Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Development of pre and post-processing steps to a pipeline aimed to identify silent cerebral infarcts

Supervisor:
Prof. Gastone Castellani

Co-supervisors:
Prof. Daniel Remondini
Dr. Riccardo Biondi

Submitted by:
Nicolas Biondini

Academic Year 2021/2022

Abstract

Sickle Cell Disease (SCD) is a group of disorders of red blood cells that cause abnormal hemoglobin and could lead to different symptoms.

Brain MRI scans of patients affected by this condition show peculiar lesions in White Matter, without any apparent neurological evidence, called Silent Cerebral Infarcts (SCI).

To increase the comprehension of this condition is necessary to find and segment the lesions. Up to now this process is performed manually, comporting an high consumption of time and a dependence on the experience of the involved operator.

The European project GenoMed4All aims to provide solutions, control and prevention for haematological diseases, including SCD, by applying AI technologies. In this context, a pipeline for identifying and segmenting SCIs was proposed.

This work of thesis aims to develop and implement a pre-processing and post-processing steps to improve the results obtained with the proposed segmentation technique.

The pre-processing step includes a phase of brain automatic extraction and segmentation of its main tissues. The post-processing step consists in the classification of the lesions found in the segmentation step, aiming to remove the false positives.

The proposed steps were developed and tested on a data set of MRI scans provided by different medical centers in Italy. The performances of the pre-processing step were tested comparing the obtained results with those of the software FSL, which is a standard in the analysis of MRI scans. The brain mask comparison measured a Dice coefficient of 0.87, the white matter of 0.78, the grey matter of 0.67 and the cerebrospinal fluid of 0.66.

The post-processing step was developed training a set of three classifiers and their results were compared to the manual annotation of the SCIs in the same data set, obtaining a, AUC precision-recall for the best classifier of 0.75.

Contents

Introduction

Sickle Cell Disease (SCD) is a group of inherited red blood cell disorders. A subject affected by SCD has abnormal hemoglobin, which causes the red blood cells to become hard and sticky and look like a sickle. Those cells die early, which causes a constant deficiency of red blood cells [1].

The main symptom of SCD is anaemia, a medical condition that consists in a low value of the haemoglobin (the substance that resides in the red blood cells, responsible of the oxygen transport) in blood. Another frequent symptom is the occurrence of painful episodes known as *sickle cell crisis* due to blood vessels that become blocked and are frequently localized in limbs or back. People with SCD are more vulnerable to infections, particularly in the early ages of life [2].

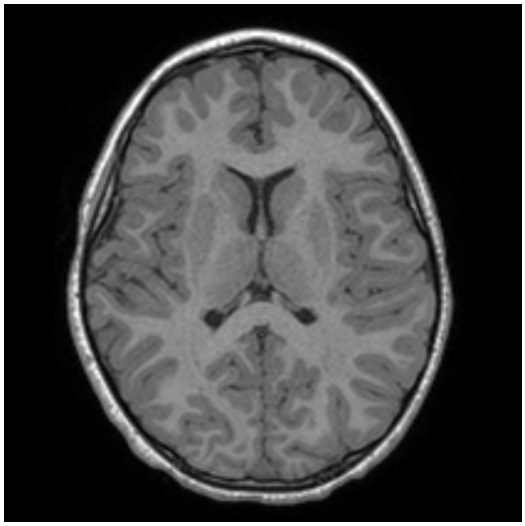
The most common neurological complication in subjects with SCD is the presence of Silent Cerebral Infarcts (SCIs)[3]. SCIs are defined as an abnormal magnetic resonance imaging of the brain in the setting of a normal neurologic examination, without a history or physical findings associated with an overt stroke [4]. In the brain magnetic resonance imaging(MRI) SCI usually present itself as an hypointense region in the T-1 weighted image, or as an hyperintense region in the FLAIR image, and are localized in the white matter as show in Figure 1.

SCIs' consequences includes a decrement in general intellectual abilities, poor academic achievement, working abilities and a minor quality of life [4]. Specific morbidity can lead to a progression to overt stroke and progressive SCIs [4, 5]. Neuropsychological and neuroimaging studies are needed to understand how SCI negatively affect cognition and provide a starting point for the identification of potential targets for preventive therapies [6]. Up to now the segmentation of those lesions is manually made by high trained and specialized neuroradiologists, which is highly time consuming(severaly hours) and can be very subjectively, influenced by the experience of the operator.

