Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Máster en Data Science & Entrepreneurship
Master in Data Science & Entrepreneurship

Trabajo Fin de Máster
Master Thesis

Breast Cancer Segmentation with Weakly-Supervised Connected U-Nets

Breast Cancer Segmentation with Weakly-Supervised Connected U-Nets

Autor: Guillermo Nájera Lavid
Author: Guillermo Nájera Lavid

Madrid, septiembre, 2024 /September, 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid.

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Trabajo Fin de Máster

Master Thesis

Máster en « Breast Cancer Segmentation With Weakly-Supervised Connected U-Nets»

Master in « Breast Cancer Segmentation With Weakly-Supervised Connected U-Nets»

Título: Breast Cancer Segmentation With Weakly-Supervised Connected U-Nets

Title: Breast Cancer Segmentation With Weakly-Supervised Connected U-Nets

«Septiembre 2024» /«September 2024»

Author: Guillermo Nájera Lavid

Tutor

Supervisor:

Alejandro Rodríguez

Full Professor at the Department of Computer Languages and Systems and Software Engineering

Universidad Politécnica de Madrid

Co-Tutor

Co-supervisor:

Simon Malinowski

Professor and responsible for M2 Miage Data Science Master

ISTIC Faculty

Institute for Research in Computer Science and Random Systems (IRISA) research center: https://www.irisa.fr/en

Université de Rennes 1

Resumen

El cáncer de mama es una de las principales causas de muertes relacionadas con el cáncer en mujeres a nivel mundial, lo que denota la imperiosa necesidad de una detección temprana y un diagnóstico preciso. La resonancia magnética dinámica con contraste (DCE-MRI) se ha convertido en una herramienta fundamental en la detección del cáncer de mama, ya que ofrece información detallada sobre la morfología, la textura y la cinética del tumor. Sin embargo, segmentar con precisión las regiones cancerosas en estas resonancias es una tarea compleja que tradicionalmente se ha realizado de manera manual. Un proceso laborioso, que consume mucho tiempo y está sujeto a variabilidad. Este proyecto se centra en abordar estos desafíos mediante la implementación de una estrategia de aprendizaje débilmente supervisado llamada Similarity-Aware Propagation Learning (SimPLe), diseñada para mejorar el proceso de segmentación en DCE-MRI utilizando anotaciones mínimas.

La estrategia SimPLe introduce un proceso de aprendizaje en tres fases que optimiza la red de segmentación utilizando solo anotaciones mediante puntos extremos que representan los límites más distantes de la región cancerosa en las resonancias 3D. El método comienza con la generación de pseudo-máscaras iniciales utilizando un algoritmo específico, guiado por los puntos extremos para aproximar los bordes del tumor. Estas pseudo-máscaras se utilizan en la primera fase de entrenamiento, permitiendo que la arquitectura de U-Net conectada aprenda tareas básicas de segmentación. En la segunda fase, la red se somete a un ajuste fino mediante una estrategia de propagación consciente de la similitud, que refina la segmentación propagando etiquetas basadas en similitudes de características entre los vóxeles etiquetados y no etiquetados. La fase final implica reentrenar la red con las pseudo-máscaras actualizadas, mejorando aún más la precisión de la segmentación.

Este estudio implementó inicialmente la estrategia SimPLe en el conjunto de datos original de DCE-MRI proporcionado por los investigadores, que contenía 206 exploraciones de cáncer de mama confirmadas por biopsia y anotadas por radiólogos. Para ampliar la aplicabilidad de este enfoque, se ha buscado aplicarlo al conjunto de datos Duke-Breast-Cancer-MRI, que consta de 922 exploraciones DCE-MRI con resoluciones de imagen variadas y anotaciones exhaustivas de radiólogos. Sin embargo, adaptar la estrategia SimPLe a este nuevo conjunto de datos plantea desafíos significativos debido a diferencias en los estilos de anotación y las estructuras de datos.

Para abordar esto, se desarrollará un enfoque modificado de generación de pseudo-máscaras, ajustando estos puntos extremos de una manera distinta, que permita adaptar el algoritmo a un uso de datos más general, para después entrenar nuestro propia implementación del modelo mencionado ajustándolo no solo a nuestro conjunto de datos, sino también a cualquier posible representación más generalizada de este tipo de datos.

Abstract

Breast cancer is one of the leading causes of cancer-related deaths in women worldwide, highlighting the urgent need for early detection and accurate diagnosis. Dynamic contrast-enhanced magnetic resonance imaging (DCE-MRI) has become a fundamental tool in breast cancer detection, providing detailed information about tumor morphology, texture, and kinetics. However, accurately segmenting cancerous regions in these MRIs is a complex task that has traditionally been done manually, a laborious, time-consuming process subject to variability. This project focuses on addressing these challenges through the implementation of a weakly supervised learning strategy called Similarity-Aware Propagation Learning (SimPLe), designed to enhance the segmentation process in DCE-MRI using minimal annotations.

The SimPLe strategy introduces a three-phase learning process that optimizes the segmentation network using only annotations through extreme points that represent the most distant boundaries of the cancerous region in 3D MRIs. The method begins with the generation of initial pseudo-masks using a specific algorithm guided by the extreme points to approximate the tumor edges. These pseudo-masks are used in the first training phase, allowing the connected U-Net architecture to learn basic segmentation tasks. In the second phase, the network undergoes fine-tuning through a similarity-aware propagation strategy, which refines the segmentation by propagating labels based on feature similarities between labeled and unlabeled voxels. The final phase involves retraining the network with updated pseudo-masks, further improving the segmentation accuracy.

This study initially implemented the SimPLe strategy on the original DCE-MRI dataset provided by the researchers, containing 206 scans of biopsy-confirmed breast cancer, annotated by radiologists. To expand the applicability of this approach, an attempt will be made to apply it to the Duke-Breast-Cancer-MRI dataset, which consists of 922 DCE-MRI scans with varied image resolutions and exhaustive radiologist annotations. However, adapting the SimPLe strategy to this new dataset presents significant challenges due to differences in annotation styles and data structures.

To address this, a modified approach for pseudo-mask generation will be developed, adjusting these extreme points differently to make the algorithm adaptable to more general data usage. This will enable us to train our own implementation of the mentioned model, adjusting it not only to our dataset but also to any possible generalized representation of this type of data.

Table of Contents

1 Introduction

Breast cancer remains one of the most significant health challenges worldwide, being a leading cause of cancer-related deaths among women. As of 2024, it continues to be a significant health concern and accounts for 32% of all new cancer diagnoses in women, marking it as the most prevalent cancer in this demographic. The incidence of breast cancer has been increasing annually by about 0.6% since the mid-2000s. This trend is primarily attributed to the rise in localized and hormone receptor–positive disease diagnoses. In terms of mortality, breast cancer is the second leading cause of cancer-related deaths among women, emphasizing the critical need for effective management and treatment strategies. [1]

Notably, there has been a shift in cancer diagnoses towards younger women under the age of 50, a trend that differs from previous years. This shift is believed to be caused by a combination of factors, including lifestyle changes and environmental exposures. Despite this, overall cancer survival rates have improved, with breast cancer no longer being an exception due to advancements in early detection and treatment. However, disparities persist, with certain racial groups experiencing higher incidences and mortality rates due to factors like access to healthcare and socio-economic differences. [2]

The early detection and precise diagnosis of breast cancer are critical, directly influencing treatment decisions and improving patient prognosis. In this context, Dynamic Contrast-Enhanced Magnetic Resonance Imaging (DCE-MRI) has emerged as a crucial tool in the breast cancer screening arsenal. Its ability to provide detailed insights into tumor morphology, texture, and kinetic heterogeneity sets it apart from traditional imaging modalities like mammography and ultrasonography.

However, the interpretation of DCE-MRI data poses its own set of challenges, particularly in the accurate segmentation of cancerous regions. This segmentation process is pivotal in aiding radiologists with treatment planning and prognosis assessment. Traditionally, this task has been performed manually, which is not only time-consuming and labor-intensive but also prone to variability between different observers. The advent of automated segmentation algorithms, especially those leveraging the power of deep learning, has marked a significant advance. These algorithms, particularly Convolutional Neural Networks (CNNs), have shown promising results in enhancing the accuracy and efficiency of the segmentation process. Yet, they face a significant hurdle: the requirement for extensive, accurately labeled datasets, which are particularly challenging to acquire in the medical field due to the complexity and labor involved in the annotation process.

Weakly-supervised learning is a methodology that has risen in prominence to address the need for large volumes of labeled data. This approach utilizes limited or imprecise annotations to train models, thus substantially reducing the burden of annotation. In the realm of breast cancer segmentation in DCE-MRI, including the use of extreme points, bounding boxes, and scribbles.

Before we decided to delve into this final approach, we performed additional research probing different things that had been done before, falling into a range of seminal papers, each contributing uniquely to the field. This includes a paper presenting an innovative Multi-level Semantic-guided Contrastive Domain Adaptation (MSCDA) framework for breast MRI segmentation, which addresses the challenge of domain shift in small datasets. We also examined a significant study on large-scale classification of breast MRI exams using deep convolutional networks, highlighting the effectiveness of machine learning in distinguishing between malignant and benign findings. Further, we explored research articles discussing the state-of-the-art AI methodologies and their implications in clinical settings for breast MRI analysis, such as a Multi Scale Curriculum CNN, a as well as a Nature publication that offers critical insights into recent technological advancements and clinical studies in the field. Additionally, a PubMed article provided substantial research findings in a Machine Learning Approach to Radiogenomics of Breast Cancer, or comprehensive reviews related to breast cancer MRI. Together, these studies form a rich tapestry of knowledge, offering diverse perspectives on the challenges and advancements in breast MRI analysis, particularly in the context of cancer detection and segmentation.

The focus of our study is a novel weakly-supervised strategy known as "Similarity-Aware Propagation Learning" (SimPLe), specifically designed for breast cancer segmentation in DCE-MRI. This innovative approach employs extreme points as sparse annotations and encompasses a three-phase learning process: initial training with pseudo-masks, fine-tuning through a similarity-aware propagation strategy, and retraining with refined pseudo-masks. The SimPLe strategy aims to optimize the learning capacity of the network while relying on minimal annotations, representing a significant stride in the field of efficient and precise medical image segmentation.

This report is dedicated to an in-depth examination of the SimPLe strategy, scrutinizing its methodology, execution, and effectiveness. Through this comprehensive analysis, we aim to contribute meaningfully to the ongoing advancements in breast cancer diagnostic techniques and treatment planning methodologies.

2 The SimPLe Strategy

2.1 Overview

The paper we based the work on is titled "SimPLe: Similarity-Aware Propagation Learning for Weakly-Supervised Breast Cancer Segmentation in DCE-MRI." [1] This paper presents a significant contribution to the field of medical imaging, particularly in the segmentation of breast cancer in DCE-MRI images. Authored by Yuming Zhong and Yi Wang, affiliated with the Smart Medical Imaging Learning and Engineering (SMILE) Lab and Medical UltraSound Image Computing (MUSIC) Lab at Shenzhen University Medical School.

Zhong et al. bring their extensive expertise in medical imaging to address one of the persistent challenges in the field: the labor-intensive and time-consuming process of segmenting breast cancer regions in DCE-MRI images. Recognizing the limitations of existing segmentation methods, which predominantly rely on extensive and detailed annotations, the authors propose a novel approach to streamline this process while maintaining high accuracy.

The core of their study is the introduction and exploration of a weakly-supervised learning strategy named SimPLe, which stands for Similarity-Aware Propagation Learning. This innovative approach is poised to transform the way segmentation tasks are handled in medical imaging, particularly in DCE-MRI. By employing a three-phase learning process, SimPLe initially trains a neural network using pseudo-masks generated from extreme points within the images. This is followed by a fine-tuning phase that utilizes feature similarity between labeled and unlabeled data points to propagate labels more accurately. The final phase involves retraining the network with these refined pseudo-masks, enhancing the overall accuracy of the segmentation.

