UNIVERSITÀ DEGLI STUDI DI CASSINO E DEL LAZIO MERIDIONALE

Corso di Laurea Magistrale in Ingegneria Informatica

Segmentazione del seno in immagini MRI

Versione 0.1

Studente: Achille Cannavale
Matricola: 59721
Relatore: Alessandro Bria

June 30, 2025

Contents

Chapter 1

1.1 Introduzione

La segmentazione di strutture anatomiche in immagini mediche 3D rappresenta un task importante nell'ambito clinico e diagnostico. In particolare la segmentazione del seno da immagini MRI (Magnetic Resonance Imaging) 3D è un task complesso, ma essenziale per applicazioni come la pianificazione di interventi chirugici o l'analisi di anomalie tessutali.
In passato, la segmentazione del seno in immagini MRI 3D veniva effettuata principalmente con **metodi classici**, spesso semi-automatici o manuali, basati su:

- **Approcci a soglia** (thresholding) e **region-growing**, che sfruttavano differenze di intensità tra tessuti ma richiedevano regolazioni manuali e fallivano in presenza di rumore o basso contrasto.
- **Deformable models** (es.: Active Contours) e **algoritmi a clustering** (es.: k-means), sensibili all'inizializzazione e poco robusti alla variabilità anatomica.
- Metodi **atlas-based**, che allineavano immagini a template pre-annotati, limitati però dalla diversità inter-paziente.

Con l'avvento del **deep learning**, in particolar modo delle reti convoluzional e di architetture come **U-Net 3D**, la segmentazione del seno ha raggiunto livelli di accuratezza più alti rispetto al passato.

1.2 Contesto e Motivazione

La segmentazione automatica di strutture anatomiche in immagini mediche 3D rappresenta una sfida cruciale nell'elaborazione digitale di segnali, con ricadute significative in ambito clinico e diagnostico. In particolare, la segmentazione del seno in immagini **MRI (Magnetic Resonance Imaging) 3D** è un task complesso ma essenziale per applicazioni come la pianificazione di interventi chirurgici, il monitoraggio di terapie oncologiche o l'analisi di anomalie tessutali.

Nonostante i progressi del **deep learning**, la segmentazione del seno presenta difficoltà specifiche:

- **Variabilità anatomica**: la forma e le dimensioni del seno variano notevolmente tra pazienti, rendendo difficile l'applicazione di modelli generali.
- **Artefatti e rumore**: le immagini MRI sono soggette a distorsioni e rumore, come il movimento del paziente o le inhomogeneità del campo magnetico, che possono compromettere la qualità delle immagini e la precisione della segmentazione.
- **Limitazioni dei dataset**: la scarsità di dati annotati da esperti, specialmente per volumi 3D ad alta risoluzione, limita la capacità di addestrare modelli robusti e generalizzabili.

> Questo lavoro nasce durante un tirocinio universitario, dove ho sviluppato una **pipeline di deep learning** per automatizzare la segmentazione del seno in MRI 3D, con l'obiettivo di migliorare accuratezza e riproducibilità rispetto a metodi tradizionali.