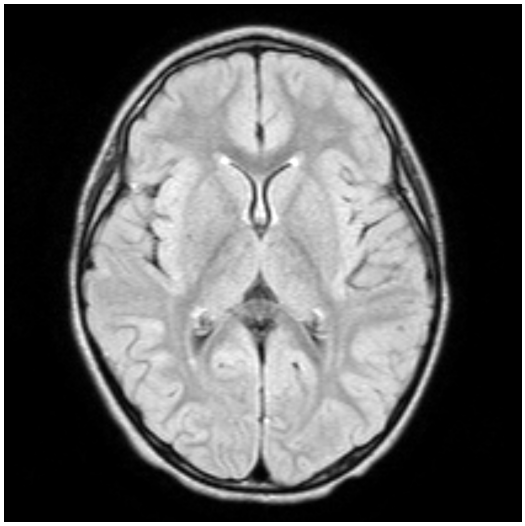
To overcome those issues it was proposed the usage of a neural network to automatically segment the SCI in a brain MRI. A stepwise procedure was used to achieve the segmentation of those lesions, differentiating the segmented regions from other non-clinically relevant hypo regions in the white matter [5].

To do that it was necessary to process the MRI images before (pre-processing) and after (post-processing) the application of the implemented SCI segmentation tool. In the pre-processing stage the aim is to prepare the image to be processed by the segmentation tool, cleaning it from everything could badly influence the success of the AI segmentation. In the post-processing stage the purpose was to refine the results obtained by the neural network.

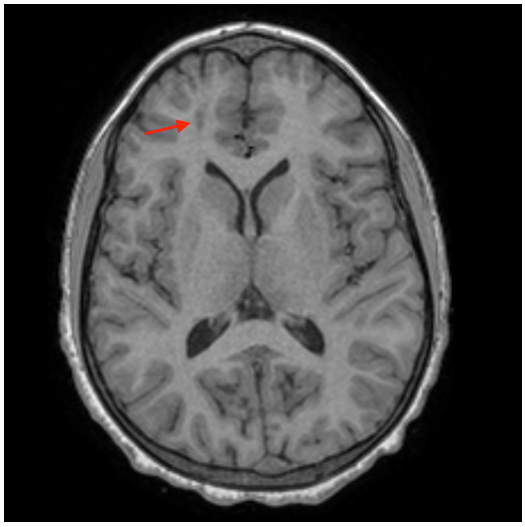
This work of thesis, made in collaboration with the Department of Neuroscience of the University of Padova, fits in the contest of the European project "Genomed4All"[7] which aims to find correlation between -omics data and pheno-



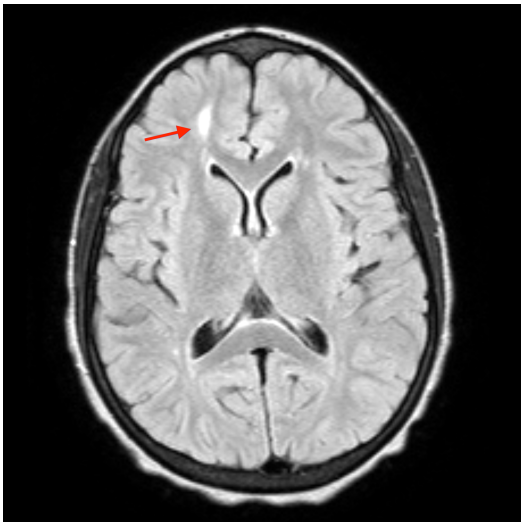
(a) *Axial slice of a T1-weighted MR scan without SCI evidences*



(b) *Axial slice of a FLAIR MR scan without SCI evidences*



(c) *Axial slice of a T1-weighted MR scan with SCI evidences*



(d) *Axial slice of a FLAIR MR scan with SCI evidences*

Figure 1: *Comparison between healthy brain image and brain with SCI in both T1-weighted and FLAIR MR scan. It's possibile to appreciate how the SCI appears as an hyperintense region in the FLAIR image and as an hypointense region in the T1-w image, pointed by the red arrow.*

type by the help of Artificial Intelligence [8] using European data of patients affected by haematologic diseases, that often presents anomalies in the DNA.

The purpose of this work of thesis is to develop reliable, open source and relatively quick strategies for the pre-processing and post-processing phase.