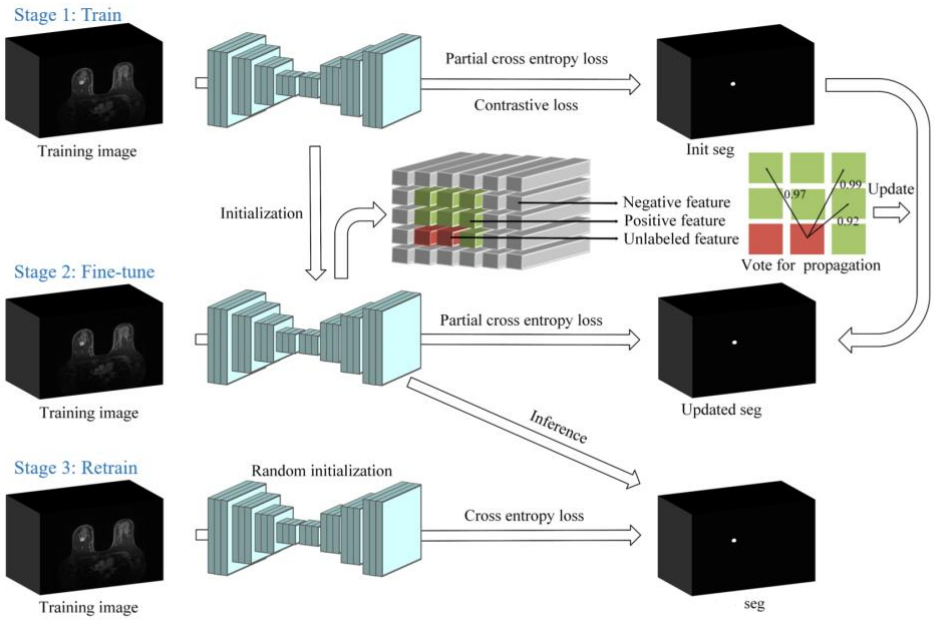

Figure 1: The schematic illustration of the proposed similarity-aware propagation learn- ing (SimPLe) and the train - fine-tune - retrain procedure for the breast cancer seg- mentation in DCE-MRI.

The ambition of this paper extends beyond just proposing a new method. It aims to demonstrate the efficacy of SimPLe in reducing the annotation burden drastically, compared to fully supervised methods and other weakly-supervised approaches. In doing so, the authors endeavor to make a meaningful impact in the realm of medical imaging, specifically targeting improvements in the tools and techniques available for breast cancer screening and diagnosis. This innovative approach has the potential to offer more efficient, accurate, and accessible breast cancer detection methods, which is crucial in the ongoing fight against this widespread disease.

2.2 Dataset

The dataset utilized by their team is a significant collection of clinical data gathered from the Cancer Center of Sun Yat-Sen University. Comprising 206 DCE-MRI scans, each one of them containing biopsy-proven breast cancer.

Each scan in this dataset was performed using a 1.5T MRI scanner, which gave DCE-MRI sequences obtained with gadolinium-based contrast agents, enhancing the visibility and detail of blood vessels and tissue flow, including a TR/TE of 4.43ms/1.50ms and a flip angle of 10 degrees.

The dataset includes images with two distinct resolutions: 0.379×0.379×1.700 mm^3 and 0.511×0.511×1.000 mm^3, which, according to the authors, increases the dataset's robustness, capturing varying levels of detail across different scans. All cancerous regions and extreme points in the images were annotated by an experienced radiologist using ITK-SNAP software. These annotations were then confirmed by another radiologist, ensuring the precision of the segmentation labels used in the study.

The dataset was divided in such a way that 21 scans were destined the training set and the remainder for testing.

3 Initial Pseudo Mask Generation

The initial pseudo-mask generation, as described in the paper, is the first and most critical step in the weakly-supervised learning strategy. This process is implemented in the generate_init_mask.py script which can be found in their GitHub repository, in addition to the rest of the code. It begins with the identification of extreme points within the 3D DCE-MRI scans. These points, representing the furthest extents of the cancerous region in each dimension, provide an informative basis for initial tumor segmentation.

The script, adhering to the methodology outlined in the paper, translates these extreme points into scribbles using the Dijkstra algorithm. These scribbles act as preliminary sketches, delineating the rough boundaries of the tumor. Following this, the script dilates these scribbles to increase the number of foreground seeds to enhance the accuracy of the subsequent random walker algorithm.

The random walker algorithm is then applied to this preliminary segmentation. This algorithm generates a probabilistic map indicating the likelihood of each voxel belonging to the tumor region. The script's implementation of the random walker algorithm involves an iterative process, refining the segmentation with each iteration (7 in total, while the reason is not explicitly mentioned, it is a number likely originated from empirical evidence). This iterative refinement progressively classifies voxels as new foreground or background seeds based on updated probability thresholds, thereby improving the accuracy of the tumor representation.

The culmination of this process is the creation of the final pseudo-masks. These masks categorize each voxel into one of three categories: negative (non-tumor) marked as "0", positive (tumor) marked as "1", or unlabeled marked as "2". This categorization, derived from the probability map and informed by the bounding box encompassing the extreme points, provides an initial segmentation of the tumor. This is a critical starting point for the neural network training since everything resulting from it later relies mainly on this.

4 The U-Net Architecture

The paper introduces a novel connected U-Net architecture, a significant adaptation of the standard U-Net model, known for its efficacy in not only medical image segmentation but also every segmentation task performed with deep learning. This connected U-Net forms the backbone of the authors' approach to weakly-supervised learning for breast cancer segmentation in DCE-MRI images. The architecture is specifically designed to handle the challenges associated with sparse and weak annotations, the common problem we are facing in medical image analysis.

At its core, the connected U-Net relies on the foundational principles of the traditional U-Net architecture, known for its encoding-decoding structure. This structure is great at capturing complex contextual information while ensuring precise localization, crucial for accurate segmentation tasks. What sets the connected U-Net apart is its enhanced connectivity. By introducing additional connections between various layers in the network, it ensures a more effective flow of information and gradients. This enhancement is vital for the network's ability to learn from limited and imprecise data, typical of weakly-supervised learning scenarios.

A key feature of this architecture is its method of feature concatenation and fusion across different levels. By integrating multi-scale features, the network gains a comprehensive understanding of the tumor's structure and context within the DCE-MRI images. This multi-scale feature fusion is particularly important for segmenting complex tumor structures, where different levels of contextual information are crucial.

The connected U-Net is designed to address the challenges posed by the sparse annotations. Through its deep learning architecture, it incorporates these limited annotations effectively, facilitating better feature representation and learning, something that is crucial for handling the uncertainties inherent in weakly-supervised learning environments.

Furthermore, the architecture is intricately tied to the paper's SimPLe strategy, particularly in the fine-tuning phase. Post initial training with pseudo-masks, the connected U-Net is fine-tuned using the similarity-aware propagation learning approach. This phase allows the network to refine its segmentation capabilities by evaluating the similarity between labeled and unlabeled voxels previously defined in the initial mask generated, thereby enhancing its understanding of tumor characteristics.

```
class Unet(nn.Module):
    Abner
    def __init__(self, in_channels, out_channels):
        super(Unet, self).__init__()
        self.conv_blocks_context = nn.ModuleList([
            DoubleConv(in_channels_1=in_channels, out_channels_1=16, kernel_size_1=[1, 3, 3], stride_1=(1, 1, 1),
                       padding_1=[0, 1, 1],
                       in_channels_2=16, out_channels_2=16, kernel_size_2=[1, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[0, 1, 1]),
            DoubleConv(in_channels_1=16, out_channels_1=32, kernel_size_1=[3, 3, 3], stride_1=[2, 2, 2],
                       padding_1=[1, 1, 1],
                       in_channels_2=32, out_channels_2=32, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=32, out_channels_1=64, kernel_size_1=[3, 3, 3], stride_1=[2, 2, 2],
                       padding_1=[1, 1, 1],
                       in_channels_2=64, out_channels_2=64, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=64, out_channels_1=128, kernel_size_1=[3, 3, 3], stride_1=[2, 2, 2],
                       padding_1=[1, 1, 1],
                       in_channels_2=128, out_channels_2=128, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=128, out_channels_1=256, kernel_size_1=[3, 3, 3], stride_1=[2, 2, 2],
                       padding_1=[1, 1, 1],
                       in_channels_2=256, out_channels_2=256, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
        ])


        self.conv_blocks_localization = nn.ModuleList([
            DoubleConv(in_channels_1=256, out_channels_1=128, kernel_size_1=[3, 3, 3], stride_1=(1, 1, 1),
                       padding_1=[1, 1, 1],
                       in_channels_2=128, out_channels_2=128, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=128, out_channels_1=64, kernel_size_1=[3, 3, 3], stride_1=(1, 1, 1),
                       padding_1=[1, 1, 1],
                       in_channels_2=64, out_channels_2=64, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=64, out_channels_1=32, kernel_size_1=[3, 3, 3], stride_1=(1, 1, 1),
                       padding_1=[1, 1, 1],
                       in_channels_2=32, out_channels_2=32, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
            DoubleConv(in_channels_1=32, out_channels_1=16, kernel_size_1=[3, 3, 3], stride_1=(1, 1, 1),
                       padding_1=[1, 1, 1],
                       in_channels_2=16, out_channels_2=16, kernel_size_2=[3, 3, 3], stride_2=(1, 1, 1),
                       padding_2=[1, 1, 1]),
        ])

        self.tu = nn.ModuleList([
            nn.ConvTranspose3d( in_channels: 256,  out_channels: 128, kernel_size=(2, 2, 2), stride=(2, 2, 2), bias=False),
            nn.ConvTranspose3d( in_channels: 128,  out_channels: 64, kernel_size=(2, 2, 2), stride=(2, 2, 2), bias=False),
            nn.ConvTranspose3d( in_channels: 64,  out_channels: 32, kernel_size=(2, 2, 2), stride=(2, 2, 2), bias=False),
            nn.ConvTranspose3d( in_channels: 32,  out_channels: 16, kernel_size=(2, 2, 2), stride=(2, 2, 2), bias=False)
            ])

        self.seg = nn.Conv3d( in_channels: 16, out_channels, kernel_size=(1, 1, 1), stride=(1, 1, 1), bias=False)
```

```
    def forward(self, x):
        encoders_features = []
        for i in range(len(self.conv_blocks_context)):
            x = self.conv_blocks_context[i](x)
            encoders_features.insert( _index: 0, x)

        encoders_features = encoders_features[1:]
        for u in range(len(self.tu)):
            x = self.tu[u](x)
            x = torch.cat( tensors: (x, encoders_features[u]), dim=1)
            x = self.conv_blocks_localization[u](x)
        output = self.seg(x)
        return output, x

10 usages  Abner
class DoubleConv(nn.Module):
    Abner
    def __init__(self, in_channels_1, out_channels_1, kernel_size_1, stride_1, padding_1,
                 in_channels_2, out_channels_2, kernel_size_2, stride_2, padding_2):
        super(DoubleConv, self).__init__()
        self.blocks = nn.ModuleList([
            nn.Sequential(OrderedDict([
                ('conv', nn.Conv3d(in_channels_1, out_channels_1, kernel_size_1, stride_1, padding_1)),
                # ('dropout', nn.Dropout3d(p=0.5, inplace=True)),
                ('lrule', nn.LeakyReLU(negative_slope=0.01, inplace=True)),
                ('instnorm',
                 nn.InstanceNorm3d(out_channels_1, eps=1e-05, momentum=0.1, affine=True, track_running_stats=False))
            ])),
            nn.Sequential(OrderedDict([
                ('conv', nn.Conv3d(in_channels_2, out_channels_2, kernel_size_2, stride_2, padding_2)),
                # ('dropout', nn.Dropout3d(p=0.5, inplace=True)),
                ('lrule', nn.LeakyReLU(negative_slope=0.01, inplace=True)),
                ('instnorm',
                 nn.InstanceNorm3d(out_channels_2, eps=1e-05, momentum=0.1, affine=True, track_running_stats=False))
            ]))
        ])

    Abner
    def forward(self, x):
        for block in self.blocks:
            x = block(x)
        return x

if __name__ == '__main__':
    os.environ['CUDA_VISIBLE_DEVICES'] = '0'
    net = Unet( in_channels: 1,  out_channels: 2).cuda()
    out, _ = net(torch.randn(1, 1, 96, 128, 128).cuda())
    print(out.shape)
```

Figure 2: Code Snippet of the U-Net

5 The Training Pipeline

As discussed previously, the training pipeline for this breast cancer segmentation in employs a three stage process: train, fine-tune, and retrain. based on the Similarity-Aware Propagation Learning (SimPLe) strategy. The first stage begins by generating initial pseudo-masks from extreme points, which are processed and harmonized accordingly to meet the requirements of the network. In the second stage, the network is fine-tuned by propagating labels to unlabeled voxels based on their feature similarity to positive (tumor) voxels, refining the pseudo-masks for improved segmentation accuracy, and the final retraining stage uses these refined pseudo-masks to further enhance the network's ability to delineate tumor boundaries with greater precision.