1.3 Background e Stato dell'Arte

1.4 Background e Stato dell'Arte

La segmentazione semantica di immagini mediche rappresenta uno dei compiti fondamentali nell'ambito della diagnostica per immagini assistita da computer (CAD - Computer Aided Diagnosis). Essa consente di delineare strutture anatomiche e lesioni, migliorando l'accuratezza e l'efficienza dei processi diagnostici e terapeutici.
Negli ultimi anni, l'avvento del deep learning ha rivoluzionato il campo dell'elaborazione di immagini mediche, introducendo architetture sempre più sofisticate per la segmentazione automatica. Una delle architetture più note e utilizzate è la **U-Net**, proposta da Ronneberger et al. [**ronneberger2015unet**], che ha introdotto un design encoder-decoder con connessioni skip per preservare i dettagli spaziali durante il processo di ricostruzione dell'immagine segmentata.
Successivamente, sono state sviluppate numerose varianti e miglioramenti, tra cui architetture basate su reti convoluzionali più profonde, reti residuali e approcci multi-scala. Tuttavia, negli ultimi anni l'interesse si è spostato verso l'impiego di **Transformers**, inizialmente sviluppati per il Natural Language Processing (NLP), e adattati con successo al dominio delle immagini mediche. Esempi notevoli sono **TransUNet** [**chen2021transunet**] e **Swin UNETR** [**hatamizadeh2022swin**] che combinano l'efficacia dei Transformers nel modellare le dipendenze globali con la capacità di apprendere rappresentazioni spaziali locali, tipiche delle CNN.
Il framework **MONAI** (Medical Open Network for AI) [**cardoso2022monai**] si è affermato come uno standard de facto nello sviluppo di pipeline per la segmentazione e classificazione di immagini mediche. Basato su PyTorch, MONAI offre un'ampia gamma di trasformazioni, modelli predefiniti, reader DICOM e metriche di valutazione, risultando particolarmente adatto per workflow in ambito sanitario.
Nel presente lavoro, è stato impiegato MONAI per la gestione dei dati e la costruzione della pipeline di addestramento. Per il task di segmentazione della mammella su risonanza mag-

netica, si è fatto uso del dataset **Duke-Breast-Cancer-MRI**, disponibile su The Cancer Imaging Archive (TCIA) [**duke˙breast˙mri**], che contiene serie DICOM annotate manualmente da radiologi esperti.

1.5 Strumenti Utilizzati

Durante il tirocinio sono stati impiegati diversi strumenti software e librerie fondamentali per la gestione, il pre-processing e l'elaborazione di immagini medicali, nonché per lo sviluppo e il training di modelli di deep learning.

- Il linguaggio principale di lavoro è stato **Python**, scelto per la sua flessibilità e per l'ampio ecosistema di librerie scientifiche.
- Per la manipolazione delle immagini DICOM è stata utilizzata la libreria **MONAI** (Medical Open Network for AI), un framework open-source basato su PyTorch e progettato specificamente per applicazioni di imaging medicale. MONAI ha permesso di gestire agevolmente il caricamento dei dati, le trasformazioni e la normalizzazione delle immagini, grazie a un sistema modulare di trasformazioni componibili.
- Il framework di deep learning impiegato è stato **PyTorch**, scelto per la sua semplicità d'uso, il supporto attivo della community e le sue prestazioni elevate, specialmente in combinazione con l'accelerazione GPU fornita da CUDA.
- Lo sviluppo del codice è stato realizzato su una macchina Linux, con accesso remoto e con l'ausilio dell'editor **Visual Studio Code**.

1.6 Composizione del Dataset

Il dataset utilizzato durante il tirocinio proviene dal database **Duke Breast Cancer MRI**, un archivio pubblico contenente immagini DICOM di risonanze magnetiche della mammella. Ogni caso clinico è organizzato in una struttura gerarchica che riflette l'identificativo univoco del paziente e delle relative acquisizioni.
Il dataset è composto da sequenze **DICOM** tridimensionali, ciascuna rappresentante un'acquisizione volumetrica del seno. In tutti i casi sono disponibili anche annotazioni o maschere segmentate manualmente da esperti, utilizzate come ground truth per il training dei modelli di segmentazione.
Durante la fase di preparazione, le immagini sono state convertite in formato compatibile con MONAI tramite una pipeline di preprocessing, che ha incluso:

- il **caricamento** delle immagini in memoria;
- la standardizzazione dell'intensità e il ridimensionamento spaziale;
- l'organizzazione dei dati in **dizionari** contenenti sia l'immagine sia i relativi metadati;
- la **salvataggio** del dataset preprocessato

Per ogni volume è stata mantenuta l'associazione tra immagine e metadati (es. spacing, orientamento, posizione del paziente), necessari per garantire una corretta elaborazione e interpretazione dei dati durante le fasi di training, validazione e inferenza.
L'intero dataset è stato infine suddiviso in tre sottoinsiemi: **training**, **validation** e **test**, rispettando la proporzione e la varietà dei casi clinici per assicurare una valutazione affidabile delle performance del modello.

