
2. **Shape** - ROI filtered with the segmentation
3. **Full** - Original ROI image

The three inputs are chosen since the shape and margin has an influence on the pathology of a mass tumor as analyzed in section 5 and the full image in added as well to ensure the model knows about the full context. This way the two networks for margin and shape should be "forced" to focus on features related to those aspects resulting in a more confident model.

7 Implementation

The following chapter describes the most important implementation details of the models as well as the different tools used and how the final solution is deployed.
All development is done in Python due the extensive selection of ML libraries and it being very high level making development of complex algorithms faster. This is at the expense of having no strict type checking and more ambiguous error messages, which can, in some cases, slow down development.

7.1 Structural preprocessing

The CBIS-DDSM dataset comes with folder structure having a separate folder for each patient. To get a better overview of all the images, they are restructured to a */train* and */test* folder containing their respective images. To better identify and filter the mammograms, the md5 hash is calculated for each image based on its pixel values. This also makes it easy to get an overview of how many masks belongs to a specific image.
Due to the architecture containing multiple models requiring different *views* of the dataset (object detection trains of full images, whereas pathology classification only needs the ROIs), a separate dataset is created containing the ROI images, to be used when training the image segmentation model and pathology classifier.

7.2 Libraries and tools

Multiple different libraries and tools are used for implementing and training the models as well as preprocessing the images.

7.2.1 Tensorflow and Keras

Tensorflow is a multi-language ML library providing all the core features to be able to build ML models (Martín Abadi et al. 2015). Due to Tensorflow being very lowlevel, a more highlevel library called Keras is provided to enable faster development (Chollet et al. 2015).

7.2.2 Data manipulation and analysis tools

For data manipulation the Pandas library is used (team 2020) as it provides good utilities for querying data as well as a built-in plotting feature built on top of Matplotlib (Hunter 2007) making it easy to plot and analyze the data.
For image manipulation the OpenCV library (OpenCV 2023) is used as it provides a wide variety of image processing features such as contrast manipulation, image blur, binarization and contour detection which is used in the preprocessing of the mammograms.
For more lowlevel datamanipulation, the NumPy library (NumPy 2023) is used which

is fast and flexible. NumPy is furthermore integrated into Tensorflow making it very suitable to use in combination.

7.2.3 Models and training tools

The Tensorflow Model Garden (Tensorflow 2023) is used for implementing the object detection model. It provides multiple implementation of state-of-the-art models which can be configured in many ways, though the documentation is sparse.

Due to training ML models being resource heavy, all training is performed on kaggle.com (Kaggle 2023) which provides up to 30 hours of free computations on dedicated GPUs per week.

7.3 Preprocessing

As a consequence of the large amount of computation needed for preprocessing the mammograms, multiprocessing is utilized spawning a processes in a pool equal to the number of available CPU cores as shown in listing 7.1.

```
177 # ./dataset/scripts/preprocessing.py
178 with multiprocessing.Pool(16) as pool:
179     results = pool.map(preprocess_images, df_unique_images.iterrows())
```

Listing 7.1: Multiprocessing

The image is conditionally flipped based on the sum of the left and right half. As darker (background) pixels has a lower value than brighter pixels, it must mean that the part of the image having the largest sum must be the one containing the breast as shown in listing 7.2.

```
46 # ./dataset/scripts/preprocessing.py
47 center = width // 2
48 left_side = pixel_array[:, :center].sum()
49 right_side = pixel_array[:, center:].sum()
50 is_flipped = left_side < right_side
51
52 if is_flipped:
53     pixel_array = np.fliplr(pixel_array)
```

Listing 7.2: Horizontal flip

The image is applied a CLAHE (Contrast Limited Adaptive Histogram Equalization) operation which enhances the contrast locally by sliding a kernel across the image and creating a histogram of the pixel values. The brightest pixels are redistributed resulting in a higher contrast image and lastly a cleanup is performed setting all leftover pixel values below a 100 to 0 to reduce the amount of noise.

```
61 # ./dataset/scripts/preprocessing.py
62 CLIP_LIMIT = 100
63 clahe = cv.createCLAHE(clipLimit=CLIP_LIMIT, tileGridSize=(5, 5))
64 clahe_img = clahe.apply(pixel_array_cropped)
65 clahe_img[clahe_img <= CLIP_LIMIT + 1] = 0
```

Listing 7.3: CLAHE

Listing 7.4 show the implementation of the mask operation. First the 16 bit image is translated into 8 bit required by the threshold function on line 75. Next a Gaussian blur is applied to smooth out the image which improves the thresholding which binarizes the image. A contour detection is applied where the contour with the largest circumference is selected and filled with pixel values of 255. Lastly the image is masked with the filled contour by applying a bitwise AND operation.

```
72 # ./dataset/scripts/preprocessing.py
73 clahe_image_uint8 = clahe_img // 0xFF
74 clahe_image_uint8 = clahe_image_uint8.astype(np.uint8)
75 blurred_clahe_img = cv.GaussianBlur(clahe_image_uint8,(151,151),0)
76 ret, thresh = cv.threshold(blurred_clahe_img, LOWER_LIMIT, UPPPER_LIMIT, 0)
77 contours, hierarchy = cv.findContours(thresh, mode=cv.RETR_EXTERNAL, method=cv.
       CHAIN_APPROX_NONE)
78
79 largest_contour = utils.get_largest_contour(contours)
80
81 img = cv.drawContours(np.zeros(shape=clahe_img.shape), largest_contour, -1, (255),
       thickness=10)
82 mask = cv.fillPoly(img, [largest_contour], color=(255)).astype(np.uint8)
83 if channels.shape != mask.shape:
84     mask = mask[:, :, np.newaxis]
85 masked_img = channels & mask
```

Listing 7.4: Mask

7.4 Models

7.4.1 ROI Detection

The ROI detection model is implemented as a RetinaNet using Tensorflow Model Garden. The configuration of the desired model can be initialized by the provided experiment factory (exp_factory) module. In listing 7.5 it is specified that the RetinaNet with a ResNet50 FPN backbone should be initialized with pretrained weights on the COCO dataset.

```
1 # ./roi-detection/notebooks/cbis-roi-detection.ipynb (cell 5)
2 exp_config = exp_factory.get_exp_config('retinanet_resnetfpn_coco')
```

Listing 7.5: RetinaNet config initialization

The initialized configuration can be customized as in this case the image size is set to be scaled to 1024x512 as most of the images are of a rectangular form after the preprocessing step. The images are then scaled further to 640x640 which is done in the preprocessing layer of the RetinaNet (Garden 2023).

```
1 # ./roi-detection/notebooks/cbis-roi-detection.ipynb (cell 9 and 10)
2 batch_size = 8
3 num_classes = 1
4
5 HEIGHT, WIDTH = 1024, 512
6 IMG_SIZE = [HEIGHT, WIDTH, 3]
7 ...
8 # Model config.
9 exp_config.task.model.input_size = IMG_SIZE
```

```
exp_config.task.model.num_classes = num_classes + 1

# Training data config.
...
exp_config.task.train_data.parser.aug_scale_max = 1.8
exp_config.task.train_data.parser.aug_scale_min = 0.5
exp_config.task.train_data.parser.aug_rand_hflip = True
...
exp_config.trainer.optimizer_config.learning_rate.type = 'cosine'
exp_config.trainer.optimizer_config.learning_rate.cosine.decay_steps = train_steps
exp_config.trainer.optimizer_config.learning_rate.cosine.initial_learning_rate = 0.001
exp_config.trainer.optimizer_config.learning_rate.end_learning_rate = 0.000001
exp_config.trainer.optimizer_config.warmup.linear.warmup_learning_rate = 0.001
...
exp_config.trainer.best_checkpoint_eval_metric = "AP"
exp_config.trainer.best_checkpoint_export_subdir = "best_ckpt"
```

Listing 7.6: RetinaNet configuration

The amount of classes is set to 2 (background and mass) and some augmentation is configured randomly flipping the image horizontally and scaling it with a factor of 0.5 to 1.8.
The learning rate is configured to decay for each epoch using the cosine function and for each epoch a checkpoint is saved. The best checkpoint is saved in a separate directory and is evaluated by the Average Precision (AP) metric (here mAP and AP are used interchangeably).