5.1 First Stage

The first stage is the most crucial step of this approach, since it is where the preprocessing steps take place, which process and harmonizes the images so they can be fed to the pipeline to start the training process

5.1.1 Preprocessing

The preprocessing in the project plays an essential role in preparing the DCE-MRI images for the training process, implementing several steps. Firstly, the script begins with reorienting the MRI images to the Right-Anterior-Superior (RAS) coordinate system. This standardization ensures that all images have consistent orientation, something fundamental for accurate analysis and comparison across different scans.

Following the reorientation, the script performs image resampling. This process adjusts the resolution or voxel size of the images, to ensure uniformity in scale and size. This is a vital step for maintaining consistency in the features that the neural network will learn during training.

Once the images are reoriented, the script saves these modified images in a directory which the training step will access. This organizational step is essential for efficiently managing the preprocessed data, making it easily accessible for subsequent processing steps. Additionally, the script converts the MRI images into numpy arrays, a format that is particularly amenable to processing with machine learning algorithms. This conversion is a standard practice in preparing image data for deep learning models, allowing for efficient manipulation and analysis of the image data.

There are also other steps such as normalization and data augmentation, which take place in the very training process, and will be discussed there.This preprocessing is done to the data generated by the initial pseudo mask generation script.

```python
def f_img(tr_img_path):
    # reorient to ras
    tr_img = sitk.ReadImage(tr_img_path)
    direction = np.array(tr_img.GetDirection())
    origin = np.array(tr_img.GetOrigin())
    spacing = np.array(tr_img.GetSpacing())[[2, 1, 0]]
    size = np.array(tr_img.GetSize())[[2, 1, 0]]

    tr_img = nib.load(tr_img_path)
    origin_affine = tr_img.affine
    ras_img = tr_img.as_reoriented(io_orientation(tr_img.affine))
    ras_img_path = tr_img_path.replace('raw_data', 'resample_1.0X0.6X0.6').replace('.nii.gz', '_reorientation.nii.gz')
    nib.save(ras_img, ras_img_path)

    tr_img = sitk.ReadImage(ras_img_path)
    tr_img_array = sitk.GetArrayFromImage(tr_img)

    # img resample
    new_size = np.floor(((spacing / np.array(target_spacing)).astype(float) * size)).astype(int)
    img_zoom = transform.resize(tr_img_array, new_size, order=3, preserve_range=True)

    np.savez_compressed(tr_img_path.replace('raw_data', 'resample_1.0X0.6X0.6').replace('.nii.gz', '.npz'),
                        data=img_zoom,
                        origin_spacing=spacing,
                        origin_size=size,
                        origin_direction=direction,
                        origin_affine=origin_affine,
                        origin=origin)
    os.remove(ras_img_path)
    print(tr_img_path, size, new_size, spacing, target_spacing)
```

Figure 3: Code snippet for MRI preprocessing

```python
def f_seg(tr_seg_path):
    # reorient to ras
    tr_seg = nib.load(tr_seg_path)
    ras_seg = tr_seg.as_reoriented(io_orientation(tr_seg.affine))
    ras_seg_path = tr_seg_path.replace('noisy_label', 'noisy_label_resample_1.0X0.6X0.6').replace('.nii.gz',
                                                                                                  '_reorientation.nii.gz')
    nib.save(ras_seg, ras_seg_path)

    tr_seg = sitk.ReadImage(ras_seg_path)
    spacing = np.array(tr_seg.GetSpacing())[[2, 1, 0]]
    size = np.array(tr_seg.GetSize())[[2, 1, 0]]
    tr_seg_array = sitk.GetArrayFromImage(tr_seg)

    # seg resample
    new_size = np.floor(((spacing / np.array(target_spacing)).astype(float) * size)).astype(int)
    seg_zoom = transform.resize(tr_seg_array.astype(float), new_size,  order: 0, mode="edge", clip=True,
                                anti_aliasing=False).astype(np.uint8)

    np.savez_compressed(
        tr_seg_path.replace('noisy_label', 'noisy_label_resample_1.0X0.6X0.6').replace('.nii.gz', '.npz'),
        seg=seg_zoom)
    os.remove(ras_seg_path)
    print(tr_seg_path, size, new_size, spacing, target_spacing)
```

Figure 4: Code snippet for Initial Segmentation Mask preprocessing

5.1.2 Training

The training process is performed by 3 scripts inside the stage1 directory: train.py, data.py and data_transform.py. The data.py script is responsible for loading the images along with their corresponding initial pseudo-masks. In addition to this, the data_transform.py script plays a pivotal role in applying essential transformations and augmentations. It includes the normalization of image data to standardize intensity values and implements random cropping to focus on regions of interest, particularly on tumor areas. These transformations are key in enhancing the model's ability to generalize from the training data for improving the segmentation process. Additionally, the script ensures uniformity in image dimensions through padding, especially for images smaller than the desired crop size, maintaining standard input sizes for the neural network.

Following these preprocessing steps, the train.py script takes over. It sets up the connected U-Net architecture and the loss functions, including partial cross-entropy and contrastive losses, managing the training epochs. The network, leveraging these preparations, learns to segment the cancerous regions based on the annotations in the pseudo-masks. This stage is designed to prevent overfitting to the inaccuracies present in the initial pseudo-masks and to retain the network's ability to generalize.

The output of this initial training stage is a preliminary segmentation of the breast cancer regions. While not the final segmentation, it provides a basis for the subsequent stages of the SimPLe approach, where fine-tuning and retraining will refine the segmentation accuracy during the next 2 stages. The effectiveness of this first stage is crucial, as it determines the baseline from which the network will further enhance its understanding of the segmentation task in the later stages of fine-tuning and retraining.

```python
def main():
    reproduce(args.seed)
    logging.basicConfig(filename=os.path.join(args.exp_name, 'log.txt'), level=logging.INFO,
                        format='[%(asctime)s.%(msecs)03d] %(message)s', datefmt='%H:%M:%S')
    logging.getLogger().addHandler(logging.StreamHandler(sys.stdout))
    logging.info(str(args))

    net = Unet(1, 2).cuda()

    if torch.cuda.is_available():
        device = torch.device("cuda")
        print("Using GPU:", torch.cuda.get_device_name(0))
    else:
        device = torch.device("cpu")
        print("CUDA is not available. Using CPU.")

    net.load_state_dict(torch.load(args.pretrain))

    train_data_list = os.listdir('../data/zym/workspace/noisy/noisy_label_resample_1.0X0.6X0.6')

    transform_fg_train = transforms.Compose([Norm(),
                                             RandomCrop(args.patch_size, seed: 1, fg_rate: 1.),
                                             ToTensor(0)])
    train_fg_dataset = BreastTumor(train_data_list, transform=transform_fg_train)
    fg_dataloader = DataLoader(train_fg_dataset,
                               batch_size=args.batch_size,
                               shuffle=True,
                               num_workers=args.num_workers,
                               pin_memory=False,
                               worker_init_fn=worker_init_fn,
                               drop_last=False)

    transform_bg_train = transforms.Compose([Norm(),
                                             RandomCrop(args.patch_size, seed: 0, fg_rate: 1.),
                                             ToTensor(0)])
    train_bg_dataset = BreastTumor(train_data_list, transform=transform_bg_train)
    bg_dataloader = DataLoader(train_bg_dataset,
                               batch_size=args.batch_size,
                               shuffle=True,
                               num_workers=args.num_workers,
                               pin_memory=False,
                               worker_init_fn=worker_init_fn,
                               drop_last=False)

    optimizer = optim.SGD(net.parameters(), lr=args.base_lr, momentum=0.99, weight_decay=1e-4, nesterov=True)
    writer = SummaryWriter(os.path.join(args.exp_name, 'tbx'))

    CE = torch.nn.CrossEntropyLoss(ignore_index=2)

    iter_num = 0
    max_iter = int(args.max_iter)
    for epoch_num in tqdm(range(math.ceil(max_iter / len(fg_dataloader))), ncols=70):
        epoch_num = epoch_num + 1

        fg_prefetcher = data_prefetcher(fg_dataloader)
        bg_prefetcher = data_prefetcher(bg_dataloader)
        fg_sample = fg_prefetcher.next()
        bg_sample = bg_prefetcher.next()
        while fg_sample is not None and bg_sample is not None:
            iter_num = iter_num + 1
            net.train()

            fg_img, fg_seg = fg_sample['image'], fg_sample['label']
            bg_img, bg_seg = bg_sample['image'], bg_sample['label']
            fg_img, fg_seg = fg_img.cuda(), fg_seg.cuda()
            bg_img, bg_seg = bg_img.cuda(), bg_seg.cuda()

            fg_outs, fg_fea = net(fg_img)
            bg_outs, _ = net(bg_img)
```

```
        fea = fg_fea.detach().permute(0, 2, 3, 4, 1).contiguous()

        unlabel_mask = torch.zeros_like(fg_seg, dtype=torch.bool)
        unlabel_mask[fg_seg == 2] = 1
        unlabel_samples = fea[unlabel_mask]

        positive_mask = torch.zeros_like(fg_seg, dtype=torch.bool)
        positive_mask[fg_seg == 1] = 1
        positive_samples = fea[positive_mask]
        positive_samples = positive_samples[torch.randperm(positive_samples.size(0))]
        positive_samples = positive_samples[:args.N]

        positive_sim = F.cosine_similarity(positive_samples[..., None, :, :], unlabel_samples[..., :, None, :], dim=-1)
        positive_pseudo = torch.gt(torch.gt(positive_sim, args.lambda_).sum(1), args.alpha_ * args.N)
        pseudo = 2. * np.ones(unlabel_samples.shape[0])
        pseudo = torch.LongTensor(pseudo).cuda()
        pseudo[positive_pseudo == 1] = 1

        pseudo_loss = args.omega_ * CE(fg_outs.permute(0, 2, 3, 4, 1).contiguous()[unlabel_mask], pseudo)
        loss = CE(fg_outs, fg_seg) + CE(bg_outs, bg_seg) + pseudo_loss

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        writer.add_scalar('loss/L_supervised', loss.item(), iter_num)
        writer.add_scalar('loss/L_pseudo', pseudo_loss.item(), iter_num)
        writer.add_scalar('ratio/fg_pseudo', (positive_pseudo == 1).sum() / unlabel_mask.sum(), iter_num)

        if iter_num % 10 == 0:
            image = fg_img[0, 0:1, 10:51:10, :, :].permute(1, 0, 2, 3).repeat(1, 3, 1, 1)
            grid_image = make_grid(image, nrow: 5, normalize=True)
            writer.add_image('train/Image', grid_image, iter_num)

            outputs_soft = F.softmax(fg_outs, dim: 1)
            image = outputs_soft[0, 1:2, 10:51:10, :, :].permute(1, 0, 2, 3).repeat(1, 3, 1, 1)
            grid_image = make_grid(image, nrow: 5, normalize=False)
            writer.add_image('train/Predicted', grid_image, iter_num)

        fg_sample = fg_prefetcher.next()
        bg_sample = bg_prefetcher.next()

        lr_ = args.base_lr * (1 - iter_num / max_iter) ** 0.9
        for param_group in optimizer.param_groups:
            param_group['lr'] = lr_

        # save
        if iter_num % args.save_per_iter == 0:
            save_model_path = os.path.join(args.exp_name, f'iter_{iter_num}.pth')
            torch.save(net.state_dict(), save_model_path)

        if iter_num > max_iter:
            break
    if iter_num > max_iter:
        break

writer.close()

save_model_path = os.path.join(args.exp_name, f'iter_{max_iter}.pth')
torch.save(net.state_dict(), save_model_path)
```

Figure 5: Code Snippet for the first training stage

5.2 Second Stage

The second stage of the training process is dedicated to fine-tune the neural network. This stage refines the segmentation results obtained from the initial training phase, with the objective of enhancing the precision of the tumor segmentation by capitalizing on the previously learned features from the first stage. The network enters this phase equipped with the weights and insights it garnered previously, specifically the 200th iteration, a common practice in these type of deep learning models setting the stage for more advanced and refined learning.