Chapter 1

Preliminary Knowledge

The aims of this work of thesis was to prepare head MR images for a segmentation using a U-Net and then to refine the outcomes cleaning the misclassified ones. In order to do that different techniques was used and is necessary to briefly define, show and explain the approaches used. In this chapter, medical images are briefly introduced, then the main segmentation techniques used are illustrated; finally, the metrics useful for evaluating the results obtained are treated.

1.1 Medical Images

An image is a collection of measurements in two-dimensional (2-D) or three-dimensional (3-D) space [9]. A medical image is a discrete representation of the internal structure or function of an anatomic region in form of a tensor of picture elements called pixels/voxels. A pixel is a discrete numeric representation of intensity or gray-level[10]. The representation results from a process that maps every numerical value in a position in space and the number of picture elements involved express the level of detail with which the subject will be depicted [11]. The physical meaning of the numerical value of a pixel changes according to the imaging modality, the acquisition protocol, the reconstruction and eventually the post-processing. Nevertheless there are 4 concepts in common for every medical image: `pixel depth`, `photometric interpretation`, `metadata` and `pixel data` [12]. In the following lines these characteristics will be rapidly described.

Pixel Depth Pixel depth is the number of bits used to encode the information stored in each pixel. A higher number of bits per pixel permits to store a greater information but requires a greater usage of memory [12]. We can exemplify the concept just expressed considering a square image with a side of 256 pixel of depth 2 $bytes = 16$ bit (1 $byte = 8$ $bits$). In this case each pixel can express $2^{16} = 65,536$ levels which are usually arranged as integers in range $[0; 65,535]$. It is also possible store the values in the interval $[-32,768; 32,767]$ using 15 bits to store the level and a bit to represents the sign. In this example the image will occupy $256 \times 256 \times 2 = 131,072$ $bytes$.

It is to mention also the possibility for a pixel to store a real number. For this case the Institute of Electrical and Electronics Engineers created a standard (IEEE-754) in which defines two basic formats for the binary encoding of a floating point number: *single precision* with 32 bit depth and *double precision* with 64 bit depth.

Photometric Interpetation The photometric interpretation specifies how the pixel data should be interpreted for the correct image display as a monochrome or color image. To clarify that sentence it is useful to introduce the concept of *number of channels*, also known as *sample per pixel*. Monochrome images have only 1 sample per pixel and all the bits in the pixel depth is used to represents the gray level [12]. Typically x-ray computed tomography (CT) and magnetic resonance (MR) images are monchrome, and so, in that elaborate, we will mainly consider those kind of images. Furthermore nuclear medicine images, such as positron emission tomography (PET) and single photon emission tomography (SPECT) the data are usually visualized with a color map, that map is however predefined and the colors are not information stored in the pixels. In that case the number of channels is 1 and that images are usually referred to be in *pseudo-color*. To encode color information into pixels is necessary to have more than one number of channels. It is common to have RGB (Red, Green and Blue) images, which are composed of 3 channels. In this case the pixel depth can be obtained multiplying the number of bits per channels (usually 8 bits [13]) for the number of channels [12]. An example of colored images in medical usage are the Doppler ultrasound in which the color is used to encode blood flow direction and velocity. [11].

Metadata Metadata are informations that describe the image. It is usually stored at the beginning of the file as a header and contains at least the image matrix dimensions, the spatial resolution, the pixel depth, and the photometric interpretation [12]. Using the metadata, advanced medical image visualization software are able to correctly read, display and elaborate medical images if the format is supported. In medical images metadata acquires an also greater importance due to the nature of the images itself. It is possible to store information about how the image was produced and even informations on the patient [11].

Pixel data Pixel data represents the numerical value stored in the pixel. According to the data type, pixel data are stored as integers or floating point numbers. Although is not common, it is also possible to store complex numbers in pixels. An example is pixel data from MRI before the reconstruction that provides informations about both the phase and the intensity (The so called *k-space*) [12].

1.1.1 Medical Image Format

Image file formats provide a standard way to store the information describing how the image data are organized inside the image file and how the pixel should be interpreted by software for the correct loading and visualization. First of all, it is necessary to distinguish between two main categories of formats: The first is intended to standardize the images generated by diagnostic modalities and the seconds has the purpose of facilitate and strengthen post-processing analysis [12]. The example here discussed are `DICOM` for the first category, that is the original file format of the datas analyzed in that paper, and `Nifti` for the second category, which is the format mainly used during during this thesis work.