The training dataset is augmented by a random rotation, translation, zoom, brightness and contrast. Here it is important to do identical operations on the image and mask when manipulation of pixel position is involved such as rotation. Listing 7.7 shows how this is handled by using a seed ensuring that both the mask and image are rotated the same amount.

```
# ./dataset/scripts/augmentations.py
def augmentation(image, mask):
    ...
    image = tf.keras.layers.RandomRotation(1.0, fill_mode="constant", interpolation="
    nearest", seed=seed)(image)
    mask = tf.keras.layers.RandomRotation(1.0, fill_mode="constant", interpolation="
    nearest", seed=seed)(mask)
    ...
```

Listing 7.7: Full image augmentation

7.4.2 Mass segmentation

The mass segmentation model is implemented using the functional API with Keras which is useful when building complex models with skip connections.
The base U-Net is implemented using different blocks each represented as its own function for reusability.
Listing 7.8 contains the function for creating the entire model. Besides the number of classes and image shape, it also takes the structure of the encoder and decoder stack which is set to a default if not specified. The stack is specified in the order of the encoder

and then reversed for the decoder.
The residual connections between the encoder and decoder are stored in the *skips* variable storing the output of a conv block before the max pooling is applied. Line *8-9* performs the encoding of the image where the *encode()* function outputs a tuple with the current convolution without and with maxpooling.
The decoding is applied on line *16-17* where the stack is reversed as it has to mirror the encoder. Furthermore the skips from the encoder are added for each layer which is done by reverse iterating as well. The output layer performs a convolution on the last layer of the decoder with the sigmoid activation function as the classification is binary.

```
1  # ./mass-segmentation/notebooks/roi-segmentation-transunet.ipynb (cell 16)
2  def create_unet_model(num_classes, image_shape, stack=[32, 64, 128, 256, 512]):
3      skips = {}
4      inputs = tf.keras.Input(shape=image_shape + (3,))
5      x = inputs
6      trans = transformer_block(inputs)
7
8      for i, filters in enumerate(stack[:-1]):
9          skips[i], x = encode(x, filters)
10
11     x = res_conv_block(x, stack[-1])
12     trans = tf.keras.layers.Reshape((x.shape[1], x.shape[2], -1))(trans)
13     x = tf.keras.layers.Add()([x, trans])
14     x = tf.keras.layers.ReLU()(x)
15
16     for i, filters in enumerate(reversed(stack[:-1])):
17         x = decode(x, skips[(len(stack) - 2) - i], filters)
18
19     outputs = tf.keras.layers.Conv2D(classes, 3, padding="same", activation="sigmoid")(x
       )
20     model = tf.keras.Model(inputs, outputs)
21
22     return model
```

Listing 7.8: U-Net model function

The transformer block executes in parallel with the encoder and therefore take the same input. The implementation of the transformer is inspired by the example from the Keras documentation (Salama 2021).
The transformer block is multilayered defaulting to 8 layers which can be processed in parallel increasing performance. For each layer, a multilayered attention matrix is calculated (defaulting to 4 layers) which is combined to one final attention matrix for each layer (line *10-12)*. The generated matrices are combined into one by addition on line *14* and further embedded by a dense layer containing learnable parameters which the model can refine during training. Lastly the function returns the final attention matrix (x) as a vector which can be reshaped to fit the last layer in the U-Net when added.

```
1  # ./mass-segmentation/notebooks/roi-segmentation-transunet.ipynb (cell 16)
2  def transformer_block(x, num_layers = 8, num_heads = 4):
3      patches = Patches(PATCH_SIZE)(x)
4      encoded_patches = PatchEncoder(NUM_PATCHES, PROJECTION_DIM)(patches)
5
6      # Create multiple layers of the Transformer block.
7      for _ in range(num_layers):
8          x1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
```

```
9
10          attention_output = tf.keras.layers.MultiHeadAttention(
11              num_heads=num_heads, key_dim=PROJECTION_DIM, dropout=0.1
12          )(x1, x1)
13
14          x2 = tf.keras.layers.Add()([attention_output, encoded_patches])
15          x3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)(x2)
16          x3 = mlp(x3, hidden_units=[PROJECTION_DIM * 2, PROJECTION_DIM], dropout_rate
    =0.1)
17          encoded_patches = tf.keras.layers.Add()([x3, x2])
18
19      x = tf.keras.layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
20      x = tf.keras.layers.Flatten()(x)
21      x = tf.keras.layers.Dropout(0.5)(x)
22
23      return x
```

Listing 7.9: Transformer block

7.4.3 Pathology classification

The pathology classification is composed of multiple models which outputs are concatenated and further processed by a final dense layered model.
When training the three input images are augmented but for this to be possible it is necessary to apply the same transformations to each image which is achieved by generating a seed fed to each augmentation function for each image as shown in listing 7.10.

```
1 # ./pathology-classification/notebooks/pathology-ensemble-networks.ipynb (cell 6)
2 def augmentation(inputs, labels):
3     seed = random.randint(0, 1000)
4     inputs["original"] = apply_aug(inputs["original"], seed)
5     inputs["margin"] = apply_aug(inputs["margin"], seed)
6     inputs["shape"] = apply_aug(inputs["shape"], seed)
7
8     return inputs, labels
```

Listing 7.10: Pathology classification augmentation

Each submodel is created with the *create_submodel()* function which creates a ResNet50 model pretrained with the "imagenet" weights. The model has a binary output and therefore uses the sigmoid activation function. The ADAM (Adaptive Moment Estimation) optimizer is used with a default learning rate of 1e-4 as this shows to be a good starting point. Furthermore label smoothing is used in the loss function to act as a regularization to the labels making the model less prone to overfitting when training.

```
1 # ./pathology-classification/notebooks/pathology-ensemble-networks.ipynb (cell 15)
2 def create_submodel(input_name):
3     inputs = tf.keras.Input(shape=IMAGE_SHAPE + (3,), name=input_name)
4     resnet = create_resnet(inputs, trainable=True)
5
6     outputs = tf.keras.layers.Dense(1, activation="sigmoid")(resnet)
7     model = tf.keras.Model(inputs=inputs, outputs=outputs)
8
9     model._name = input_name + "_model"
10
```

```
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss=tf.keras.losses.BinaryCrossentropy(label_smoothing=0.25),
        metrics=[
            "accuracy",
            tf.keras.metrics.Precision(),
            tf.keras.metrics.Recall(),
            tf.keras.metrics.AUC()
        ],
    )
    ...
```

Listing 7.11: Pathology classification submodel creation

The final full model is created by concatenating the output of three submodels. Though the final binary output of each model is not wanted as this will reduce the amount of parameters and important information will be lost. Therefore the second last layer of each submodel is selected and concatenated to be input to the first dense layer as shown in listing 7.12 line *10-14*.
The input of each submodel is as well selected as input for the final model as shown on line *22-26* and between the inputs and output, three fully connected layers are created which will do inference on the outputs of the three submodels.
Lastly it is possible to fine tune the final model by setting the three submodels *trainable* attribute to *True* which will unfreeze the weights of the models and make them updatable.

```
# ./pathology-classification/notebooks/pathology-ensemble-networks.ipynb (cell 29)
...
original_model.trainable = False
margin_model.trainable = False
shape_model.trainable = False

x = tf.keras.layers.concatenate([
    original_model.layers[-2].output,
    margin_model.layers[-2].output,
    shape_model.layers[-2].output
])
x = tf.keras.layers.Dense(512, activation="relu")(x)
x = tf.keras.layers.Dropout(0.3)(x)
x = tf.keras.layers.Dense(256, activation="relu")(x)
x = tf.keras.layers.Dropout(0.3)(x)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.3)(x)

outputs = tf.keras.layers.Dense(1, activation="sigmoid")(x)

model = tf.keras.Model(
    inputs=[
        original_model.input,
        margin_model.input,
        shape_model.input
    ],
    outputs=outputs
)
...
```

Listing 7.12: Pathology classification full model

7.5 Deployment

The models are deployed with Flask which is a small lightweight webserver framework written in Python (Pallets 2023). This is convenient for a proof of concept (PoC) as most of the existing code for preprocessing can be reused. This enables the system to run offline with the models loaded in memory fulfilling requirement **NF2.**.
Listing 7.13 show implementation for the endpoint called by the frontend when a mammogram is uploaded. The image is sent as base64 which is preprocessed using the same steps as in section 6.2 and then returned. This also solves the issue of displaying DICOM images in the browser as they are converted to png by the preprocessing module. This enables the application to show the radiologist the preprocessed image fulfilling requirement **F1.2.**.

```
# ./application/server.py
@app.post("/preprocess/<image_type>")
def preprocess(image_type):
    image_bytes = flask.request.data
    pixel_array = preprocessing.preprocess_image(escape(image_type), image_bytes)

    return encode_image_base64(pixel_array)
```

Listing 7.13: Server preprocessing endpoint

When the frontend receives the preprocessed image the analysis endpoint is requested performing the full mammogram analysis pipeline as shown in listing 7.14. Each model is contained in its own module which makes the code more readble but also makes it easy to swap out a model with another, as long as they both conform to the required input and output.
The final result of the analysis is aggregated into an array of each detected mass containing the segmentation image, roi confidence score, pathology classification and metadata. This data can then be used by the frontend to render the masses and their bounding boxes.

```
# ./application/server.py
@app.post("/mammogram/analysis/<roi_confidence_threshold>")
def mammogram_analysis(roi_confidence_threshold):
    image_base64 = flask.request.data
    img_decoded = base64.decodebytes(image_base64)
    image = tf.image.decode_png(img_decoded, channels=3)
    image = tf.expand_dims(image, axis=0)

    candidate_images = roidetection.detect_rois(image, float(roi_confidence_threshold))
    segmentation_images = segmentation.create_segmentations(candidate_images)
    pathologies = classification.classify_rois(candidate_images, segmentation_images)

    result = {
        "segmentations": [{
            "segmentation": segmentation.numpy().flatten().tolist(),
            "width": segmentation.shape[0],
            "height": segmentation.shape[1],
            "roi_bbox": candidate_images[i]["bounding_box"],
            "bbox": get_bounding_box(segmentation, candidate_images[i]["bounding_box"]),
            "bboxConfidence": candidate_images[i]["confidence"].numpy().flatten().tolist
    ()[0],
            "pathology": pathologies[i].numpy().flatten().tolist()[0]
```

```
57          } for i, segmentation in enumerate(segmentation_images)],
58      }
59
60      return json.dumps(result)
```

Listing 7.14: Server analysis endpoint

Figure 7.1 depicts an example of the application where a mammogram has been uploaded and analyzed. Here it is possible to choose the file to be analyzed as well as choosing the ROI confidence threshold for the analysis.