This stage leverages on assessing the similarity of features between labeled (tumor) and unlabeled voxels. The network evaluates each unlabeled voxel, determining whether its features closely resemble those of the labeled tumor voxels. This assessment is crucial as it allows for the identification of unlabeled voxels that are likely part of the tumor region, based on their feature similarity. Following this assessment, the pseudo-masks, initially used for training, are updated to reflect a more accurate understanding of tumor boundaries. These revised pseudo-masks serve as a more refined guide for the network's training in this stage.

The fine-tuning process involves retraining the network with these updated pseudo-masks. This step is crucial as it helps the network to better understand and delineate the tumor regions, rectifying any inaccuracies or misclassifications that may have occurred in the initial training phase. The loss functions employed during this stage mirror those used initially (partial cross-entropy and contrastive losses), and are applied to these refined masks, ensuring the continued effectiveness of the learning process.

$$\mathcal{L}_{pce} = -\sum_{Y_{init}(\boldsymbol{k})=0} log(1 - f(X;\theta)(\boldsymbol{k})) - \sum_{Y_{init}(\boldsymbol{k})=1} log(f(X;\theta)(\boldsymbol{k})).$$

Figure 6: Partial Cross Entropy Loss

$$\mathcal{L}_{ctr} = -\frac{1}{2N-1} \sum_{\boldsymbol{k_n} \in \mathcal{N}(\boldsymbol{k})} log(1 - \sigma(sim(\mathcal{Z}(\boldsymbol{k}), \mathcal{Z}(\boldsymbol{k_n}))/\tau)) - \frac{1}{2N-1} \sum_{\boldsymbol{k_p} \in \mathcal{P}(\boldsymbol{k})} log(\sigma(sim(\mathcal{Z}(\boldsymbol{k}), \mathcal{Z}(\boldsymbol{k_p}))/\tau)),$$

Figure 7: Contrastive Loss

The output of this fine-tuning stage is a significantly enhanced segmentation of breast cancer regions. This improved segmentation is expected to be more accurate and closely aligned with the actual ground truth compared to the results from the first training stage.

The process of obtaining these new updated masks is done in the infer.py script. It loads the original dataset and utilizes a U-Net model preloaded with weights from the second stage training phase and performs inference to generate updated segmentation masks. These refined masks, more accurate than the initial pseudo-masks, are key outputs of this script and form the essential input for the subsequent third stage of training.

5.3 Third Stage

The third and final training stage focuses on the retraining of the neural network. This stage leverages the refined segmentation masks generated during the second stage's fine-tuning phase. These masks, as previously discussed, offer a more precise delineation of cancerous regions compared to the initial pseudo-masks, and are used in this third stage to further train and refine the network's segmentation capabilities. The retraining process involves applying the network to these updated masks, enabling it to adjust and correct any remaining inaccuracies in its understanding of the tumor boundaries. The same loss functions are used, such as partial cross-entropy and contrastive losses.

As per the paper, the outcome of this stage is a final, highly refined segmentation of breast cancer regions within the DCE-MRI images. The effectiveness of this training stage is reflected in the reported performance metrics. The method achieved a mean Dice coefficient of 81.20% and a Jaccard index of 70.01%, demonstrating a significant improvement in segmentation accuracy compared to the initial training stage, where the Dice coefficient was 69.39% and the Jaccard index was 57.36%. This shows the substantial enhancement in segmentation accuracy achieved through the complete SimPLe training process.

6 Our Problem

In this project, our approach has been to implement all the methodologies explained above with our own dataset. For this, we decided to access the code available alongside the paper and reverse engineer it to make it work with our own data. The need to do such reverse engineering is due to the lack of a code description from the authors, which has made this process quite more challenging than initially expected, since the original target of the project was to just run and test their untouched approach. From this, we expected to show the potential of these new avant-garde technologies and techniques and show the viability of learning transference with the most minimal amount of fine tuning possible. As we will see below, this has happened to be a major technical challenge, from which much has been learnt not only in terms of technological knowledge, but also design choices vital for the development of these new approaches.

6.1 The Dataset

The Duke-Breast-Cancer-MRI dataset is the one which was chosen for the task of implementing our own version of the SimPLe strategy. It represents a substantial and diverse collection of medical data, specifically focused on invasive breast cancer. This single-institutional, retrospective dataset encompasses records from 922 biopsy-confirmed patients, collected over a decade at Duke Hospital, and it stands out as a comprehensive resource for research in the field of breast cancer, particularly radiogenomics and outcomes prediction.

A key component of the dataset is the pre-operative dynamic contrast-enhanced (DCE)-MRI images, which makes it great for our goals since it is the same nature of the dataset used by the researchers. These images, acquired from PACS systems and de-identified for release through The Cancer Imaging Archive (TCIA), include axial breast MRI scans performed using 1.5T and 3T scanners in the prone position. The dataset provides a range of MRI sequences in DICOM format, including non-fat saturated T1-weighted sequences, fat-saturated gradient echo T1-weighted pre-contrast sequences, and typically three to four post-contrast sequences. These comprehensive imaging data are crucial for assessing the tumor morphology and other vital features of breast cancer.

In addition to the MRI images, the dataset includes meticulously annotated locations of lesions by radiologists, which are the equivalent to the extreme points used by the researchers and will be the ones we will use to generate our own initial pseudo masks. These points are stored in an Excel file called **“Annotation_Boxes”.** Moreover, the dataset features a set of 529 computer-extracted imaging features, encompassing size, shape, texture, and enhancement characteristics of both the tumor and surrounding tissue. This extensive collection of features, extracted using in-house software, offers a rich base for advanced imaging analysis. These complimentary data may be extremely useful for further fine-tuning or even enrichment for the process, but due to the limitations of both the project and the subject, will be overseen.

The dataset's rigorous annotation process involved two distinct procedures carried out by panels of fellowship-trained radiologists. For a subset of patients, the radiologists annotated up to five lesions per study, focusing on areas of mass and non-mass enhancement. For the remaining patients, the procedure was slightly modified, with radiologists annotating the largest biopsied lesion, based on provided biopsy locations. The high level of precision and quality in these annotations is critical for the reliability and applicability of the dataset in various studies.

This dataset has been widely used and validated in numerous research papers and studies, particularly focusing on radiogenomics, machine learning applications in breast cancer, and predictive modeling. These include assessments of feature variability due to MRI protocol changes, evaluations of inter-reader feature stability, and validations in predicting tumor subtypes, receptor status, and treatment responses. Such extensive research utilization underscores the dataset's value and impact in the field of breast cancer research.

Aspect	SimPLe Dataset	Duke Breast Cancer
Size	206 DCE-MRI Scans	922 DCE-MRI Scans
MRI Scanner Type	1.5T MRI Scanner	Both 1.5T and 3T MRI Scanners
Image Resolutions	$0.379 \times 0.379 \times 1.700$ mm³ and $0.511 \times 0.511 \times 1.000$ mm³	Varied Resolutions
Annotation Style	Extreme Points located on the edges of the tumors, stored as data in the images	Extreme points creating a boundary box outside the area of the tumor, stored in Excel file

6.2 Our Initial Preprocessing

One of our key preprocessing decisions was to focus exclusively on the first phase of Dynamic Contrast-Enhanced (DCE) MRI, specifically the 'ph1ax' images. This decision was driven by a need for simplicity and consistency in our data processing pipeline. By concentrating on this initial phase of DCE-MRI, we aimed to maintain uniformity in the type of imaging data being analyzed. This approach helped in streamlining the data processing and ensured that the characteristics of the images were consistent across all patients in the dataset.

Exclusion of Non-3D Images: Something else we decided to implement was to exclude of all non-3D images from our dataset. Given that the methodology we are following in this approach is designed to to work exclusively with 3D images. Therefore, it was essential to filter out any 2D or non-volumetric images. This helped preventing potential errors and complications during the subsequent stages of mask generation and training.

Cropping Based on Extreme Points: Initially, we also considered cropping every MRI scan to include only the slices present in the extreme points Excel file. This approach was intended to focus the analysis specifically on the regions of interest highlighted by the extreme points and was mainly driven by the need of discarding useless data that occupied space and made the model more complex. However, this was later discarded in the final stages of implementation. The reason for discarding it was due to the added complexity it introduced in later stages of the project. We realized that this cropping process complicated the preprocessing, initial mask generation, and training phases. It added an additional layer of complexity in aligning the cropped images with the corresponding masks and ensuring that the training process was effective.

7 Key Differences in Initial Pseudo Mask

There was a necessity of implementing a new script that handled the initial pseudo masks generation. This was due to some challenges posed the way our dataset stored the extreme points delimiting the tumorous areas, especially when compared to the dataset used by the research group. Unlike the dataset from the researchers, where radiologists annotated extreme points directly on the tumor edges within the MRI images, in our dataset, these points were placed to form a rectangular box around the tumor area. However, critically, they did not touch the tumor itself. This difference in annotation style has resulted to be catastrophic, as it impeded the use of their methods, like the random walker algorithm, which are reliant on the extreme points marking the tumor boundaries directly. Consequently, this presented a substantial obstacle in generating accurate pseudo-masks for the tumor regions, as traditional methodologies were not equipped to handle extreme points located away from the tumor edges.

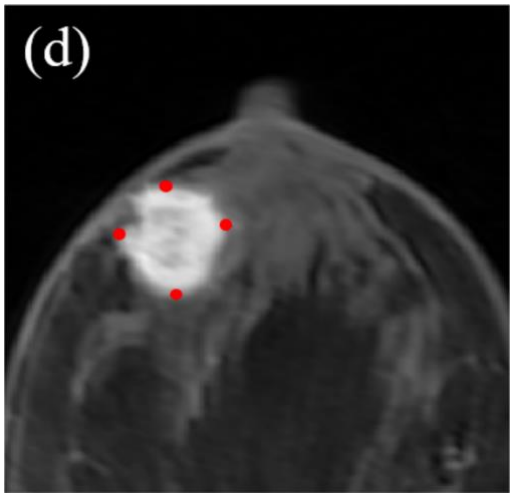

Figure 9: Extreme Points in Research Paper

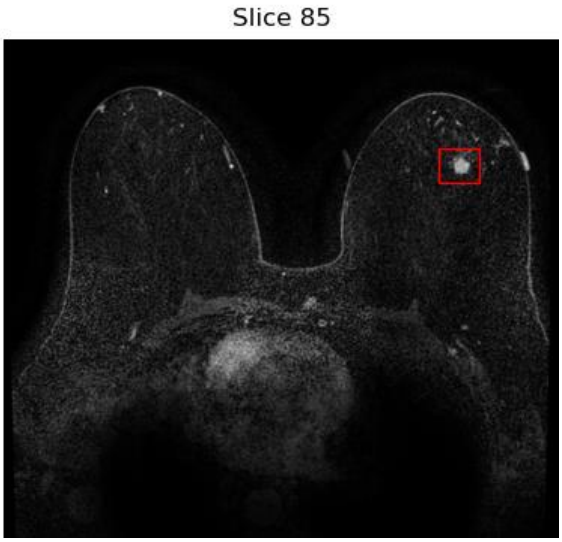

Figure 8: Extreme Points in our dataset: Patient 20

Another major challenge came from the different ways extreme points were stored and represented in the datasets. In contrast to the researcher´s dataset, where extreme points were embedded within the MRI files themselves, our dataset stored these points in an external Excel file as start_row, end_row, start_column, end_column, start_slice and end_slice numbers. This variation in data representation required a different approach to process and utilize these extreme points in segmentation tasks. The need of extracting the extreme points data from this Excel file instead of accessing it directly in the data introduced additional complexity that couldn't be properly addressed.