DICOM The Digital Imaging and COmmunications in Medicine (which acronyms is `DICOM`) standard was established by the American College of Radiology and the

National Electric Manufacturers Association in 1993 and was introduced in imaging departments a the end of the '90s. Now it is the backbone of every medical imaging department, also because it is not only a file format but also a network communication protocol. The value of `DICOM` is that the pixel data and the description of the medical can't be separated, accentuating the concept that an image is quite meaningless without its metadata and so they are merged in a unique file. The `DICOM` header contains a full description of the entire procedure used to acquire the image, including patient information such as name, gender, age, weight, and height [12]. It has 2 main limitations: As a matter of fact pixel that can only store integer values (altough floating point data can be stored in the metadata), and also it is born for only 2D images, so a 3D volume is described by a series of files containing the single slices [11].

Nifti The `Nifti` format was originally created at the beginning of 2000s by a committee based at the United States' National Institutes of Health (known as NIH) [12]. It's strength is due to the header that can store information about image orientation image centre and origin, and that can, for example, resolve ambiguity between left and right in brain hemisphere MRI. It is supported by many software of image viewing and can store 3D images. For those reasons it is the format used in this work.

1.2 Magnetic Resonance Imaging (MRI)

Magnetic Resonance Imaging is an imaging technique base on the physics phenomenon of Nuclear Magnetics Resonance (NMR), a process that involves the magnetic moments of the atoms that, immersed in a constant magnetic field, align themselves with the external magnetic field, giving rise to a small paramagnetic polarization contrasted only by the thermal agitation. The constant magnetic field also give rise to a phenomenon called *Larmor precession*, in which the nuclei's magnetic moment precess around the external magnetic field direction with a particular frequency, called *Larmor frequency*, dependent on the external field. Applying an oscillating magnetic field, perpendicular to the constant magnetic field (z axis), at the Larmor frequency puts in resonance the nuclei, which magnetic moment's latitude will change accordingly to the magnetic pulse received. Inserting in the apparatus a coil with axis perpendicular to the plane where lie the magnetic fields, it is possible to register an oscillating induced voltage [14].

The nuclei, immersed in the constant magnetic field, will tend to come back to their original orientation once the oscillating magnetic field is turned off. NMR informations, and in particular the ones needed for the recording of an MRI images, comes just from the relaxation time of the magnetic polarization, that is the time needed for the vector to return parallel to the constant field. It is necessary to underline that magnetic polarization is a vector and so it can be splitted in two components, one longitudinal (z axis) and one transversal (xy plane). The relaxation is so defined as the time to made the transversal component go back to 0 and the longitudinal component return to the initial value. Those two processes has different individual relaxation times: T_1 for the longitudinal component and T_2 for the transversal component. The processes of realignment in longitudinal and

transversal directions have different natures. The longitudinal relaxation for a nucleus is due to the interaction of the spin with the lattice and so the T1 time is also called *spin-lattice relaxation time*. The transversal relaxation otherwise is due to the interaction between the nucleus' spin with other spins, then the T2 time is also called spin-spin relaxation time. It is also to be noticed that the T1 time is longer than T2. This process of relaxation produce a signal called Free Induction Decay (FID) that is the signal registered.

1.2.1 Pulse Sequences Images

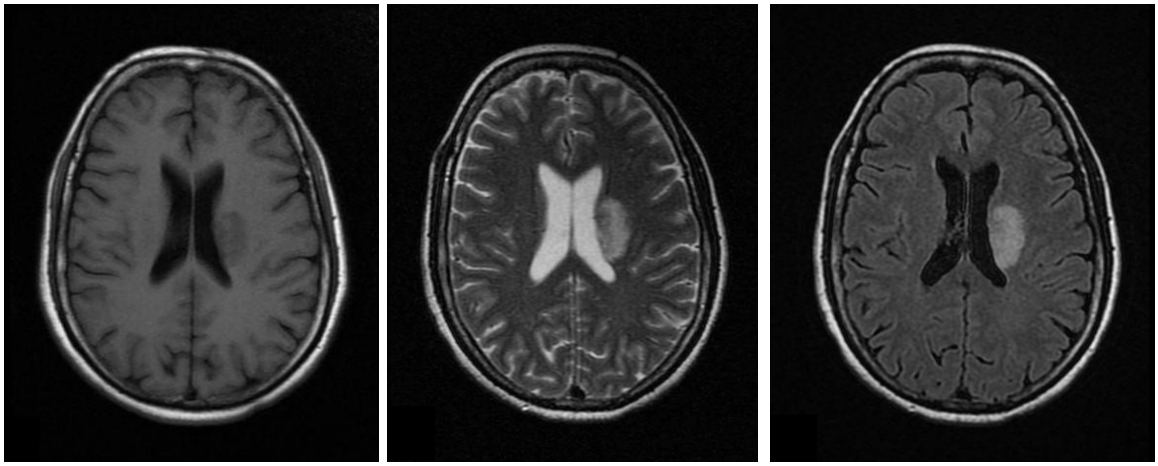
In MRI the constant magnetic field is usually not uniform, but has a gradient. The presence of a not uniform field lead to nuclei with different Larmor frequency depending on their displacement in the space. Adjusting the frequency of the pulsing field is possible to select only a small slice of space in which nuclei will have that determined resonance frequency.