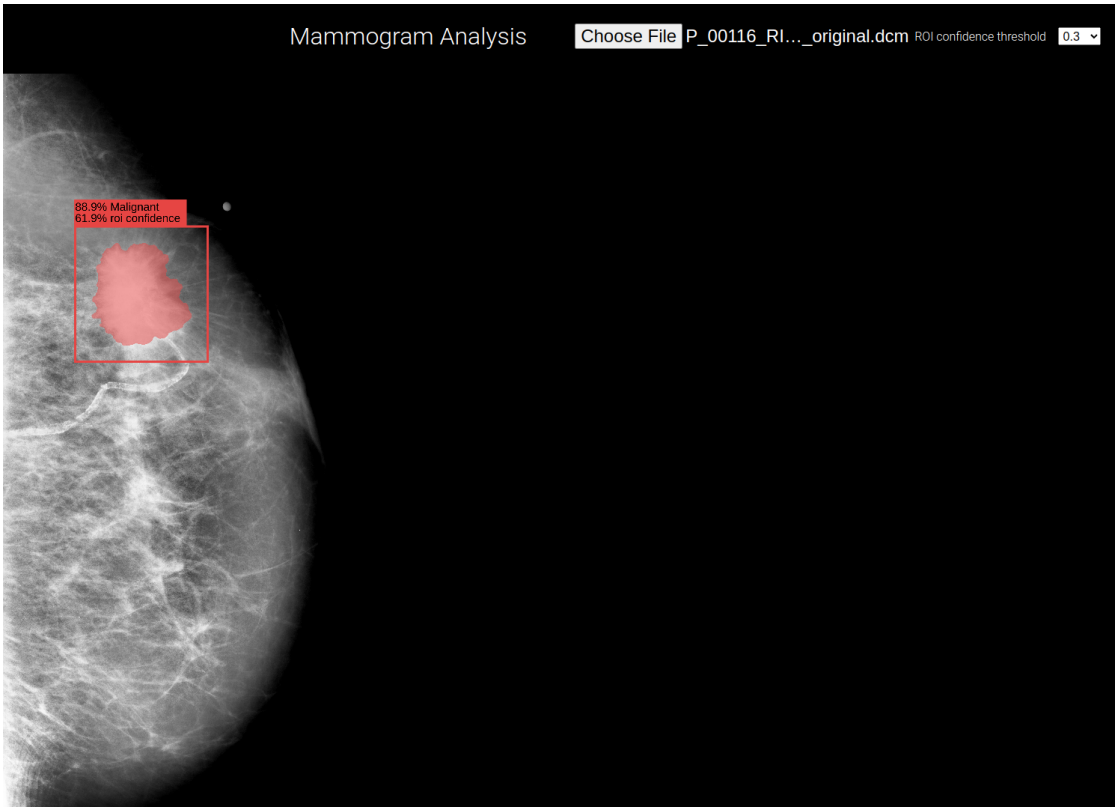


Figure 7.1: Example of application

8 Results

In the following the results of the test dataset is described for each model as well as the results of the end-to-end test of the whole pipeline.
All models have been trained on the training set from where a validation set has been created by splitting the training set 80/20 for hyperparameter tuning. Only the training set have had applied augmentation. All models are finally evaluated on the original test set with no modifications.