Chapter 2

Deep Learning

2.1 Trasformazioni

Per garantire una buona generalizzazione del modello, sono state provate diverse trasformazioni, usando la libreria **Albumentation**, tra le quali sono state scelte le seguenti migliori per il nostro caso di studio: (anche una tabella va bene)

Trasformazione
A.HorizontalFlip
A.VerticalFlip
A.ColorJitter
A.RandomBrightnessContrast
A.MotionBlur
A.GaussNoise
A.CLAHE
A.CoarseDropout
A.Resize
A.Normalize

Table 2.1: Lista delle trasformazioni utilizzate per il dataset.

In particolare, I parametri della normalizzazione sono stati calcolati dal dataset stesso:

Parametro
Media RGB
Deviazione standard RGB

Table 2.2: Valori di media e deviazione standard per la normalizzazione delle immagini.

Una delle trasformazioni che hanno garantito un forte miglioramento è rappresentata da **CoarseDropout**, che randomicamente oscura regioni rettangolari dall'immagine, simulando l'occlusione ottica e variando la grandezza degli oggetti nel mondo reale. (esempi immagine)

2.2 Defined Environments

<MINTED> The template relies on the excellent `tcolorbox` package for formatting the boxes within the document and for that end different styles were created.

Sometimes one needs to quote either a proverb or to create drama, for this use the `excerpt` environment with the following notation and effect.

<MINTED>

Compiling this code snippet would show as in the document

> To be, or not to be...

<MINTED> During the preparation of your document, it is useful to showcase some code either in the shape of all the document or a snippet of it. There are two (2) ways of doing this where the first one will be discussed here.

For example to print out a hello world in python, please use the following environment

```
\begin{code}{python}
print("Hello, World!")
\end{code}
```

Producing the following:
<MINTED>
The class also come with some predefined environments to modify the behaviour/aesthetics of the document. Highlighting text is **very easy**, here is an example on how to write one.
<MINTED>

<MINTED> Sometimes you need to showcase an example or need to highlight a certain idea. For these things the environment Example could be useful.

For example to show as simple example or give a slight attention to a topic you can do the following.

> This is an example. This could be anything which you would like to have a certain amount of attention but not too much as to distract from the flow of the document.

<MINTED> Or sometimes you need to give a clear break to the flow of the document and ask the reader to look at your banner. For that use highlight.

> Hey! Pay attention as this is a highlight box.

2.2.1 Writing Equations

One of the strong suits of LaTeX compared to other editors and programs is it simplicity and ease of use methods of writing equations. Consider the following equation:

$$f(x) = x^2 + 2x + 1$$

In code form this would be written as:
<MINTED>
All equations that has their newline and centre staged are mostly written in an environment where it has a <MINTED> and an <MINTED>. You may have noticed the asterisks sign just after the equation. This implies the environment is **not numbered**, meaning you won't be able to reference it. This is used to limit the numbering of equations to just the essential parts in the document and not reach 3 digits by the time you are in page 8. For a numbered equation like the following

$$f(x) = x^2 + 2x + 1 \tag{2.1}$$

You only need to do:
<MINTED>
where <MINTED> is the equation reference label. You could also make matrices as well as <MINTED> is preloaded into this template.

2.2.2 Designing a Table

Finally, no template is done without someone telling you how a table should be designed. Below is a standard table And the code used to generate it:

Section
Introduction
Methods
Results
Discussion

Table 2.3: A Detailed look into the scientific method.

<MINTED>

Chapter 3

Plotting your data using PGF/TikZ

3.1 Introduction

PGFplots and Tikz are powerful scripting languages allowing you to draw high-quality diagrams using only a programming language. PGFplots are generally used for plotting data from a wide variety of representations from simple 2D plots to complex 3D geometries.
But wikipedia description put it best:

> PGF/TikZ is a pair of languages for producing vector graphics (e.g., technical illustrations and drawings) from a geometric/algebraic description, with standard features including the drawing of points, lines, arrows, paths, circles, ellipses and polygons. PGF is a lower-level language, while TikZ is a set of higher-level macros that use PGF. The top-level PGF and TikZ commands are invoked as TeX macros, but in contrast with PSTricks, the PGF/TikZ graphics themselves are described in a language that resembles MetaPost.