In response to these challenges, a new script was developed to handle this mask generation in a much simpler and hopefully effective away of generating the needed labels for the training process to work properly.

In this approach, we would focus on creating a binary mask based on extreme points that define a 3D rectangular box around the tumor area. However, we do not mark the entire box as the tumor area. Instead, these coordinates are adjusted slightly inward, concentrating on a smaller region within the box. This leads to a two-level mask creation: the outer region, which is the entire box minus a margin, is marked as unlabeled data and the inner core is labeled as tumor data. Then, the rest of the data outside the cube delimited by the extreme points is labeled as background, or non-tumor. This methodology generates the pseudo-mask with three distinct labels, just like the original script from the researchers. This way, the coordinates are adjusted inwardly and make us able to focus on a refined core area, providing an adjustment which is key to approximate accurately the actual tumor location and therefore refining the region of interest for segmentation.

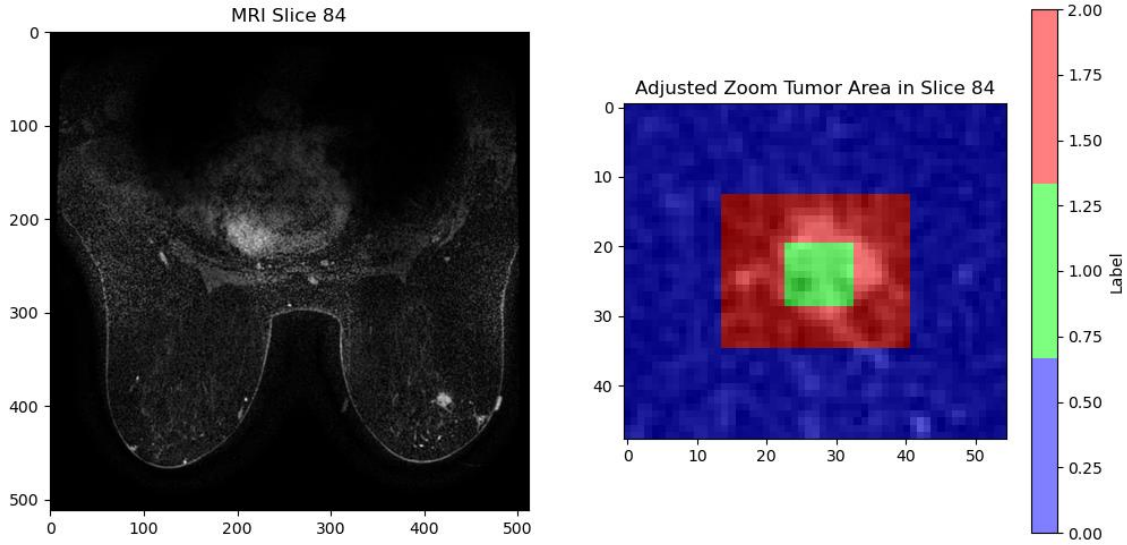

Figure 10: Original MRI (Left) and adjusted Tumor area (Right): Green: Tumor; Red: Unlabeled; Blue: Non-Tumor

7.1 From the first to the second version

Given the limitations and challenges posed by our dataset's unique annotation style, we explored alternative approaches to generate the initial pseudo masks. One of the methods initially considered was the scribble-based mask generation using the random walker algorithm. This method, heavily tailored to the specific dataset used in the original research, relies on extreme points accurately marking the boundaries of the tumor regions within the MRI images. In the original study, scribbles created from these points acted as seeds for the random walker algorithm, which refined the mask by propagating labels based on image gradients and scribble positions.

However, the random walker approach proved impractical for our dataset. Its performance is highly sensitive to the precise placement of extreme points, and since our extreme points only created a bounding box that did not directly touch the tumor edges, the algorithm struggled to generate meaningful segmentations. The algorithm was effectively too fine-tuned for the original dataset, where extreme points were inherently more precise, marking the tumor contours directly. Consequently, the method could not adapt to our broader, box-like annotations that encapsulate but do not contour the tumors, making it an unsuitable choice for our segmentation tasks.

To address these shortcomings, we adopted a Gaussian smoothing-based approach for mask refinement, which provided a more flexible and efficient solution for our needs. This method starts by defining a binary mask from the bounding box coordinates derived from the extreme points. Gaussian smoothing is then applied to this binary mask to soften the edges and produce rounded boundaries. A threshold is applied afterward to convert the smoothed mask back into a binary format, which effectively delineates the regions of interest. Unlike the random walker approach, this method does not depend on finely-tuned extreme points but rather provides a robust way to generate initial masks that can accommodate the variability in how different datasets represent tumor regions.

This change in approach proved advantageous in multiple ways. Firstly, it allowed us to create a pipeline that is less sensitive to variations in extreme point annotations and does not require meticulous adjustments to the scribbles for each case. Secondly, the computational efficiency of Gaussian smoothing, compared to the more iterative and resource-heavy random walker algorithm, enabled faster processing times while still delivering reasonably accurate segmentation masks. The resulting masks were well-suited for use in subsequent training stages, where initial accuracy can be further refined by deep learning models.

```python
# Path to the Excel file
excel_file_path = '../CASE STUDY PREPROCESS/Annotation_Boxes.xlsx'

# Read the extreme points from the Excel file
extreme_points_df = pd.read_excel(excel_file_path)
patient_extreme_points = extreme_points_df[extreme_points_df['Patient ID'] == 'Breast_MRI_020']

# Function to generate scribbles based on extreme points
 GNajeral
def generate_scribbles(img_array, extreme_points):
    # Extract extreme points
    start_slice, end_slice = extreme_points['Start Slice'], extreme_points['End Slice']
    start_row, end_row = extreme_points['Start Row'], extreme_points['End Row']
    start_col, end_col = extreme_points['Start Column'], extreme_points['End Column']

    # Initialize scribbles array
    scribbles = np.zeros_like(img_array)

    # Create scribbles
    scribbles[start_slice:end_slice + 1, start_row:end_row + 1, start_col:end_col + 1] = 1
    return scribbles


# Load the MRI scan
img_sitk = sitk.ReadImage('data/C1.nii.gz')
img_array = sitk.GetArrayFromImage(img_sitk)

# Normalize the image array
img_array_norm = (img_array - np.min(img_array)) / (np.max(img_array) - np.min(img_array))

# Extract extreme points for 'Breast_MRI_020'
extreme_points = patient_extreme_points.iloc[0]
start_slice, end_slice = extreme_points['Start Slice'], extreme_points['End Slice']
start_row, end_row = extreme_points['Start Row'], extreme_points['End Row']
start_col, end_col = extreme_points['Start Column'], extreme_points['End Column']

# Create scribbles based on extreme points
scribbles = np.zeros_like(img_array)
scribbles[start_slice+5:end_slice-5, start_row+5:end_row-5, start_col+5:end_col-5] = 1
scribble_sitk = sitk.GetImageFromArray(scribbles)
scribble_sitk.SetSpacing(img_sitk.GetSpacing())
scribble_sitk.SetDirection(img_sitk.GetDirection())
scribble_sitk.SetOrigin(img_sitk.GetOrigin())
sitk.WriteImage(scribble_sitk, 'data/scribble.nii.gz')

# Define spacing
spacing = img_sitk.GetSpacing()

# Adjust the random walker parameters
labels = random_walker(img_array_norm, scribbles, beta=30, mode='cg_mg', return_full_prob=False, spacing=spacing)

# Check the shape of the labels
print("Labels shape:", labels.shape)

# Convert the labels to a binary mask (tumor = 1, background = 0)
mask = (labels == 1).astype(np.uint8)

# Ensure the mask is 3D and matches the MRI scan dimensions
if len(mask.shape) == 4 and mask.shape[0] == 1:
    mask = mask.squeeze(0)  # Remove the first dimension if it's extraneous

print("Corrected mask shape:", mask.shape)

# Convert the mask to a SimpleITK image and copy information from the original MRI
mask_sitk = sitk.GetImageFromArray(mask)
mask_sitk.CopyInformation(img_sitk)

# Save the mask
sitk.WriteImage(mask_sitk, 'data/initial_mask_Breast_MRI_020.nii.gz')
```

Figure 11: Code Snippet for first version

```python
def generate_modified_mri(mri_scan_path, extreme_points, sigma=3, threshold=0.5):
    # Read the MRI scan
    img_sitk = sitk.ReadImage(mri_scan_path)
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Define bounding box and create a binary mask
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]
    mask = np.zeros_like(img_array)
    mask[start_slice + 5:end_slice, start_row + 2:end_row - 2, start_col + 2:end_col - 2] = 1

    # Apply Gaussian smoothing to the mask for rounded borders
    smoothed_mask = gaussian_filter(mask, sigma=sigma)

    # Threshold the smoothed mask to retain distinct boundaries
    smoothed_mask = (smoothed_mask >= threshold).astype(mask.dtype)

    # Create a modified MRI array using the smoothed mask
    modified_img_array = img_array * smoothed_mask

    # Convert the modified array back to a SimpleITK image
    modified_img_sitk = sitk.GetImageFromArray(modified_img_array)
    modified_img_sitk.CopyInformation(img_sitk)

    return modified_img_sitk


# Read extreme points from the Excel file and generate the mask
excel_file_path = '../CASE STUDY PREPROCESS/Annotation_Boxes.xlsx'
mri_scan_path = 'data/C1.nii.gz'


annotations = pd.read_excel(excel_file_path)
patient_data = annotations[annotations['Patient ID'] == 'Breast_MRI_020']
extreme_points = [
    (patient_data['Start Slice'].iloc[0], patient_data['Start Row'].iloc[0], patient_data['Start Column'].iloc[0]),
    (patient_data['End Slice'].iloc[0], patient_data['End Row'].iloc[0], patient_data['End Column'].iloc[0])
]

# Generate the modified MRI
modified_img_sitk = generate_modified_mri(mri_scan_path, extreme_points, sigma=0.5, threshold=0.5)

# Save the modified MRI as a NIFTI file
sitk.WriteImage(modified_img_sitk, 'data/init_mask.nii.gz')
```

Figure 12: Code Snippet for second version

7.2 Third Version

Building on the refinements made in the second version, a third iteration of the pseudo-mask generation script was developed to address some of the limitations observed in the earlier approach. While the second version focused on creating a binary mask refined through Gaussian smoothing, this third version introduces a more nuanced approach by incorporating multi-level thresholding. This enhancement allows for the generation of a pseudo-mask that differentiates between tumor regions, background, and areas of uncertainty, thus providing a more detailed and flexible segmentation output.

In this third version, the process begins similarly to the previous approach by reading the MRI scan and generating a binary mask based on the bounding box defined by the extreme points. The mask is then smoothed using Gaussian filtering, a step carried over from the second version to ensure smoother transitions along the tumor boundaries. However, rather than applying a single threshold to produce a binary mask, this version introduces dual thresholds: a high threshold to identify tumor regions and a low threshold to determine background areas. The space between these thresholds is used to label uncertain or ambiguous regions as "unlabeled," adding a third dimension to the segmentation output.