8.1 ROI detection

The ROI detection model was trained for 20 epochs with an initial learning rate of 1e-3. where a checkpoint is created for the best performing epoch. The training dataset was augmented by random rotation, translation, zoom, brightness, horizontal flipping and contrast.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.257
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.611
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.158
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.235
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.281
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.318
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.426
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.445
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.450
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.440
```

Listing 8.1: ROI detection test results

Listing 8.1 shows the results of the best epoch where an AP of 0.26 was reached which is an improvement of 0.12 compared to training with no augmentation. Though the results are in the very low end even when comparing to the benchmark of the RetinaNet on the COCO dataset which contains 80 classes (Lin, Goyal, et al. 2018) as shown in table 8.1. This means the model does not fulfill requirement **NF1.2.2.** (*The ROI detection model must have an mAP of >=40%).*

Model	Dataset	AP
RetinaNet-resnet50	COCO	0.32
RetinaNet-resnet50 (**mine**)	CBIS-DDSM	0.26

Table 8.1: AP scores of RetinaNet on different datasets

This low score can potentially be an issue as all misdetections are propagated to the

subsequent models. This heavily affect how high the confidence score threshold can be when filtering the results of the ROI detection for a particular mammogram.

A qualitative inspection of the results shows that the model is able to correctly find some of the ROIs with a fairly good accuracy though at the cost of having a low confidence score which increased the risk of false positives.

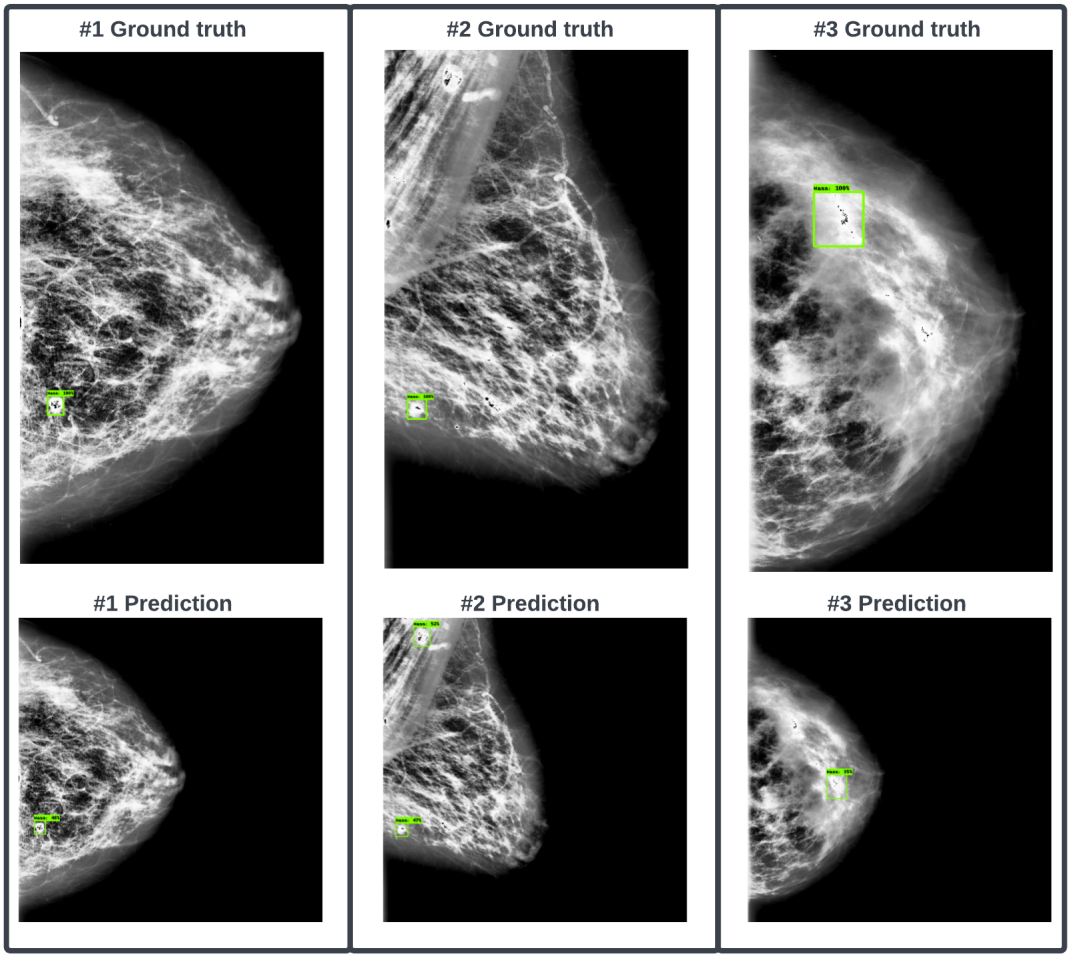


Figure 8.1: ROI detection qualitative inspection

Figure 8.1 shows three samples from the test set with the ground truth on the first row and the prediction of the model on the last row. The confidence threshold is set to 0.3 meaning all detections with lower scores are not considered.
The model is able to find some of the relevant regions though in case 2 it finds an additional region which is a false positive. In case 3 the model completely fails to find the correct ROI but instead detects a false positive which will make the work of the subsequent models redundant as no mass is present in reality.

8.2 Mass segmentation

The mass segmentation model was trained for a total of 50 epochs with 10 epochs with a learning rate of 1e-3, 20 with 1e-4 and further 20 with 1e-5. The training dataset was augmented with random horizontal and vertical flip.

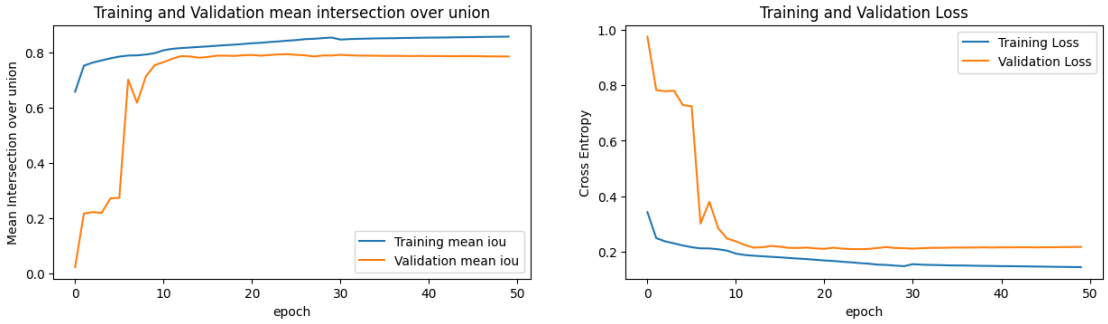


Figure 8.2: Mass segmentation model training history

The model achieved a IoU of 81.12% and an AUC score of 0.95 on the test dataset, which fulfills requirement **NF1.2.3.** (*The segmentation model must have a mean IoU >=80% and an AUC score >=0.79*). Figure 8.2 show the training history and by examining the loss curve it reveals that the model slowly begins to overfit around epoch 20.

Model	IoU	AUC
Connected U-Nets	80.02%	0.79
Multiscale adversarial network	69.72% (dice 82.16%)	0.99
AUNet	68.63% (dice 81.40%)	–
YOLO-LOGO	64.04%	–
TransRes U-Net (mine)	**81.12%**	**0.95**

Table 8.2: Results of segmentation task on CBIS-DDSM

Table 8.2 shows the results of different proposed models for mass segmentation.

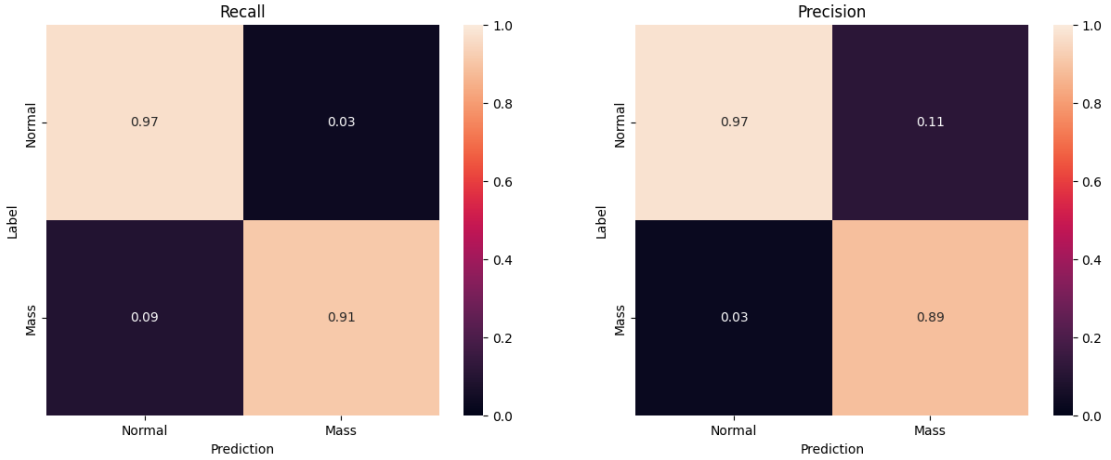


Figure 8.3: Mass segmentation precision and recall on test set

Figure 8.3 shows the recall and precision for each class (Normal or Mass) and reveals that the model handles differentiating between a mass tumor and normal tissue very well by classifying 91% of the pixels correctly as mass and 97% as normal. Furthermore it shows robustness as the precision is 89% for mass and 97% for normal, meaning that when the model predicts one of those classes for a pixel it is correct 93% of the times on average.