For more info please look at the documentation here. It is of course up to the user to select which graphical software to produce the necessary visual components but unless it requires complex functions/processing, it would be be easier to have it in PGF/TikZ format for easy editing/maintenance.
For this manual we will be looking at the three (3) plot types you may encounter in your studies.

3.1.1 A Simple 2D Plot

2D plots are simple yet powerful to show the relation of a single parameters and its related function. Below is an example of a simple comparison of two (2) functions. The image above is generated using the following code:
<MINTED>

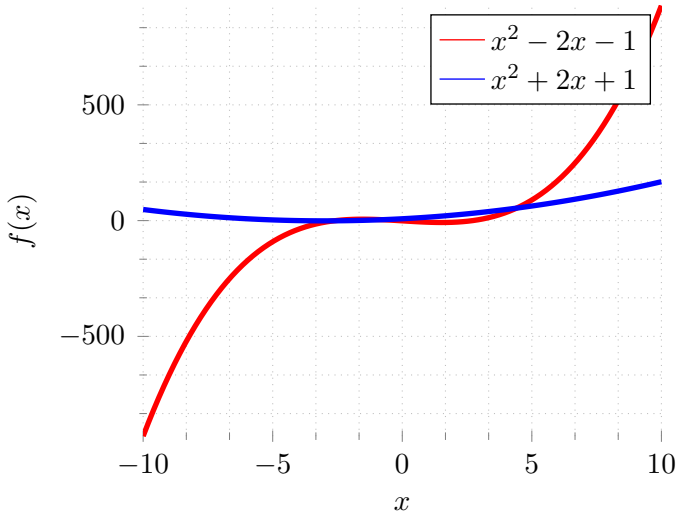

Figure 3.1: This is an example of a 2D PGF plot comparing two functions where these functions are calculated using PGF itself rather than entering/reading from data.

As can be seen it is relatively standard to create plots. Some aspect which need mentioning.

`<MINTED>` You invoke this command when you want to create a plot. In the square brackets (i.e., []) you insert your **configuration** of your plot. The most important ones are

`<MINTED>` the range in which the function will be calculated

`<MINTED>` the number of calculations will be done within the defined domain.

3.1.2 Plotting 3D plots

Plotting data with PGFplots is also quite possible and will generate great plot (as long as it is not massively complicated). For more information on the precautions on designing 3D plots, please have a look at here.
Below is the prototypical plot to showcase the 3D capabilities of PGF:

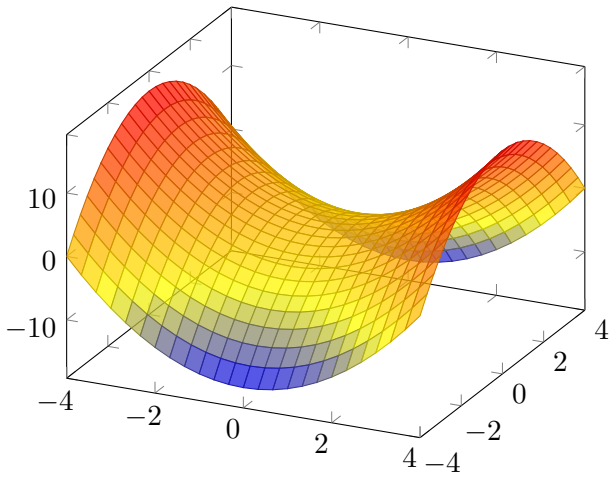

Figure 3.2: An example 3D plot done wit PGFplots.

And, of course the code for generating the plot is given as follows:

<MINTED>

Some options worth mentioning are as follows:

<MINTED> Generates a **surface** based on the 2D data it was given (in this case these are x and y.

<MINTED> Describes, basically how each segment should be filled.

<MINTED> Similar to 2D plots, tells how many data points will be measured. However, make a note that 3D is significantly more taxing on the TeX memory than 2D and making this sampling high may result in exceeding the memory limit.