It is possible to obtain different contrast between tissues applying particular sequences of impulses for the oscillating magnetic field that permits to maximize or minimize the effect of some particular physics phenomena of the FID. It is, for example, the case in which the FID from T1 relaxation time or the T2 relaxation time are maximized, as briefly explained below. Another example of phenomena which FID can be maximized is the diffusion, but they wouldn't be treated because are not relevant for this work of thesis. In the next lines will be briefly discussed the main images obtained by the different sequences and the most used during the work.

T1-Weighted T1 Weighted image (also referred as *spin-lattice*) is one of the more frequently recorded image modality in MRI and demonstrates differences in the T1 relaxation times of tissues. For example fat's longitudinal magnetization realigns rapidly with B_0, and it therefore appears bright. On the contrary water has a slower relaxation time in the longitudinal axis and therefore has a lower signal appearing dark [15].

T2-Weighted T2 Weighted image (known also as *spin-spin*) is also one of the basic pulse sequences on MRI and highlights differences on the T_2 relaxation time of tissues [16]. In this kind of images the water appears bright and conversely fat appears darker than the T1-w images. T2-weighted images are often most helpful for assessing areas of pathology because diseased or injured tissue contains a higher water content than is normal, resulting in signal hyperintensity [17].

FLAIR FLuid Attenuated Inversion Recovery (FLAIR) is a special sequence that removes signal from the cerebrospinal fluid in the resulting images. Brain tissue on FLAIR images appears similar to T2W images with grey matter brighter than white matter, but cerebrospinal fluid (CSF) is dark instead of bright. It is very useful in evaluating cerebral infarts and multiple sclerosis lesions because they will appear hyperintense in this kind of images [18].



(a) *T1-W image* **(b)** *T2-W image* **(c)** *FLAIR Image*

Figure 1.1: *Comparison between 3 images obtained with different pulse sequences of the same brain [19]. It is possible to observe how the cerebrospinal fluid (CSF), the white matter (WM) and grey matter (GM) appear in the different images. The lesion here observable therefore appears hypointense in the T1W image and hyperintense in the T2W and FLAIR images.*

1.2.2 Image Acquisition Type

As explained in the previous section, in MRI the resonance is possible to encode spatial information by using a magnetic gradient, due to the dependency of the Larmor frequency on the intensity of the local magnetic field. Traditionally this effect was used to select only a small slice of tissue and the image recorded was in two dimensions. The MRI volume was in this case recorded as a set of planar images. In more recent years another technique has been developed that permits to record a volume in its integrity, obtaining images with truly cubic voxels and with a higher resolution [20].

1.2.2.1 2D Imaging

In two dimensional imaging a constant field with a gradient is applied on the z axis and a selective radiofrequency pulse is applied on the xy plane. In this way only the nucleis on a small slice will be excited and that permits to select only a set of voxels in an xy plane. The spatial informations about the remaining two dimensions (x and y) are provided by an encoding of frequency and phase.

It is possible to select thinner slices increasing the gradient on the z axis but there is a limitation to the thickness that can be choose. Usually is difficult to select slices of a thickness lower than 3 mm. This is due to the Signal-to-Noise ratio (SNR) because smaller voxels will produce a lower signal. This will cause to have voxels that represent volumes with different tissues in it, this is the so called *partial volume effect*. Also the slices are usually separated by a gap of non included tissue to prevent overlapping between successive slices.

The result of this acquisition technique is then a set of 2D images with a low resolution on the z axis [20].