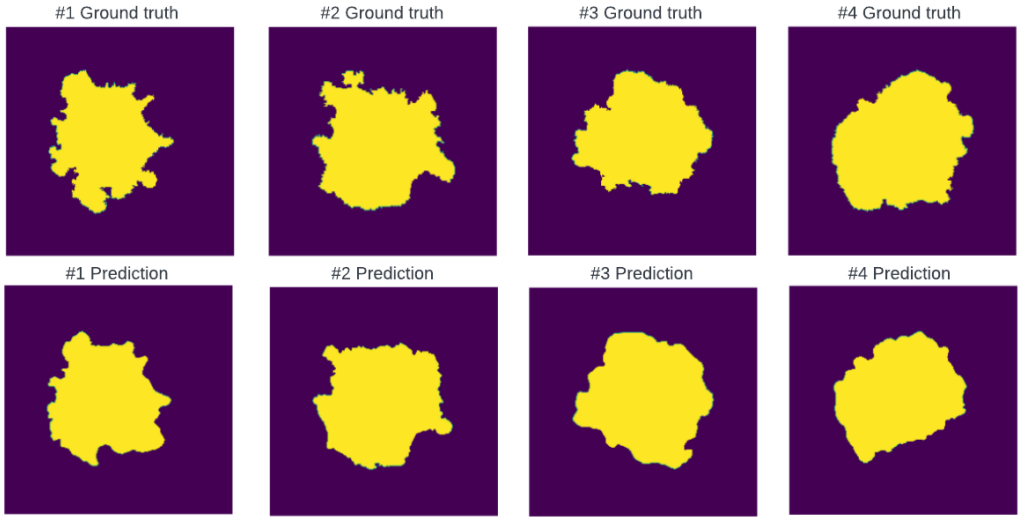


Figure 8.4: Mass segmentation qualitative results

Figure 8.4 shows a some qualitative results generated by the model. Here it is clear that the model has learned the to generate the basic shapes of a mass tumor and additionally manages to generate quite many detailed when inspecting the edges of the mass.

8.3 Pathology classification

The each submodel of the pathology classification model was trained for 60 epochs a learning rate of 1e-4 for the first half and 1e-5 for the last half. The final model was then trained in the same way.

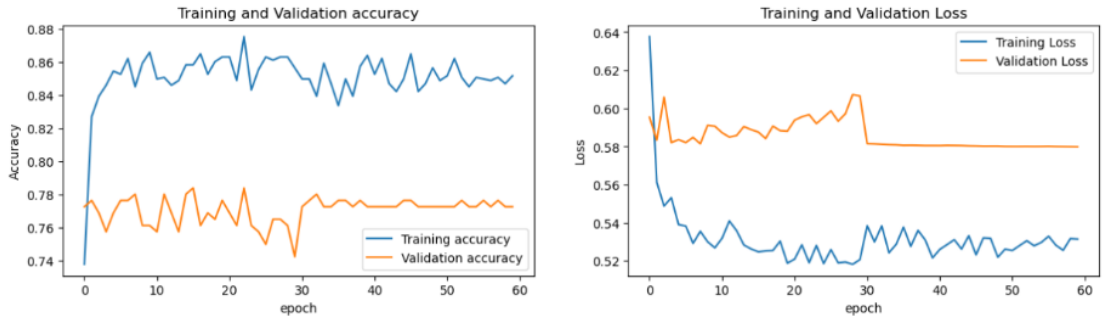


Figure 8.5: Pathology classification training history of the final model

The training history on figure 8.5 shows that the model already around 10th epoch stops learning. This is a sign that not enough parameters are present which is solved by when the full model is fine-tunes where all submodels are set as *trainable*. Though the fine tuning happens with a much lower learning rate at 1e-7 to avoid resetting the weights in the submodels.
After fine-tuning, the model achieves an accuracy of 75.93% and an AUC of 0.80. This does not fulfil requirement **NF1.2.4.** (*The pathology classification model must have an accuracy of >=90% and an AUC of >= 0.9*)

Model	Accuracy	AUC
Ensemble network (single input)	95.13%	0.95
Ensemble network (multiple inputs) **(mine)**	75.93%	0.80

Table 8.3: Results of pathology classification task on CBIS-DDSM

Table 8.3 shows a comparison between the proposed pathology classification model and the one from section 3 which clearly outperforms the ensemble network with multiple inputs.

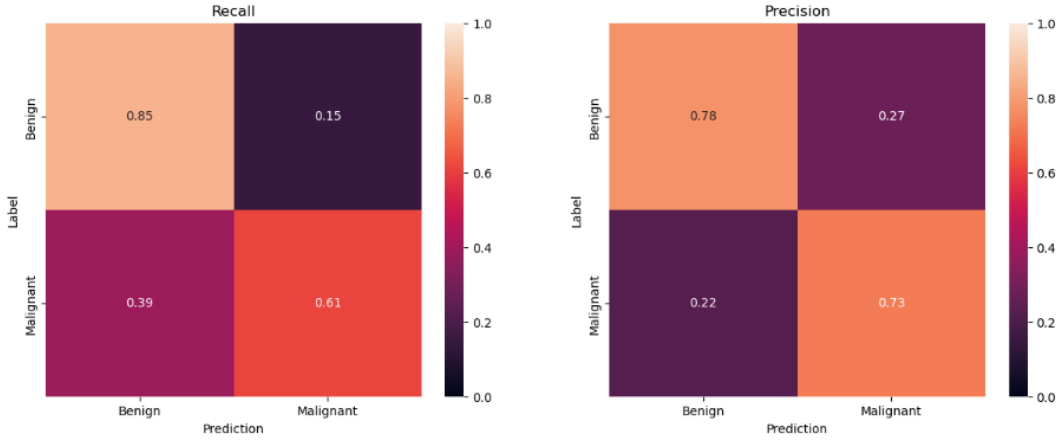


Figure 8.6: Pathology classification recall and precision

Inspecting the recall and precision of the model, it is clear that the model performs best at classifying the benign samples in the test set. Though the model has almost the same precision for each class meaning when it actually predicts one of the classes, there is a 75.5% change on average that it will be correct.

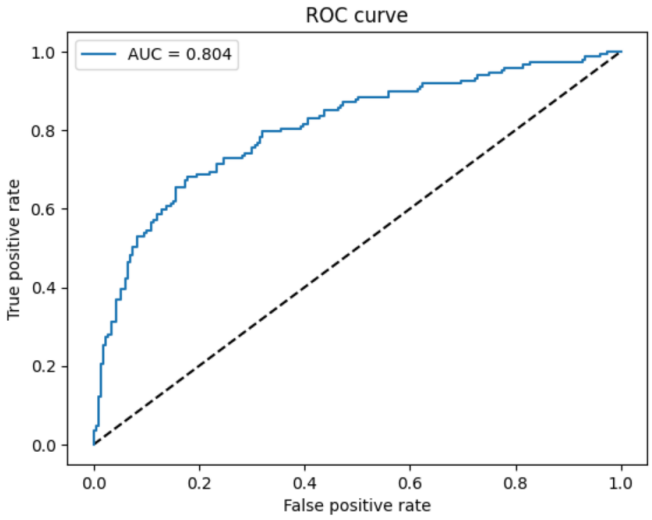


Figure 8.7: Pathology classification ROC curve

The ROC curve on figure 8.7 depicts the relationship between the true positive and false positive rate. This graphs also emphasizes the issues with the model as the true positive rate only slowly reaches 1.0 as the false positive rate increases.

8.4 End-to-end test

Finally an end-to-end test is performed evaluating the overall performance of the model but also assists in fine-tuning hyperparameters such as ROI confidence threshold.
The test is performed by setting up the full analysis pipeline and then feeding it the test dataset with various ROI confidence thresholds.

As the threshold is increase the recall and precision increases as well at the cost of a lower ratio of detected ROIs.

Figure 8.8 shows the development of the recall and precision as the threshold is increased. The recall and precision is calculated from the amount of detected ROIs with a IoU threshold of 50%. This means that for a detected ROI to be a true positive it has to have an IoU of >=50% with the ground truth and its pathology has to be correctly classified.

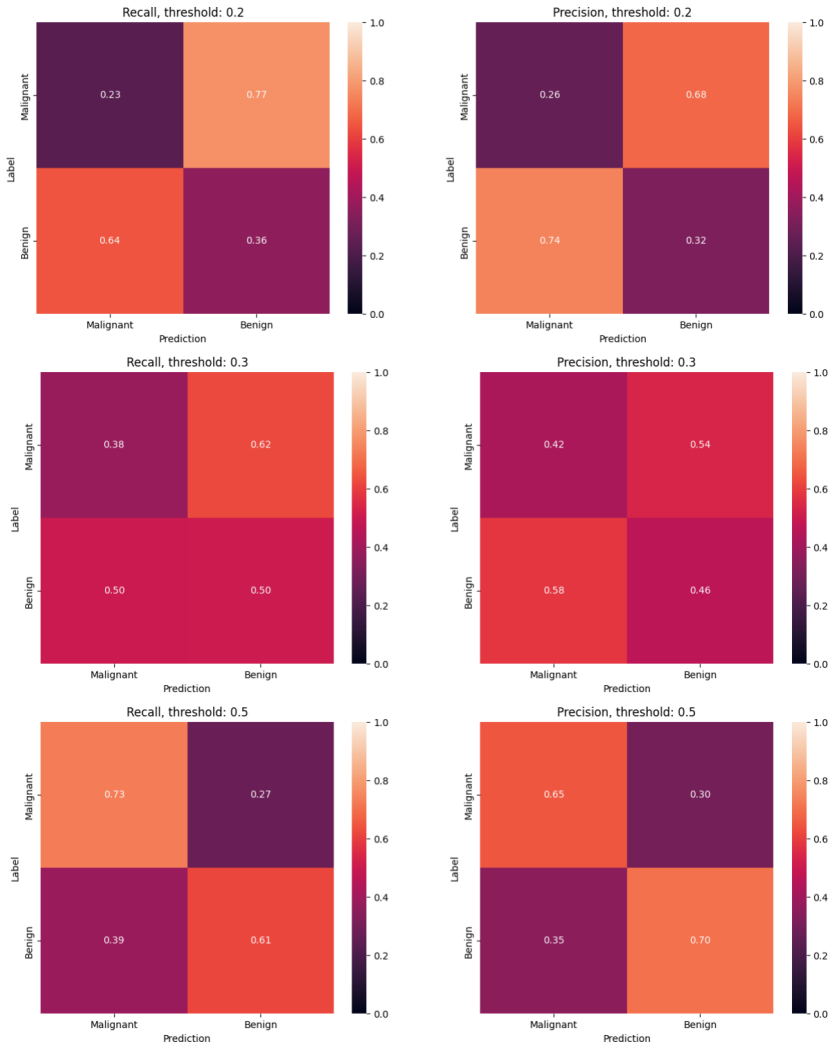


Figure 8.8: End-to-end test ROI confidence threshold 0.2, 0.3, 0.5

The relatively poor performance of the ROI detection model is reflected in the end-to-end test results. Due to the ROI detection model being able to detect up to a 100 ROIs in a single image, the full model is penalizes when the confidence threshold is lower leading to more false positives.
For instance, the recall and precision of the evaluation with a confidence threshold at 0.2 seems poor, though in this case the ROI detection detects a total of 698 regions with 260

of them being correct. In combination with the pathology classification, this corresponds to the model being able to find 54% of all cases and classifying them correctly, though at the cost of providing many false positives.