This multi-level thresholding technique was thought to represent a significant advance over the binary approach. By allowing the mask to capture areas that fall between clear-cut tumor and non-tumor classifications, the pseudo-mask would be able to better represent the complexities of the tumor region, especially in cases where the tumor boundaries are not well-defined. This additional layer of segmentation detail was particularly useful in preparing data for more advanced modeling techniques, which may benefit from distinguishing between confidently labeled regions and those that require further scrutiny.

Another key enhancement in the third version was the introduction of a visualization tool, which was absent in the earlier iterations. This tool would enable the user to view specific slices of the MRI scan alongside the corresponding pseudo-mask, with an option to zoom into the tumor region. This visual feedback were crucial for evaluating the quality of the generated masks, providing immediate insights into how well the segmentation process is capturing the tumor and highlighting areas that may need further adjustment.

This third version represented a more sophisticated and user-friendly approach to pseudo-mask generation. The dual-thresholding method offers greater flexibility and accuracy in segmentation, while the visualization tool enhances the usability and evaluation of the masks. These improvements made this third version particularly well-suited for scenarios that demand detailed and adaptable segmentation, allowing for more precise modeling and analysis in subsequent stages of the research.

```python
def generate_pseudo_mask(mri_scan_path, extreme_points, sigma=3, high_threshold=0.7, low_threshold=0.3):
    # Read the MRI scan
    img_sitk = sitk.ReadImage(mri_scan_path)
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Create a binary mask from extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]
    mask = np.zeros_like(img_array)
    mask[start_slice + 5:end_slice, start_row + 2:end_row - 2, start_col + 2:end_col - 2] = 1

    # Apply Gaussian smoothing to the mask
    smoothed_mask = gaussian_filter(mask, sigma=sigma)

    # Create pseudo mask with three labels
    pseudo_mask = np.zeros_like(img_array)
    pseudo_mask[smoothed_mask >= high_threshold] = 1  # Tumor
    pseudo_mask[smoothed_mask < low_threshold] = 0    # Background
    pseudo_mask[(smoothed_mask >= low_threshold) & (smoothed_mask < high_threshold)] = 2  # Unlabeled

    return pseudo_mask, img_sitk


# Read extreme points from the Excel file and generate the mask
excel_file_path = '../CASE STUDY PREPROCESS/Annotation_Boxes.xlsx'
mri_scan_path = 'data/C1.nii.gz'
patient_id = 'Breast_MRI_020'

annotations = pd.read_excel(excel_file_path)
patient_data = annotations[annotations['Patient ID'] == patient_id]
extreme_points = [
    (patient_data['Start Slice'].iloc[0], patient_data['Start Row'].iloc[0], patient_data['Start Column'].iloc[0]),
    (patient_data['End Slice'].iloc[0], patient_data['End Row'].iloc[0], patient_data['End Column'].iloc[0])
]

# Generate the pseudo mask
pseudo_mask, img_sitk = generate_pseudo_mask(mri_scan_path, extreme_points, sigma=0.5, high_threshold=0.5, low_threshold=0.2)

# Convert the pseudo mask back to a SimpleITK image
pseudo_mask_sitk = sitk.GetImageFromArray(pseudo_mask)
pseudo_mask_sitk.CopyInformation(img_sitk)

# Save the pseudo mask as a NIFTI file
output_mask_path = 'data/init_mask.nii.gz'
sitk.WriteImage(pseudo_mask_sitk, output_mask_path)


1 usage   ⚠ GNajeral
def visualize_mask_with_zoom(pseudo_mask, img_sitk, extreme_points, slice_number):
    """
    Visualize a specific slice of the pseudo mask and MRI scan with zoom into the tumor zone.

    :param pseudo_mask: 3D numpy array of the pseudo mask
    :param img_sitk: SimpleITK image of the MRI scan
    :param extreme_points: Tuple of extreme points (start and end) defining the tumor zone
    :param slice_number: The number of the slice to visualize
    """
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Extract extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]

    # Define ROI for zooming
    roi = (slice(start_row, end_row), slice(start_col, end_col))

    plt.figure(figsize=(12, 6))

    # Original MRI slice
    plt.subplot(1, 2, 1)
    plt.imshow(img_array[slice_number], cmap='gray')
    plt.title(f'MRI Slice {slice_number}')

    # Zoom into tumor zone on MRI slice
    plt.subplot(1, 2, 2)
    plt.imshow(img_array[slice_number][roi], cmap='gray')
    plt.imshow(pseudo_mask[slice_number][roi], cmap='viridis', alpha=0.5)
    plt.colorbar(label='Label')
    plt.title(f'Zoomed Tumor Area in Slice {slice_number}')

    plt.show()

start_slice, start_row, start_col = extreme_points[0]
end_slice, end_row, end_col = extreme_points[1]

# Example usage
# Assuming 'pseudo_mask' is your generated 3D mask array and you want to visualize slice number 50
visualize_mask_with_zoom(pseudo_mask, img_sitk, extreme_points, slice_number=round((start_slice+end_slice)/2))
```

Figure 13: Code Snippet for third version

7.3 Fourth version

After refining the pseudo-mask generation process with multi-level thresholding in the third version, I moved forward to develop a fourth version that aimed to enhance the segmentation precision and usability further. This version built upon the concept of a three-label pseudo-mask, but introduced a more sophisticated method to delineate the tumor region more effectively. The goal was to capture both the core of the tumor and its surrounding regions with greater clarity while still maintaining a structured and interpretable mask.

The fourth version began by generating a binary mask from the MRI scan using the extreme points to define the initial bounding box around the tumor area. Unlike the previous versions, this iteration involved the creation of a "full tumor mask" that defined a more extensive region around the tumor by slightly expanding the bounding box dimensions. From this expanded mask, a smaller "core" region within the tumor was defined by calculating a smaller cube at the center, using one-third of the dimensions of the larger bounding box. This core region aimed to represent the most confident area of the tumor, minimizing the ambiguity seen in the transition areas near the tumor's edges.

To implement this strategy, the mask generation process in this version first, created the full tumor mask as before, and then, using the calculated dimensions for the inner cube, it created a three-label pseudo-mask. This pseudo-mask labeled the core tumor area with the label '1', the surrounding region (excluding the core) as "unlabeled" with the label '2', and the rest of the background as '0'. By distinguishing the core of the tumor from its surroundings, this approach provided a clearer segmentation of the tumor, which could better support subsequent training steps and model development.

An important addition in this version was the enhancement of the visualization functionality. This iteration included both the capability to visualize a specific slice of the pseudo mask and MRI scan and allowed for adjustable zoom levels around the tumor zone. This functionality provided a more flexible way to inspect the segmentation, particularly helpful when trying to analyze or debug the effectiveness of the mask. I customized the color map for the visualization, with distinct colors for each label (background, tumor core, and unlabeled regions), improving the clarity of segmentation boundaries and making the visual interpretation more intuitive.

Another significant feature added was the ability to analyze the distribution of different labels within the pseudo-mask. This functionality provided insights into the proportion of voxels labeled as tumor, background, or unlabeled. This analysis was crucial for understanding how the mask generation parameters influenced the final segmentation and for ensuring that the generated masks had a balanced and meaningful distribution of labels. This step was particularly valuable in ensuring that the pseudo masks were adequately prepared for downstream tasks, such as model training, where the balance of labeled data could significantly impact performance.

```python
def generate_pseudo_mask(mri_scan_path, extreme_points):
    # Read the MRI scan
    img_sitk = sitk.ReadImage(mri_scan_path)
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Create a binary mask from extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]
    full_tumor_mask = np.zeros_like(img_array)
    full_tumor_mask[start_slice + 5:end_slice, start_row + 4:end_row - 4, start_col + 4:end_col - 4] = 1

    # Calculate the dimensions of the inner smaller cube (core of the tumor)
    core_slice_start = start_slice + (end_slice - start_slice) // 3
    core_slice_end = end_slice - (end_slice - start_slice) // 3
    core_row_start = start_row + (end_row - start_row) // 3
    core_row_end = end_row - (end_row - start_row) // 3
    core_col_start = start_col + (end_col - start_col) // 3
    core_col_end = end_col - (end_col - start_col) // 3

    # Create pseudo mask with three labels
    pseudo_mask = np.zeros_like(img_array)
    pseudo_mask[full_tumor_mask == 1] = 2  # Set exterior part as unlabeled
    pseudo_mask[core_slice_start:core_slice_end+1, core_row_start:core_row_end+1, core_col_start:core_col_end+1] = 1  # Tumor Core

    return pseudo_mask, img_sitk

1 usage  ± GNajeral
def visualize_mask_with_zoom(pseudo_mask, img_sitk, extreme_points, slice_number, zoom_factor=0.25):
    """
    Visualize a specific slice of the pseudo mask and MRI scan with adjusted zoom into the tumor zone.

    :param pseudo_mask: 3D numpy array of the pseudo mask
    :param img_sitk: SimpleITK image of the MRI scan
    :param extreme_points: Tuple of extreme points (start and end) defining the tumor zone
    :param slice_number: The number of the slice to visualize
    :param zoom_factor: Fraction to adjust the zoom level (default is 0.25)
    """
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Extract extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]

    # Adjust ROI for zooming out
    zoomed_start_row = max(0, start_row - int((end_row - start_row) * zoom_factor))
    zoomed_end_row = min(img_array.shape[1], end_row + int((end_row - start_row) * zoom_factor))
    zoomed_start_col = max(0, start_col - int((end_col - start_col) * zoom_factor))
    zoomed_end_col = min(img_array.shape[2], end_col + int((end_col - start_col) * zoom_factor))
    roi = (slice(zoomed_start_row, zoomed_end_row), slice(zoomed_start_col, zoomed_end_col))

    plt.figure(figsize=(12, 6))

    # Original MRI slice
    plt.subplot(1, 2, 1)
    plt.imshow(img_array[slice_number], cmap='gray')
    plt.title(f'MRI Slice {slice_number}')

    # Define a custom color map for the pseudo mask
    colors = [(0, 0, 1), (0, 1, 0), (1, 0, 0)]  # Blue for background, Green for unlabeled, Red for tumor
    cmap = matplotlib.colors.LinearSegmentedColormap.from_list('my_cmap', colors, N=3)

    # Zoom into tumor zone on MRI slice
    plt.subplot(1, 2, 2)
    plt.imshow(img_array[slice_number][roi], cmap='gray')
    plt.imshow(pseudo_mask[slice_number][roi], cmap=cmap, alpha=0.5)
    plt.colorbar(label='Label')
    plt.title(f'Adjusted Zoom Tumor Area in Slice {slice_number}')

    plt.show()

# Read extreme points from the Excel file and generate the mask
excel_file_path = '../CASE STUDY PREPROCESS/Annotation_Boxes.xlsx'
mri_scan_path = 'data/C1.nii.gz'
patient_id = 'Breast_MRI_020'

annotations = pd.read_excel(excel_file_path)
patient_data = annotations[annotations['Patient ID'] == patient_id]
extreme_points = [
    (patient_data['Start Slice'].iloc[0], patient_data['Start Row'].iloc[0], patient_data['Start Column'].iloc[0]),
    (patient_data['End Slice'].iloc[0], patient_data['End Row'].iloc[0], patient_data['End Column'].iloc[0])
]

# Generate the pseudo mask
pseudo_mask, img_sitk = generate_pseudo_mask(mri_scan_path, extreme_points)

# Convert the pseudo mask back to a SimpleITK image
pseudo_mask_sitk = sitk.GetImageFromArray(pseudo_mask)
pseudo_mask_sitk.SetSpacing(img_sitk.GetSpacing())
pseudo_mask_sitk.SetDirection(img_sitk.GetDirection())
pseudo_mask_sitk.SetOrigin(img_sitk.GetOrigin())

# Save the pseudo mask as a NIFTI file
output_mask_path = 'data/init_mask.nii.gz'
sitk.WriteImage(pseudo_mask_sitk, output_mask_path)

start_slice, start_row, start_col = extreme_points[0]
end_slice, end_row, end_col = extreme_points[1]

# Visualize a specific slice with zoom into the tumor zone
slice_number = round((start_slice + end_slice) / 2)
visualize_mask_with_zoom(pseudo_mask, img_sitk, extreme_points,  slice_number: 83)


1 usage  ± GNajeral
def analyze_mask_distribution(pseudo_mask):
    """
    Analyze the distribution of different labels within the pseudo mask.
```

```python
def analyze_mask_distribution(pseudo_mask):
    """
    Analyze the distribution of different labels within the pseudo mask.

    :param pseudo_mask: 3D numpy array of the pseudo mask
    """
    unique, counts = np.unique(pseudo_mask, return_counts=True)
    distribution = dict(zip(unique, counts))

    # Print out the distribution
    for label, count in distribution.items():
        print(f"Label {label}: {count} voxels ({(count / pseudo_mask.size) * 100:.2f}% of total)")

# Usage example
analyze_mask_distribution(pseudo_mask)
```

Figure 14: Code Snippet for the fourth version

7.4 Fifth Version

In progressing towards a more automated and scalable solution for pseudo-mask generation, I developed a fifth version of the code that expanded on the previous iterations by introducing enhanced batch processing capabilities. This version built on the core ideas of creating pseudo-masks with multiple labels and refined visualizations but shifted focus toward enabling the generation of masks for multiple patients in a single run. This improvement was motivated by the need to handle larger datasets more efficiently and reduce manual intervention during the preprocessing phase.