1.2.2.2 3D Imaging

In 3D imaging the constant field on the z axis has not a slice selecting gradient and the entire sample volume is excited simultaneously. The spatial information on the z axis is provided by another phase encoding. 3D images can be seen as a rectangular box that delimits the anatomic region in all dimensions subdivided in smaller contiguous voxels. In 3D scans voxels are usually isotropic, which means that have the same dimensions in all the three axis. The images acquired with a 3D technique have usually a greater spatial resolution and having smaller voxels permits to reduce the partial volume effect.

1.3 Registration

Image registration is an image processing technique used to align and overlay two or more different images or volumes. The need for a registration appear whenever it is necessary to compare two images containing different informations about the same subject [21] or also when a study on many patient is done and there is the need to have all the volumes in the same space. An example in medical field can be the case in which it necessary to compare two images of the same patient taken with different impulse modalities (i.e. T1-W and FLAIR). In this case, the examined anathomical structure is the same, however the two images are not necessary overlapping. That because may happen that a patient change his position during the exam. An other use case may be in case of multi-patients studies, where there is the necessity to compare the different images to an atlas.

Image registration process involves two images: a moving image ($I_m(x)$) which is moved and deformed, and a fixed image ($I_f(x)$), which is the reference one. The registration process allows to find the transformation ($T(x)$) which allows the moving image overlay the reference one. The best transform is estimated by minimizing a certain *cost function*. This function has to be chosen according to the image type and registration purposes [22], since different types of cost functions can lead to very different results in the registration process. An example of of this cost function can be the *mean squared error* between the pixel intensities of the fixed and moving image. This kind of function can have good results registering two images acquired with the same modality but would not work properly in registering images acquired with different modalities (e.g. a T1W scan and a FLAIR scan), because the same tissue have different pixel values in the two images. In this cases other metrics are more effective, for example the *mutual information*.

There is the possibility to distinguish between two kinds of process that lay under the name of registration. In the first meaning the word is used to indicate the process of finding a transformation that can relate the position of features in one image with the position of the corresponding feature in another image. The second meaning both relates the position of corresponding features and enables us to compare the intensity at those corresponding positions. In this second meaning the concepts of *resampling* and *interpolation* are incorporated [23]. Resampling is the process in which the orientation, resolution or field of view of an image is changed [24]. In image registration this could be necessary because the sampling grid of the original moving image is modified by the applied transformation. Due to the changing of sampling of the moving image it appear the problem of finding

the pixel intensities for the new pixels in the transformed grid. This can be done estimating the new values starting from the original ones and taking into account the transformation applied to the image. This process is called interpolation.

Another aspect to take in account is the type of transformation that can be applied to the moving image. They can be subdivided by the *degrees of freedom (dof)* of the transformation:

- **Rigid** transformation allows 6 dof. In this kind of transformation the image is treated as a rigid body which undergoes to rotation (3 dof) and translation (3 dof) along the axis. In medical application is usually applied when the subject of the image is rigid (e.g. a bone) or when the two images are acquired with the same modality but with a time lapse between the two acquisitions;

- **Affine** transformations allows 9 or 12 dof. This kind of transformations incorporate also scale (3 dof) and skews (3 dof) besides rotation and translations. Even if for the nature of the biological organs this approach don't increase greatly the applicability of image registration, because usually organs don't only stretch or shear, this kind of transformation is useful for overcoming scanner induced errors [23];

- **Non-affine** transformations have an indefinite number of dof and allows also the deformation of the moving image. There are many different non affine transformations which uses different approaches. For example the non-rigid component of the transformation can be determined using a linear combination of polynomial terms, basis functions, or B-spline surfaces [23]. An interesting example of non-affine approach is the deformation grid, in which a grid is overimposed on the moving image and then stretched and deformed to match the fixed image.

1.4 Segmentation

Image segmentation is defined as the partitioning of an image into non-overlapping, constituent regions that are homogeneous with respect to some characteristic such as intensity or texture [9]. The segmentation of an image have a great relevance for medical purposes because it can help extracting features that provides informations about organ volumes or lesions identification [11]. Usually it can be performed identifying a surface for each tissue class (*boundary identification*) or by the classification of each pixel in the volume (*volume identification*) [25]. The classification of image segmentation techniques is challenging or even impossible because of the different criteria that can be used. One classification criteria can be defined by the segmentation outcome: *hard-segmentation* that returns a net assignment of a pixel to a determined class, or *soft-segmentation*, where the output is a map of probabilities to belong to each class. It is always possible to transform a soft-segmentation to a hard-segmentation by applying a threshold to the probability value.