```
Confidence threshold: 0.2
Total found regions: 698
True regions found: 260
Correctly found regions out of all regions: 0.69
Correctly classified out of true regions: 0.78
Correctly classified out of all regions (true positives): 0.54
False positive rate: 0.71
```

Listing 8.2: Roi confidence threshold 0.2 test results

```
Confidence threshold: 0.3
Total found regions: 374
True regions found: 214
Correctly found regions out of all regions: 0.57
Correctly classified out of true regions: 0.78
Correctly classified out of all regions (true positives): 0.44
False positive rate: 0.55
```

Listing 8.3: Roi confidence threshold 0.3 test results

```
Confidence threshold: 0.5
Total found regions: 112
True regions found: 98
Correctly found regions out of all regions: 0.26
Correctly classified out of true regions: 0.77
Correctly classified out of all regions (true positives): 0.20
False positive rate: 0.33
```

Listing 8.4: Roi confidence threshold 0.5 test results

Listing 8.2, 8.3, 8.4 shows additional data about the results at different confident thresholds. The "True regions found" label describes the amount of detected regions with a IoU threshold of >=50% and "all regions" refers to the actual amount of regions in the test set.
Here it seems that a threshold about 0.3 is a good compromise as the amount of found regions (374) is close to the amount of regions in the test set (378) which lowers the risk of false positives. With this confidence threshold, the final model is able to detect and correctly classify 44% of all cases in the test set with a false positive of 55%. This is a fairly low score which does not fulfill the requirement **NF1.2.1.** (*The overall model should be able to detect >=80% of all mass cases*) making the model, for now, unsuitable for production.
Appendix A contains some qualitative results of the test set randomly selected.

Lastly the model manages to fulfill requirement **NF4.** by having an analysis time of 4-5 seconds including preprocessing the images and rendering it with the segmentations and bounding boxes.

9 Discussion

Due to the final pipeline consisting of multiple sequential layers, errors in the first layers have the biggest impact on the overall performance as the error will propagate to lower level layers. This means that the first layers must have good performance for the overall model to perform well. This is one of the issues with this particular model, as the ROI detection outputs low confidence scores making it more imprecise where the error then propagates to subsequent models.

9.1 Improvements

A perfect ROI detection model could result in a much better overall result potentially being able to detect and classify 75% of all cases correctly. This could be improved even more with a better pathology classification model.

9.1.1 ROI detection

The ROI detection model struggles to find all mass tumors which can be a consequence of the tumors being small compared to the full image and them being hard to differentiate from other tissue.
This issue could be minimized in multiple ways which also could be combined as well.
The first solution (1) is the simplest where the input size of the images could be increased from 640x640 to 1280x1280 by using a different backbone such as SpineNet-143 (Garden 2023). This would require a large amount of training as well as computational resources due to the doubling in image resolution.

The second solution (2) would split the full image into four or more patches depending on the resolution of image. Each patch would then be analyzed by the object detection model, though some post processing would be required when a mass is split between multiple patches. An algorithm should be implemented for stitching together adjacent bounding boxes.

The third solution (3) would include the addition of a new model which were trained to detect whether a mass was present in an image. This would require no change to the existing ROI detection model but would mean that a dataset had to be created containing ROI detections with a binary label. The new model would train on the generated dataset and this way be able to catch potential false positives which would enable to lower the confidence threshold of the ROI detection model.

9.1.2 Mass segmentation

Though the mass segmentation model had good results, it could be further improved. From the qualitative inspection it was clear that it struggled the most around the edges of the mass. This could maybe be optimized by fine-tuning the model with a different loss function, which would penalize the model based on how well it would predict the edges.
This could be achieved by fusioning the IoU function with a sobel operator for edge detection:

$$IoUSobelLoss = 1 - (IoU(y_{true}, y_{pred}) \times w_1 + Sobel(y_{true}, y_{pred}) \times w_2)$$

The Sobel function would return the IoU of the edges and both the IoU and Sobel function would be weighted with w_1 and w_2 where the sum of the weights adds up to 1.

9.1.3 Pathology classification

The pathology classification might benefit from more context awareness as the surrounding tissue is influence by the pathology of the mass. Here it could be interesting to experiment with a vision transformer (same as in the segmentation model) which would apply context awareness.

9.1.4 Preprocessing

The preprocessing step is the first step in the pipeline and therefore has a big impact on how the models perform. More experiments could be done in this step such as having unique CLAHE settings for each model. For instance, the ROI detection model might benefit from a high contrast image removing a lot of noise, though the pathology classification might not benefit from this, as important details could be lost.

9.1.5 Adding more data

By adding more data the models would be able to generalize better, though it can be hard to collect data within the medical field. First all public datasets could be added to the existing such as the INbreast (Moreira et al. 2012) and MIAS (SOCIETY 2017). Secondly synthetic data could be generated using a conditional generative adversarial network (cGAN) which already exists as a python library called Medigan (Osuala et al. 2023).

9.2 Bias in the study from section 3.1

The study by Baccouche, Garcia-Zapirain, and Elmaghraby 2022 on "An integrated framework for breast mass classification" has served as the main comparison for this project, though this project were not able to reach as good performance for pathology classification as they were.