The fifth version maintained the fundamental structure of generating a binary mask from MRI scans using extreme points to define the tumor region. As in the fourth version, I created a full tumor mask by expanding the bounding box slightly inward to avoid the uncertainty at the edges. The smaller core region was again calculated by defining a cube within the central third of the tumor volume, creating a pseudo-mask with three distinct labels: the tumor core, the unlabeled surrounding area, and the background. However, unlike the previous versions, the parameters for creating the core and full masks were further refined, with more precise adjustments to the bounding box dimensions to enhance the accuracy of the core and periphery delineation.

A key advancement in this version was the ability to process multiple patients' MRI scans in a single execution. I implemented a new function, generate_masks_for_patients, that took a list of patient IDs and automatically located their corresponding MRI scans and extreme points annotations. This function streamlined the mask generation process by dynamically generating and saving the pseudo masks for each patient, without the need for manually specifying paths or adjusting parameters for each run. This automation allowed for the first time to prepare a fully ready batch of patients to start preprocessing and training with.

Moreover, I incorporated functionality to automatically locate and copy the relevant MRI scans into a designated output directory, organizing the generated masks and corresponding images in a structured and accessible manner. This automated file management helped in maintaining a clear data pipeline and facilitated easier review and access for subsequent analysis and modeling.

In addition to enhancing the automation and scalability, I retained and refined the visualization capabilities from the fourth version. The visualization function was adjusted to provide a more detailed view of the segmented regions, with configurable zoom levels to allow better inspection of the tumor areas. By using a custom color map for the pseudo mask, I ensured that the different labels were clearly distinguishable, making it easier to visually assess the quality of the segmentation and adjust the parameters accordingly.

To further support the analysis of the generated masks, I included a function, analyze_mask_distribution, that analyzed the distribution of different labels within the pseudo masks, which provided quantitative feedback on the segmentation quality by calculating the proportion of voxels assigned to each label, and it became crucial for understanding how well the parameters were set for different patients and ensuring that the generated masks were suitable for the subsequent training phases in the weakly-supervised learning framework

7.5 Sixth Version, the final one

The sixth and final version of the pseudo-mask generation script marked a culmination of the improvements made in the previous iterations, integrating all the refinements into a comprehensive and optimized solution. This version focused on further enhancing the flexibility, scalability, and accuracy of the mask generation process while ensuring that it was fully automated and ready for large-scale application. The primary goal was to create a robust pipeline that could seamlessly generate high-quality pseudo masks across multiple MRI scans with minimal manual intervention, ensuring consistency and reliability in the results.

Building upon the core structure of generating a binary mask from MRI scans using extreme points, this version continued the approach of defining both a "full tumor mask" and a smaller "core tumor" region. However, significant refinements were made in how the bounding box dimensions were adjusted and how the core was calculated. The full tumor mask was created by adjusting the bounding box slightly inward, avoiding the uncertainty at the tumor's edges. Meanwhile, the inner core region was defined by a more precise calculation, ensuring that the tumor core remained well-focused and accurately represented.

A key advancement in this final version was the further optimization of the batch processing capabilities. The generate_masks_for_patients function was enhanced to handle a wider range of input scenarios, making the script more robust against variations in data formats and patient directories. This function automatically located the MRI scans and extracted the extreme points for each patient, leveraging a refined search algorithm that ensured greater reliability and reduced the chance of errors during file retrieval. Additionally, I optimized the code for saving the generated masks, ensuring that the spatial metadata from the original MRI scans, such as spacing, direction, and origin, was correctly transferred to the pseudo masks. This meticulous attention to detail guaranteed that the generated masks were directly compatible with subsequent processing stages, something that was not ensured in the previous scripts.

Another improvement was the integration of better data management practices. The script now not only generated and saved pseudo masks for each patient but also maintained a well-organized output directory structure. MRI scans were automatically copied into a separate folder, allowing for easier access and review. This organization reduced clutter and improved the workflow´s efficiency, which, in addition with the automated file handling also facilitated smoother integration into more complex pipelines, where multiple scripts and processes would be chained together.

In terms of visualization and analysis, this version retained the powerful visualization capabilities developed in earlier versions but refined them to be more interactive and informative. The zoom functionality was further adjusted to allow finer control over the region of interest around the tumor, and the custom color map was retained to clearly distinguish between different segmentation label, making it easier to visually assess the quality of the segmentation and allowing for better parameter tuning.

Additionally, I maintained the analyze_mask_distribution function, which provided critical insights into the label distribution within the pseudo masks. By offering a breakdown of voxel counts for each label, this analysis ensured that the segmentation was balanced and accurately represented across different patients. This feedback loop was essential for verifying that the pseudo masks were suitable for use in the training stages, helping to debug the code when no usable masks were generated through the pipeline.

This final version resulted in a fully mature solution for the pseudo-mask generation, combining flexibility, automation, and precision into a cohesive framework. With this, I was able to finally correctly execute the training pipeline and get useful results out of it. Result of a thorough and iterative process of constant improvement, built upon the original initial mask generation from the SimPLe strategy, and rethought and redesigned to offer a more generalized way of processing that could be expanded to other datasets and not just being fine-tuned to the data it was originally designed to work upon.

```python
def generate_pseudo_mask(mri_scan_path, extreme_points):
    # Read the MRI scan
    img_sitk = sitk.ReadImage(mri_scan_path)
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Create a binary mask from extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]
    full_tumor_mask = np.zeros_like(img_array)
    start_row = start_row + 5
    start_col = start_col + 5
    end_row = end_row - 5
    end_col = end_col - 5
    full_tumor_mask[start_slice + 5:end_slice, start_row:end_row, start_col:end_col] = 1

    # Calculate the dimensions of the inner smaller cube (core of the tumor)
    core_slice_start = start_slice + (end_slice - start_slice) // 3
    core_slice_end = end_slice - (end_slice - start_slice) // 3
    core_row_start = start_row + (end_row - start_row) // 3
    core_row_end = end_row - (end_row - start_row) // 3
    core_col_start = start_col + (end_col - start_col) // 3
    core_col_end = end_col - (end_col - start_col) // 3

    # Create pseudo mask with three labels
    pseudo_mask = np.zeros_like(img_array)
    pseudo_mask[full_tumor_mask == 1] = 2   # Set exterior part as unlabeled
    pseudo_mask[core_slice_start:core_slice_end+1, core_row_start:core_row_end+1, core_col_start:core_col_end+1] = 1  # Tumor Core

    return pseudo_mask, img_sitk
```

Figure 15: Code Snippet for the Final Version of the Initial Mask Generator

```python
def visualize_mask_with_zoom(pseudo_mask, img_sitk, extreme_points, slice_number, zoom_factor=0.25):
    """
    Visualize a specific slice of the pseudo mask and MRI scan with adjusted zoom into the tumor zone.

    :param pseudo_mask: 3D numpy array of the pseudo mask
    :param img_sitk: SimpleITK image of the MRI scan
    :param extreme_points: Tuple of extreme points (start and end) defining the tumor zone
    :param slice_number: The number of the slice to visualize
    :param zoom_factor: Fraction to adjust the zoom level (default is 0.25)
    """
    img_array = sitk.GetArrayFromImage(img_sitk)

    # Extract extreme points
    start_slice, start_row, start_col = extreme_points[0]
    end_slice, end_row, end_col = extreme_points[1]

    # Adjust ROI for zooming out
    zoomed_start_row = max(0, start_row - int((end_row - start_row) * zoom_factor))
    zoomed_end_row = min(img_array.shape[1], end_row + int((end_row - start_row) * zoom_factor))
    zoomed_start_col = max(0, start_col - int((end_col - start_col) * zoom_factor))
    zoomed_end_col = min(img_array.shape[2], end_col + int((end_col - start_col) * zoom_factor))
    roi = (slice(zoomed_start_row, zoomed_end_row), slice(zoomed_start_col, zoomed_end_col))

    plt.figure(figsize=(12, 6))

    # Original MRI slice
    plt.subplot( *args: 1, 2, 1)
    plt.imshow(img_array[slice_number], cmap='gray')
    plt.title(f'MRI Slice {slice_number}')

    # Define a custom color map for the pseudo mask
    colors = [(0, 0, 1), (0, 1, 0), (1, 0, 0)]  # Blue for background, Green for unlabeled, Red for tumor
    cmap = matplotlib.colors.LinearSegmentedColormap.from_list( name: 'my_cmap', colors, N=3)


    # Zoom into tumor zone on MRI slice
    plt.subplot( *args: 1, 2, 2)
    plt.imshow(img_array[slice_number][roi], cmap='gray')
    plt.imshow(pseudo_mask[slice_number][roi], cmap=cmap, alpha=0.5)
    plt.colorbar(label='Label')
    plt.title(f'Adjusted Zoom Tumor Area in Slice {slice_number}')

    plt.show()
```

Figure 16: Code Snippet for the Final Mask Visualizer

8 Training with this mask

In our local environment, we faced significant challenges while attempting to train beyond the second stage. This limitation was primarily encountered during the second stage of training and the subsequent inference phase. During the training of the second stage, some epochs threw warnings explicitly stating that there were not data different from NaN or Inf. This meant that after the first stage, we were feeding black images to the second stage network, which seemed to be related to some peculiarities in our dataset.

These errors suggest a misalignment between the training process and the specific features of our dataset, possibly due to the differences in extreme point annotations or how these points were integrated into the training process. This happened to be the case, since we saw that in some patients, the extreme points where correctly encapsulating the tumor area, while in others, these extreme points were covering the background of the MRI.

The patients used for the experiment were the following: 2, 10, 13, 15, 20, 22, 25 and 30. However, after meticulous examination, we discovered that for some patients in our Duke´s dataset, the data found in the different phases of the DCE MRI differed in not only number of slices, hence the duration of the scan, but also in data harmonization such as resolution and other features. Therefore, while for patient 20, the extreme points delimit the tumor area quite well, for patients such as patient 10, the extreme points where not covering it, specifically in the ph1ax phase of the DCE.