Another possible classification for the segmentation process depends on the degree of human interaction required to achieve the segmentation. Using this criterion we can distinguish between *manual*, *semi-automatic* or *automatic* segmentation.

Manual segmentation technique requires a trained and high specialised operator that has to visually recognise and identify the searched region. The operator usually visualize the images using specialized softwares and manually annotates them. Right now it is still the most reliable and precise technique but it ha the disadvantages to be highly time consuming and operator-dependent [10].

Semi-automatic segmentation techniques requires a limited interaction by an human operator to set the parameters for the image segmentation. Those techniques includes operation like clustering and thresholding in cases where the parameters are not predetermined but has to be manually set for each volume. Another highly used semi-automatic set of applications are the ones in which the images are firstly roughly segmented automatically and then manually corrected by an operator. While this approach permits to save time it keeps being operator-dependant.

Automatic segmentation Thanks to the improvement of computer power it began possible a fully automatic segmentation, which means that no human interaction is needed. Often done with the help of Artificial Intelligence (AI) as in case of Neural Networks [5]. This technique is faster and totally operator-independent, but it is hard to implement[10] and is still in a development state.

1.4.1 Principal Segmentation Methods

During the years in literature are appeared several techniques on medical image segmentation. When approaching a segmentation more than one of those techniques can be combined into pipelines to obtain the best results [9]. In the following lines will be described the ones used in this work.

1.4.1.1 Thresholding

Thresholding approach segment a scalar image by creating a binary partitioning of the image intensities [9]. In thresholding the main goal is to find the threshold that divides the histogram of the intensity occurrences of the image in two classes, one with intensities higher and one with intensities lower than the threshold, that are physically significant. An example of thresholding segmentation can be seen in Figure 1.2.

It is common that more than two classes has to be find and in this case the process is called *multithresholding* [9]. The simple application of a thresholding (or mutithresholding) approach is often not sufficient to obtain a proper segmentation due to the presence of noise or intensity nonuniformity, often found in medical images. Despite that thresholding continue to be used, due to its semplicity, when the problem permits it or as a step in a more complex approach[25].

1.4.1.2 Statistical Pattern Recognition

The statistical pattern recognition field groups many different techniques of segmentation. Even if giving a proper definition is not easy, it is possible to summarize the concept of saying that it consists in the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories [27].

(a) *Original Image* **(b)** *Thresholded Image* **(c)** *Image Histogram*

Figure 1.2: *Example of thresholding application to a common image [26]. In this case the aim of the segmentation was to recognise the subject (class 1) from the background (class 2). In the histogram image (c) it is possible to see the two classes and the threshold division.*

In order to do that classification it is necessary to evaluate a set of *features* for each pixel. A feature is an individual measurable property or characteristic of the pixel itself [27]. This changes from the original variables (the image) to some new variables (the feature) is aimed to make the pattern recognition problem easier to solve [27] as those features forms a set of patterns and the classification of that patterns permits the classification for each pixel [25]. For example the more common visual features include color (or grey level in 1-channel images), shape or texture [28].

The pattern recognition techniques are usually classified in *supervised* or *unsupervised* depending on their need for a set of example target labels in the training set or not. In this section some of the main used pattern recognition techniques will be briefly discussed, such as *classifiers* as a supervised approach and *clustering* for unsupervised.

Classifiers need for an already labeled set in the training stage and so they are considered as a supervised approach. The training data is usually manually segmented to let the classifier learn how the features can lead a pixel to belong to a class or another. Once trained the statistical classifier can assign to every pixel a probability of belonging to each class, based on the extracted features. The main disadvantage of this approach is the necessity for human work to collect the training set. Those training data must be sufficiently large to offer to the classifier enough variability in the set, otherwise the classifier risks to over specialize only on the training data [9, 25, 11].

Clustering doesn't require labelled training examples since it is an unsupervised approach. This approaches usually require an initial choice of parameters from which they are particularly dependent [9, 11]. For brain segmentation usually the number of clusters is previously known and so it is frequent that algorithms that needs this parameter as input are used. Three from the most used algorithms of that kind will be briefly explained in the next lines as examples:

- **k-means clustering:** it permits to partition n ($X(x_1, x_2, ..., x_n)$) pixels in k clusters and minimize the distance function ($\sum_{i=1}^{k} \sum_{j=1}^{n} (x_{ij} - C_i)^2$) where C_i represents the i-th cluster center. The k-means algorithm than follows this steps [29]:

  1. Initialize cluster centroids with random samples;
  2. Assign each observation (x_i) to the nearest cluster center;
  3. Recalculate and update each cluster center (C_i);
  4. Repeat steps 2 and 3 until C_i does not change.