When examining their results and the preprocessing of the dataset, it was discovered that a data leakage between the training and test set has happened which might explain the big difference when comparing the results to this project.
The data leakage occurs in their approach to data augmentation where the original training and test set is merged into one and hereafter augmentation is applied:

> *"...we have augmented the original ROIs four times by rotating them with the angles $\Delta\theta$={0°, 90°, 180°, 270°}. We have also transformed them twice differently using the Contrast Limited Adaptive Histogram Equalization (CLAHE) method. Consequently, raw data of single ROI images were augmented six times into a total of 8802..."*
>
> *"Each mammography dataset is randomly split into groups of 70%, 20%, and 10%, respectively, for training, testing, and validation sets"...* (Baccouche, Garcia-Zapirain, Olea, et al. 2021)

The table of the sample distribution reveals 6161, 1760 and 881 sample for the training, test and validation set respectively. They state that the split has been done randomly, which means that test set must contain copies of the samples in the training only differing by being rotated differently or having a different contrast. This is a problem in three ways:

1. The model evaluation will be biased due to the model having implicitly seen the test set.
2. It makes it difficult to compare their results with others who did not augment the test set
3. It could be fatal to implement this model in production, as the results are unreliable.

It is therefore hard to compare the final model of this project directly to the one from the mentioned study.

10 Conclusion

Machine learning has a big potential in assisting radiologist in analyzing mammograms as it can learn complex patterns from data and this way help detecting mass tumors.

Several techniques should be applied when constructing an ML based system for assisting in mammogram analysis. First (1) the system must be able to detect relatively small tumors (250x250 pixels) in a high resolution mammogram (4000x5000 pixels). This can be solved by introducing a object detection model using the RetinaNet architecture which learns to position bounding boxes outlining the region containing the suspected mass.
Second (2) a segmentation of the mass in the detected region can be generated to be able to visualize where the mass is located but also to increase the amount of information for the last model to process. This is achieved by using a U-Net architecture combined with a transformer to increase the context awareness. Third (3) a last model classifies the pathology of the regions including the segmentation. Because a correlation between the pathology and shape and margin was found, this model takes three inputs: shape (ROI masked with the segmentation), margin (ROI inverse masked with the segmentation) and the original ROI. Each input has a dedicated model using the ResNet50 architecture which outputs are concatenated and parse to a dense layer outputting the binary pathology prediction.

A pipline was built consisting of four steps: preprocessing, ROI detection, mass segmentation and pathology classification which attempts to simulate how a radiologist analyzes mammograms. The ROI detection reached an AP of 0.26 which introduces overall performance issues due to it being one of the first steps in the pipeline meaning errors propagates to subsequent models. The mass segmentation model reached an IoU of 81.12% and an AUC of 0.95 managing to outperform other proposed models. The pathology classification achieved an accuracy of 75.93% which was far from the results of the model proposed by Baccouche, Garcia-Zapirain, and Elmaghraby 2022. Though it was found that the different could be due to a potential data leakage between their training and test set.

Though some of the results were good, the overall performance of the pipeline was not sufficient due the a large amount of false positives. This issue could be accommodated by introducing additional models or by expanding the dataset, though for now, the pipeline did not satisfy all requirements making it currently unsuitable for production.

Bibliography

1. *About DICOM: Overview* (2023). URL: https://www.dicomstandard.org/about. (accessed: 18.04.2023).

2. Baccouche, A., B. Garcia-Zapirain, and A. Elmaghraby (2022). *An integrated framework for breast mass classification and diagnosis using stacked ensemble of residual neural networks*. URL: https://www.nature.com/articles/s41598-022-15632-6. (accessed: 07.03.2023).

3. Baccouche, A., B. Garcia-Zapirain, C. Olea, et al. (2021). *Connected-UNets: a deep learning architecture for breast mass segmentation*. URL: https://www.nature.com/articles/s41523-021-00358-x. (accessed: 20.02.2023).

4. Baeldung (2023). *Intersection Over Union for Object Detection*. URL: https://www.baeldung.com/cs/object-detection-intersection-vs-union. (accessed: 06.05.2023).

5. Bigaard, J. and A. Kvernrød (2023a). *Fordele og ulemper ved screening for brystkræft*. URL: https://www.cancer.dk/forebyg/screening/brystkraeft/fordele-og-ulemper/. (accessed: 09.05.2023).

6. — (2023b). *Sådan foregår screening for brystkræft*. URL: https://www.cancer.dk/forebyg/screening/brystkraeft/saadan-foregaar-undersogelsen/. (accessed: 09.05.2023).

7. Cancer Prevention, Division of, Centers for Disease Control Control, and Prevention (2022). *What Does It Mean to Have Dense Breasts?* URL: https://www.cdc.gov/cancer/breast/basic_info/dense-breasts.htm. (accessed: 19.04.2023).

8. Chen, Jieneng et al. (2021). *TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation*. DOI: 10.48550/ARXIV.2102.04306. URL: https://arxiv.org/abs/2102.04306. (accessed: 09.03.2023).

9. Chen, Juan et al. (2020). "A Novel Multi-Scale Adversarial Networks for Precise Segmentation of X-Ray Breast Mass". In: *IEEE Access* PP, pp. 1–1. DOI: 10.1109/ACCESS.2020.2999198.

10. Chollet, Francois et al. (2015). *Keras*. URL: https://github.com/keras-team/keras.

11. Clinic, Mayo (2023). *Tumor size*. URL: https://www.mayoclinic.org/diseases-conditions/breast-cancer/multimedia/tumor-size/img-20006260. (accessed: 13.05.2023).

12. COCO (2023). *Detection evaluation and metrics*. URL: https://cocodataset.org/#detection-eval. (accessed: 06.05.2023).