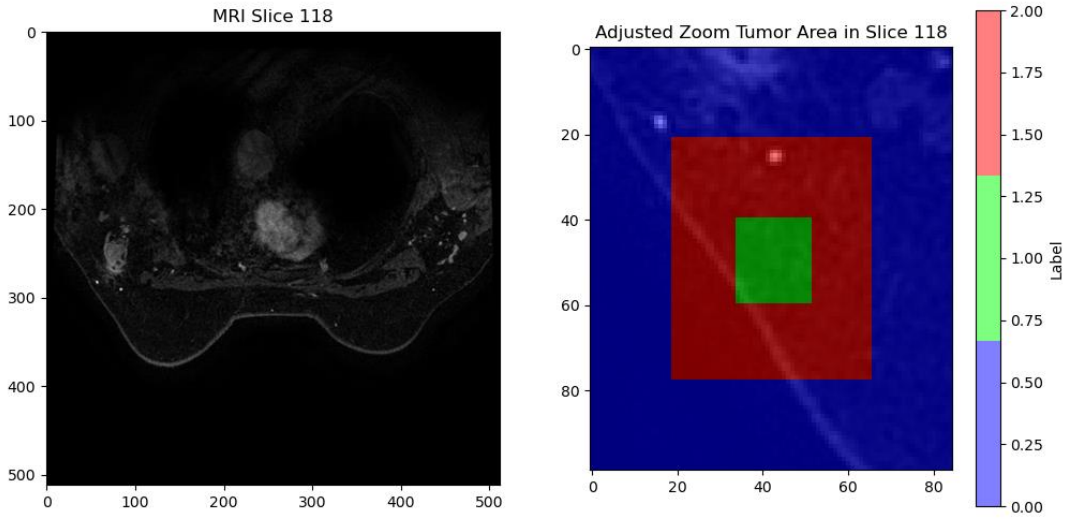

Figure 17: Patient 10 and its corresponding Mask. As we can see there is not data where the mask is placed

The challenges we faced indicate a need for a thorough investigation into the training code and its compatibility with our dataset. This would involve a more detailed examination to identify the exact root causes of these errors, whether they are originating from the training algorithms, data preprocessing methods,

or the integration of extreme points into the learning process due to the way data is structured in the dataset, something that would be left for later researchers to complete, given the actual constraints of the project.

Given these constraints, the training efforts were limited to successfully completing only the first stage and second stage, where the network was trained using initial pseudo-masks generated from our dataset and was able to produce some updated masks for selected patients, like the one below:

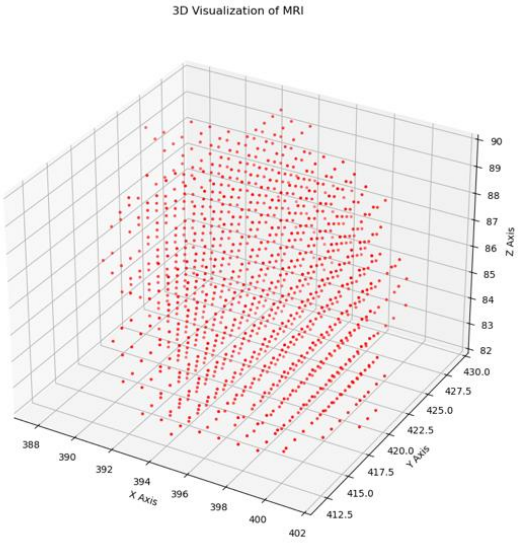

Figure 19: 3D Representation Updated Mask for patient 20.

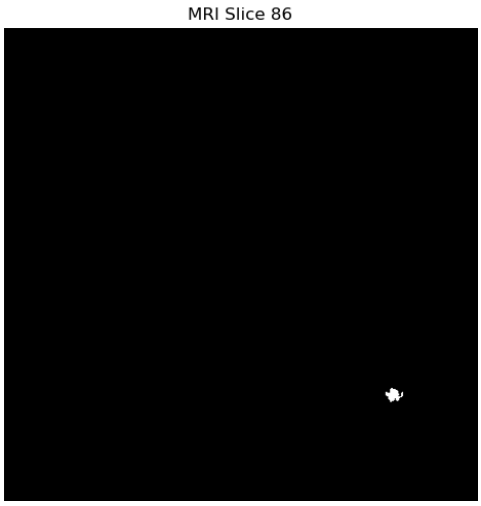

Figure 18: Updated Mask after stage 2 for patient 20.

9 Absence of Ground Truth Data

One significant challenge we have faced in the development of the project is the absence of ground truth data for evaluating the performance of our segmentation model. Without this data, assessing the accuracy and effectiveness of our model becomes very challenging and complex, unlike the research paper, that although it stipulates the methodology is based on weakly supervision, they still use ground truth data for both the training and validation, something that has needed to be excluded from the code scripts for our data to work. To address this issue, we must explore the possibility of using a thresholding methodology to automate the generation of ground truth data.

Thresholding in image processing is a technique that involves setting a specific intensity threshold in an image. Pixels above this threshold are classified as one category (e.g., tumor), and those below as another (e.g., non-tumor). This method could be particularly useful for segmenting these images based on the extreme points, creating binary images that approximate the presence of tumors.

By applying different thresholding algorithms to our scans, we could potentially generate a reliable set of ground truth data. This data would then serve as a reference to evaluate our model's segmentation accuracy. Although this approach may not be as precise as manually annotated ground truth, it offers a practical solution for validating our model in the absence of existing annotations.

10 CONCLUSIONS & LEARNING OUTCOMES

Reflecting on the initial mask generation process, it becomes clear that this step forms the backbone of the SimPLe strategy and indeed the entire project. From the beginning, our goal was to develop a method capable of extracting and refining full segmentation masks from extreme points in DCE-MRI breast cancer images. However, this is more than a mere technical challenge; it has resulted to be a difficult process that bridges data science with the fields of medical imaging, data diversity, and the always existing issue of data harmonization.

The process of generating these initial segmentation masks is not only the foundation but also the most critical stage of our methodology. Its significance lies in the fact that it directly influences the quality of training for the deep learning models. An effective initial pseudo-mask generation strategy sets the stage for the network's ability to learn meaningful features, which subsequently impacts the precision of cancer segmentation. As we encountered, any deficiency or mismatch in this initial stage disrupts the downstream processes, as evidenced by our challenges in adapting the original fine-tuned method to our dataset.

One of the major realizations from working on this project is the sheer complexity and variability inherent in medical data. The Duke-Breast-Cancer-MRI dataset, although rich and comprehensive, posed its unique challenges. Unlike the dataset used by the original researchers, where extreme points were directly annotated on tumor edges, our dataset used an external Excel file to define a bounding box around the tumor. This subtle difference was enough to require significant modifications in our approach to mask generation, teaching us the crucial lesson that minor differences in data annotation can have profound impacts on model performance and the ease of applying existing methodologies.

Working on refining this process has meant grappling with these real-world complications—understanding that developing a robust segmentation method goes beyond just applying a given algorithm. It demands an adaptable mindset that considers how medical professionals and data engineers can work with the most diverse data possible, harmonizing and refining it in a way that maximizes the efficiency and accuracy of the segmentation process.

This project has demonstrated that while technical solutions are necessary, they are often insufficient on their own. The biggest lesson learned from the disruption caused by our inability to use the fine-tuned method with our dataset was the importance of flexibility and creativity in research. Faced with a method that was not directly applicable, we had to iterate, test, and reimagine how we could adapt existing scripts to fit our needs, often reworking fundamental assumptions that underpin our approach.

Looking ahead, it is imperative to develop methods that are not only robust in their specific context but also generalizable across different types of datasets and varying medical conditions. This requires a deep understanding of both the technical and domain-specific nuances. As we have seen, even the initial step of creating these training masks must be adaptable to handle different forms of data annotation and representation. This adaptability is crucial for advancing medical image analysis to be both more efficient and more accurate, ultimately improving patient outcomes through better diagnostic tools.

This has underscored the foundational role that the initial segmentation mask generation process plays in the SimPLe strategy. It has been a journey of not only technical exploration but also strategic problem-solving, navigating the

complexities of medical imaging data and learning to adapt and innovate when faced with unexpected challenges. The experience reinforces the importance of a holistic approach to data science in medicine—one that blends technical rigor with practical flexibility and deep domain knowledge.

Moreover, Reflecting on the training process of our adaptation of the SimPLe strategy, it becomes evident that this phase was central to both the challenges and accomplishments of the project. The SimPLe strategy's design—divided into initial training, fine-tuning, and retraining—aimed to leverage a weakly-supervised learning approach to optimize segmentation accuracy with minimal manual annotation. While this method presents a promising advance in medical imaging, where annotations are labor-intensive and resource-consuming, our experience revealed both the potential and the limitations when applying this approach to a different dataset.

Our journey through the training process was marked by iterative refinement and adaptation. The initial training stage, based on pseudo-masks generated from extreme points, was crucial for establishing the network's baseline understanding of tumor segmentation. During this phase, the network began to learn fundamental features necessary for distinguishing cancerous regions, while still maintaining generalization capabilities. The goal was to ensure the model did not overfit to the initial inaccuracies inherent in the pseudo-masks, balancing specificity with generalizability. Although we did not have numerical performance metrics due to the absence of ground truth data, the qualitative results from this stage provided a strong foundation for further refinement.

The second stage, focused on fine-tuning with a similarity-aware propagation learning strategy, was particularly transformative. This phase demonstrated the architecture's ability to leverage feature similarity between labeled and unlabeled voxels, allowing the network to iteratively refine its understanding of tumor boundaries. Despite not completing the entire pipeline, we achieved meaningful results during this fine-tuning stage. Visual inspections of the segmentation masks showed that the architecture effectively refined the initial masks, producing outputs that were much closer to what would be expected in accurately segmented tumors. The refined masks closely approximated the actual tumor shapes, providing strong qualitative evidence that our approach was successful even without quantitative validation.

However, it is important to note that we were not able to run the complete pipeline, which includes the final retraining stage, due to technical challenges and specific characteristics of our dataset. Issues such as warnings of NaN or Inf values during the fine-tuning phase indicated a misalignment between the training process and the dataset's specific features. These challenges underscored the sensitivity of the training process to data variations, particularly in a weakly-supervised setting, and highlighted the need for careful adaptation when applying such methods to different datasets. The differences in annotation styles and data formats between the Duke-Breast-Cancer-MRI dataset and the dataset used in the original SimPLe study presented unique obstacles that required significant modifications, particularly in generating suitable initial segmentation masks.

Despite these setbacks and the inability to numerically quantify the model's performance, the qualitative results obtained in the first two stages were compelling. They confirmed that the SimPLe approach for breast cancer MRI segmentation could generalize beyond the dataset used in the original paper, adapting effectively to new data with a modified initial segmentation mask generator. The visual outcomes demonstrated that the architecture, through

fine-tuning its training process, was able to refine the initial segmentation masks and produce outputs that closely represented the actual tumor locations. This success, even without completing the final retraining stage, sets a powerful precedent for the SimPLe strategy's potential.

The training process revealed the importance of adaptability, both in terms of algorithmic design and practical implementation. Achieving high performance with weakly-supervised learning requires a deep understanding of data-specific challenges and the flexibility to refine methods in response to those challenges. This experience has been a testament to the necessity of a dynamic and adaptable approach in machine learning applications, especially in data-sensitive fields like medical imaging.

In conclusion, our exploration of the training process, even with its partial execution, has proven that the SimPLe strategy for weakly-supervised breast cancer segmentation is not only effective with its original data but is also capable of adapting to other datasets through careful fine-tuning and modification of the initial segmentation mask generation process. The results achieved in this project, while preliminary, provide strong qualitative validation of the approach. They underscore the need for continued research and development to fully realize the potential of this method, offering a compelling foundation for future work in weakly-supervised medical image segmentation.

11 Bibliography

[1] - https://www.breastcancer.org/facts-statistics

[2] - https://www.cdc.gov/cancer/breast/statistics/index.htm

[3] - SimPLe: Similarity-Aware Propagation Learning for Weakly-Supervised Breast Cancer Segmentation in DCE-MRI

[4] - Duke Breast Cancer Dataset Description