13. Cord Technologies, Inc. (2023). *Mean Average Precision (mAP)*. URL: https://encord.com/glossary/mean-average-precision/. (accessed: 06.05.2023).

14. Developers, Google (2022). *ROC Curve and AUC*. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. (accessed: 07.05.2023.

15. Disease Control, Centers of and Prevention (2022). *What Does It Mean to Have Dense Breasts?* URL: https://www.cdc.gov/cancer/breast/basic_info/dense-breasts.htm. (accessed: 09.05.2023).

16. Dosovitskiy, Alexey et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

17. Garden, Tensorflow Model (2023). *COCO Object Detection Baselines*. URL: https://github.com/tensorflow/models/tree/master/official/vision#coco-object-detection-baselines. (accessed: 29.05.2023.

18. Gawron, Edyta Marta (2022). *Hvor mange får kræft?* URL: https://www.cancer.dk/skole/viden-om-kraeft/kraeft-i-tal/hvor-mange-faar-kraeft/. (accessed: 09.05.2023).

19. He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

20. Huda, Walter and R. Brad Abrahams (2015). "X-Ray-Based Medical Imaging and Resolution". In: *American Journal of Roentgenology* 204.4. PMID: 25794088, W393–W397. DOI: 10.2214/AJR.14.13126. eprint: https://doi.org/10.2214/AJR.14.13126. URL: https://doi.org/10.2214/AJR.14.13126.

21. Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

22. Kaggle (2023). *Kaggle*. URL: https://www.kaggle.com/.

23. Knop, Ann Søegaard (2021). *Brystkræft, fakta*. URL: https://www.sundhed.dk/borger/patienthaandbogen/brystsygdomme/sygdomme/brystkraeft/brystkraeft-fakta/. (accessed: 03.02.2023).

24. Lin, Tsung-Yi, Piotr Dollár, et al. (2017). *Feature Pyramid Networks for Object Detection*. arXiv: 1612.03144 [cs.CV].

25. Lin, Tsung-Yi, Priya Goyal, et al. (2018). *Focal Loss for Dense Object Detection*. arXiv: 1708.02002 [cs.CV].

26. Lindskow, T. et al. (2020). *Kunstig intelligens i det danske sundhedsvæsen*. URL: https://ugeskriftet.dk/videnskab/kunstig-intelligens-i-det-danske-sundhedsvaesen. (accessed: 09.05.2023).

27. Liu, K. et al. (2021). *Weakly-supervised High-resolution Segmentation of Mammography Images for Breast Cancer Diagnosis*. URL: https://arxiv.org/pdf/2106.07049.pdf. (accessed: 07.03.2023).

28. Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

29. Moreira, I. C. et al. (2012). *INbreast: toward a full-field digital mammographic database*. URL: https://pubmed.ncbi.nlm.nih.gov/22078258/. (accessed: 13.05.2023).

30. Niknejad, Mohammadtaghi (2022). *Breast imaging-reporting and data system (BI-RADS)*. URL: https://radiopaedia.org/articles/breast-imaging-reporting-and-data-system-bi-rads. (accessed: 19.04.2023).

31. NumPy (2023). *NumPy*. URL: https://numpy.org/.

32. OpenCV (2023). *OpenCV*. URL: https://opencv.org/.

33. Osuala, Richard et al. (2023). "medigan: a Python library of pretrained generative models for medical image synthesis". In: *Journal of Medical Imaging* 10.6, p. 061403.

34. Pallets (2023). *Flask*. URL: https://flask.palletsprojects.com/en/2.3.x/. (accessed: 27.05.2023.

35. Patientsikkerhed, Dansk Selskab For (2019). *Veje til bedre diagnoser*. URL: https://patientsikkerhed.dk/wp-content/uploads/2023/02/vejetilbedrediagnoserdec2019.pdf. (accessed: 02.03.2023).

36. Redmon, Joseph et al. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. DOI: 10.48550/ARXIV.1506.02640. URL: https://arxiv.org/abs/1506.02640. (accessed: 07.03.2023).

37. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597. (accessed: 07.03.2023).

38. Salama, Khalid (2021). *Image classification with Vision Transformer*. URL: https://keras.io/examples/vision/image_classification_with_vision_transformer/. (accessed: 26.05.2023).

39. Sawyer-Lee, R. et al. (2016). *Curated Breast Imaging Subset of Digital Database for Screening Mammography (CBIS-DDSM) (Version 1) [Data set]*. URL: https://doi.org/10.7937/K9/TCIA.2016.7O02S9CY. (accessed: 01.02.2023).

40. Society, American Cancer (2022). *What Doe the Doctor Look for on a Mammogram*. URL: https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/what-does-the-doctor-look-for-on-a-mammogram.html. (accessed: 09.05.2023).

41. SOCIETY, MAMMOGRAPHIC IMAGE ANALYSIS (2017). *MiniMammographic Database*. URL: https://www.kaggle.com/datasets/kmader/mias-mammography. (accessed: 13.05.2023).

42. Su, Yongye et al. (2022). "YOLO-LOGO: A transformer-based YOLO segmentation model for breast mass detection and segmentation in digital mammograms". In: *Computer Methods and Programs in Biomedicine* 221, p. 106903. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2022.106903. URL: https://www.sciencedirect.com/science/article/pii/S0169260722002851.

43. Sun, Hui et al. (2018). *AUNet: Attention-guided dense-upsampling networks for breast mass segmentation in whole mammograms*. DOI: 10.48550/ARXIV.1810.10151. URL: https://arxiv.org/abs/1810.10151. (accessed: 07.03.2023).

44. team, The pandas development (Feb. 2020). *pandas-dev/pandas: Pandas*. Version latest. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

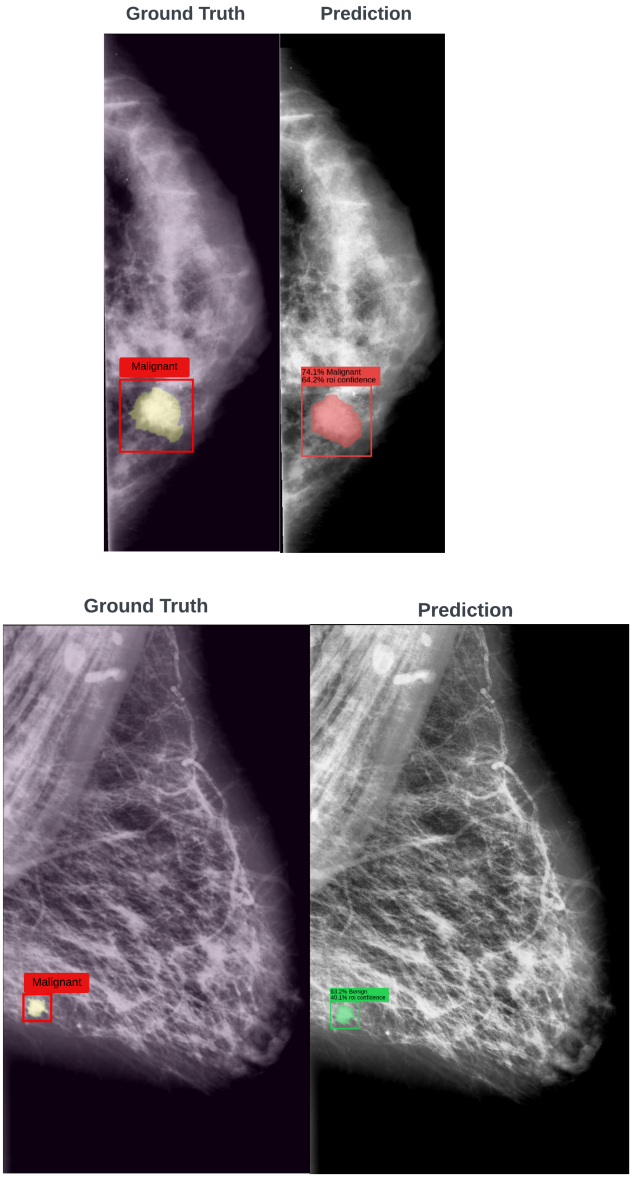
45. Teglgård, J. and Ritzau (2017). *Overlæge stillede forkert diagnose i 21 kræftsager*. URL: https://nyheder.tv2.dk/samfund/2017-10-31-overlaege-stillede-forkert-diagnose-i-21-kraeftsager-0. (accessed: 09.05.2023).

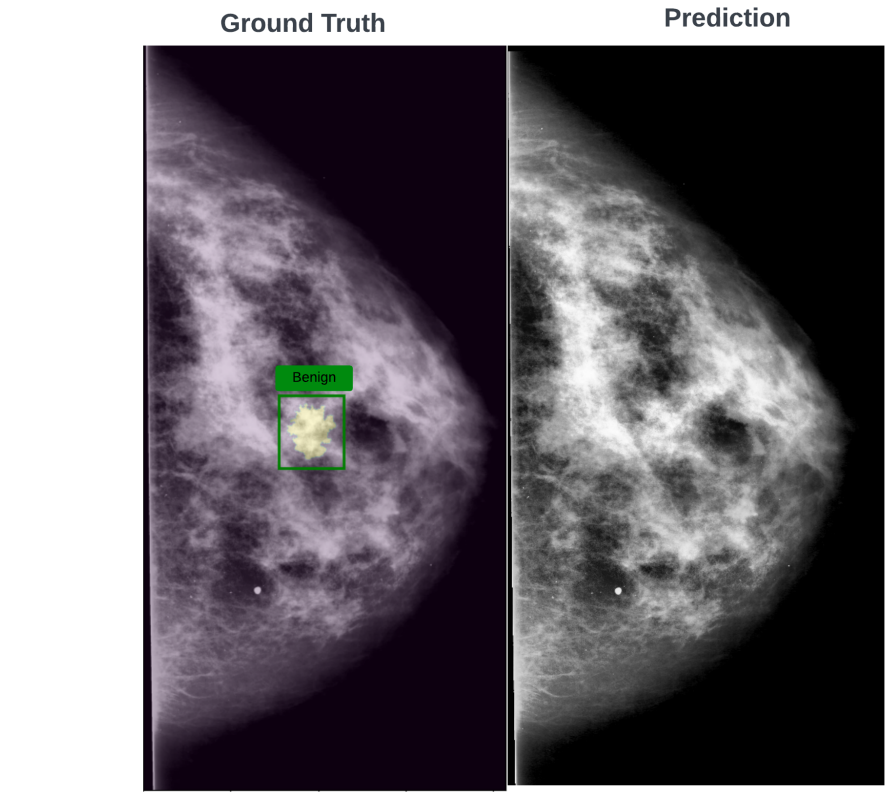
46. Tensorflow (2023). *Tensorflow Model Garden*. URL: https://www.tensorflow.org/tfmodels.

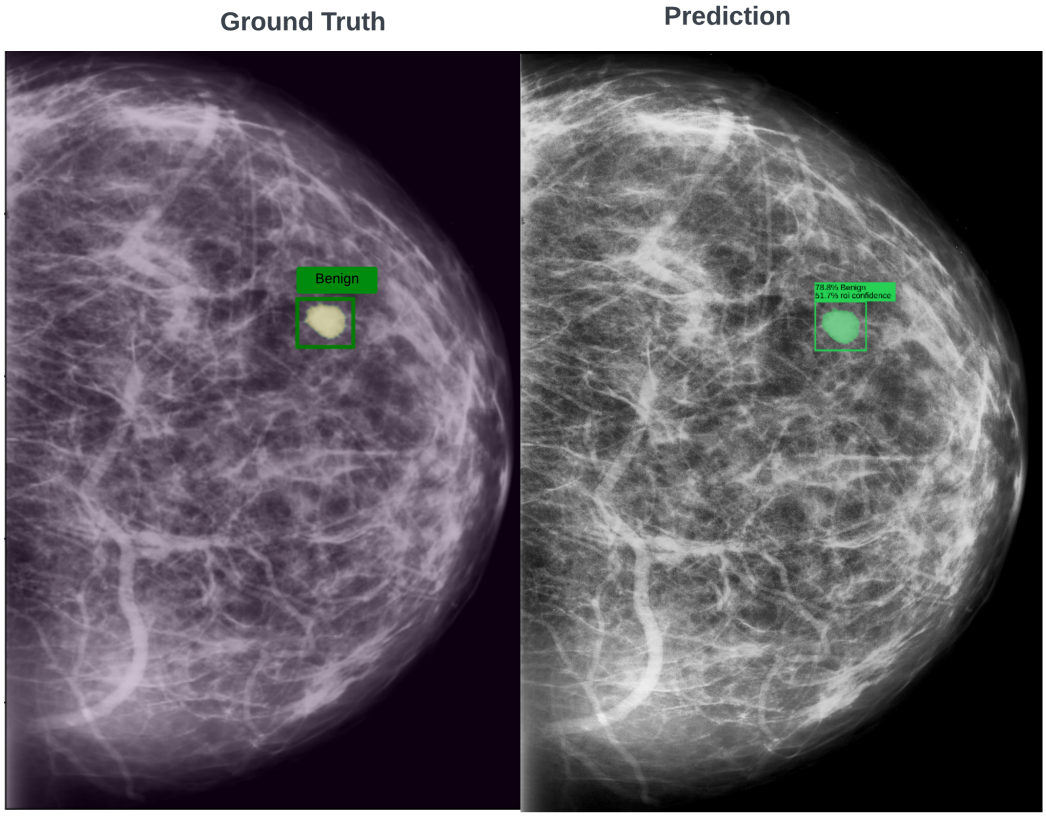
47. Valanarasu, Jeya Maria Jose et al. (2021). *Medical Transformer: Gated Axial-Attention for Medical Image Segmentation*. DOI: 10.48550/ARXIV.2102.10662. URL: https://arxiv.org/abs/2102.10662. (accessed: 07.03.2023).

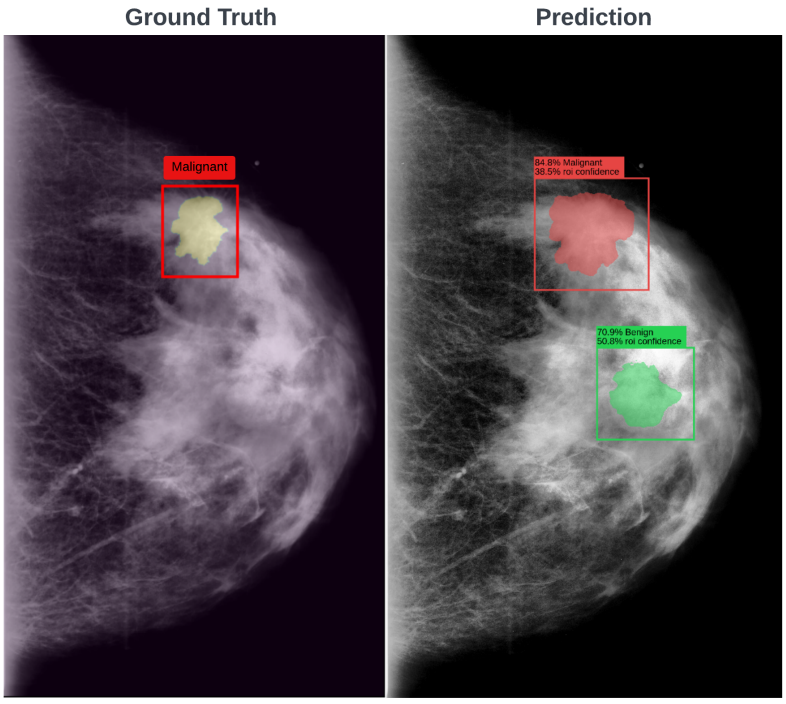
48. Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

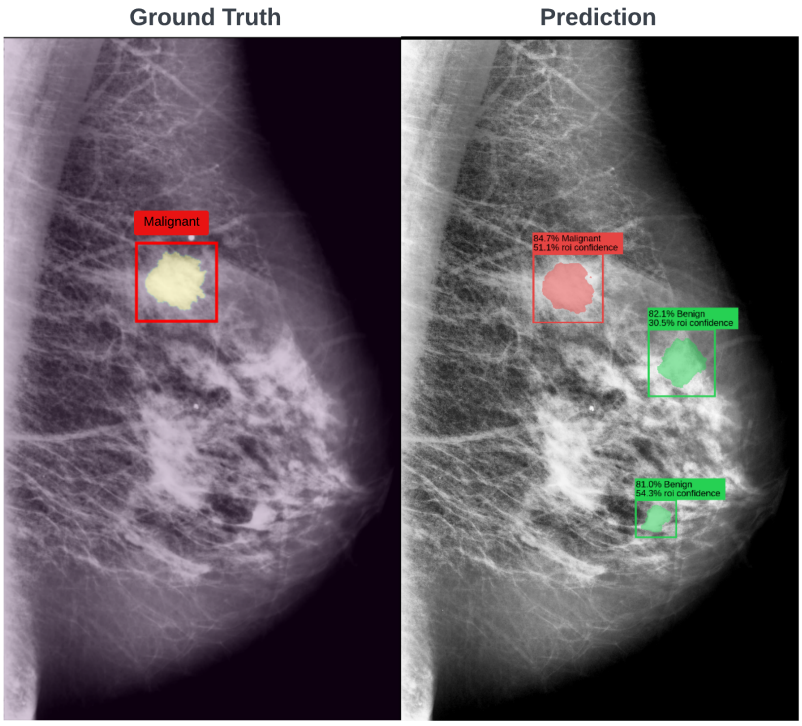
Appendix A

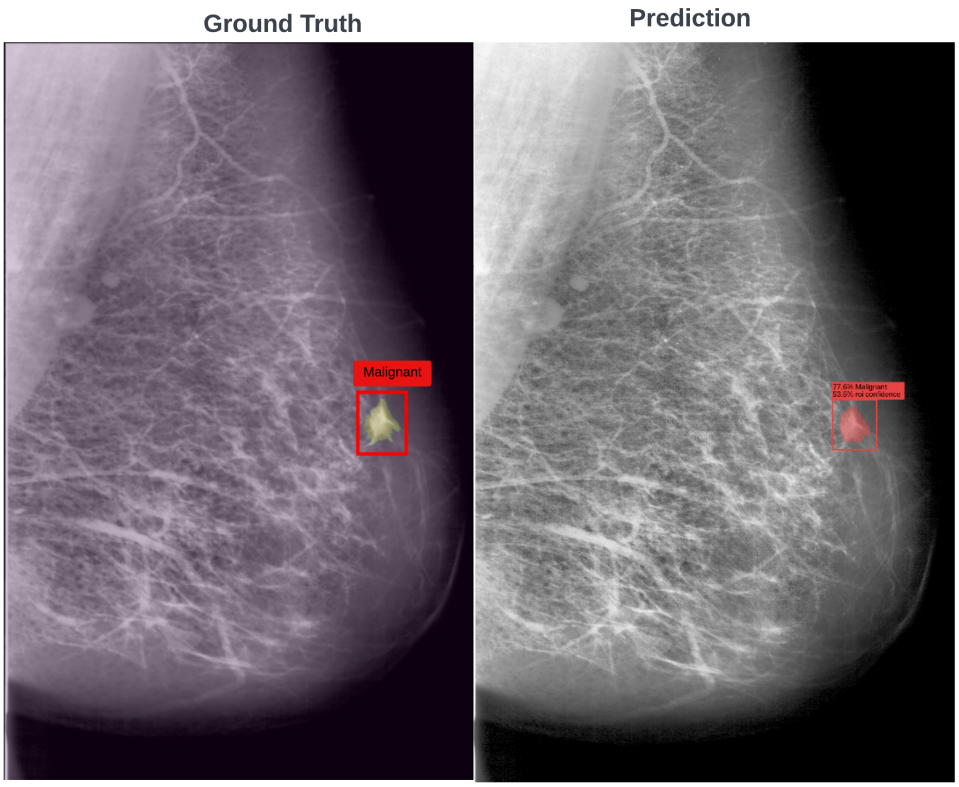

Ground Truth
Prediction
Malignant
74.1% Malignant
54.2% roi confidence
Ground Truth
Prediction
Malignant

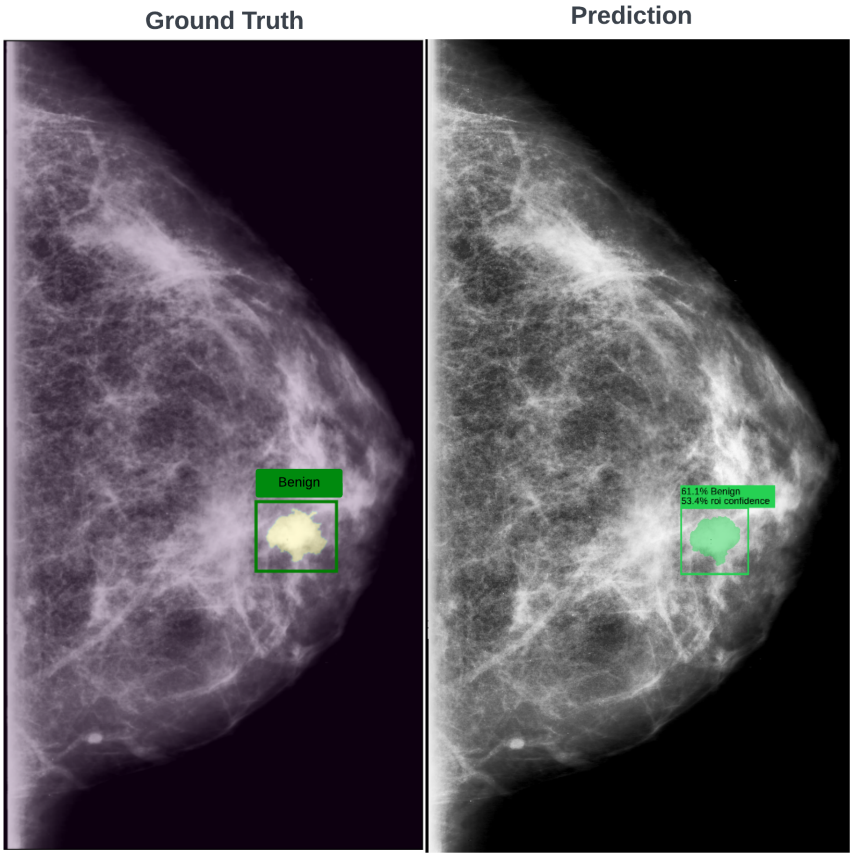

Ground Truth
Prediction
Benign


Ground Truth
Prediction
Benign


Ground Truth
Prediction
Malignant
84.8% Malignant
38.5% roi confidence
70.9% Benign
50.8% roi confidence


Ground Truth
Prediction
Malignant
84.7% Malignant
51.1% roi confidence
82.1% Benign
30.5% roi confidence
81.0% Benign
54.3% roi confidence


Ground Truth
Prediction
Malignant


Ground Truth
Prediction
Benign
61.1% Benign
53.4% roi confidence