- **Fuzzy C-means:** this algorithm is a generalization of the k-means clustering that permits *soft-segmentation*. Intuitively, the degree to which a pixel belongs to a cluster is inversely proportional to the distance from the pixel to the center of that cluster. In this algorithm the function to minimize is always a distance between each pixel and the cluster's centroids, but it is weighted over the probability of belonging to the determined class [30];

- **Expectation Maximization:** In this approach the pixel intensities of each tissue are supposed to follow a Gaussian distribution. The distribution of the pixel intensities of the whole image is supposed to follow a Gaussian Mixture distribution. The Gaussian Mixture can then be defined as

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{1.1}$$

  where $\{1, 2, ..., K\}$ is the set of mixture components and, for the k-th mixture component, π_k is the mixture weight and $N(x|\mu_k, \Sigma_k)$ is a d-variate Gaussian density, μ_k is the mean vector and is the covariance matrix.

  The Expectation Maximization algorithm follows then this steps [29]:

  1. Initial parameters are somehow initialized (randomly, by k-means or manually given) and then the log likelihood is computed;
  2. The posterior probability is computed using the current parameters (`Expectation` step);
  3. The parameters are recalculated from the current posterior probability (`Maximization` step);
  4. The log likelihood is recomputed;
  5. Repeat step 2, 3, and 4 until a predetermined convergence criterion is satisfied.

1.4.1.3 Neural Networks

Artificial Neural Networks(ANN) are a computational architecture derived from neural physiological models [11]. In this network each node can perform elementary computations, adapting the weight assigned to the connections between nodes gives to the network a property similar to the biological capacity of learning [9]. This brings out the necessity for a training stage, in which the network can compare a set

of features for a pixel with the classification outcome. After this step it is possible to give some new data to the ANN and it will classify them according to the data received during the training step [25].

There are many different architectures for Neural Networks but one of the more used in image segmentation is **Convolutional Neural Network (CNN)** [11]. CNNs are part of the class of Deep Learning Networks. It's typical structure can be divided in *input layer*, which represents the set of data, then quantity of *hidden layers* and finally the *output layer* which provides the results of the segmentation [10]. In this kind of architecture for image segmentation, the hidden layers usually belongs to three categories:

- *Convolutional layers* is made of several convolutional kernels that iteratively convolves local regions of the inputs to generate feature maps. The output of a convolutional layer is a tensor of fetaures. Usually this layer is followed by the application of an activaton function;

- *Pooling layers* which performs subsampling at their input data. For example they can take inputs from a 2×2 unit region in the corresponding feature map (from the convolutional layer) and would compute the average of those inputs. This stage permits to achieve invariance respect small shifts of the image in the corresponding regions of the input space [27]. Pooling layers are usually placed in between two convolutional layers;

- *Fully connected layers* that is a layer in which every possible connection between the input and the output layers can be found. It can be usually find after several convolutional and pooling layers in order to perform high-level reasoning

To train the CNN model certain loss functions over its parameters as to be minimized and until now many functions has been proposed, each for different scenarios. Due to the high correlation between pixels in an image CNN showed high performances in dealing with that data. It is also honorable to be mentioned that recently CNN adaptation to quantum computers starts to be proposed [31].

In 2015 was proposed a CNN architecture specialized for biomedical applications, called **U-Net**. It's main goal is the possibility to be trained also when the training dataset is not large, a condition that often happens in this kind of applications. In order to work with such small number of data it relies on a huge data augmentation. The peculiarity of the UNet is the presence of *skipping connections* which builds a short-cut from a shallow layer to a deep layer by connecting the input of a convolutional block directly to its output. Those skipping connections permits to the network to keep the features learned in the low layers and avoiding performance degradation when adding more layers [32]. The whole structure is divided into two main parts:

- `Encoder`, i.e. contraction path: this path follows the typical architecture of a CNN and consists on the repetition of pooling and convolutional layers. Each couple of pooling and convolutional layers halves the input data and double the number of feature channels;

- `Decoder`, i.e. expansion path: each step is a composition of convolutional layer and up-sampling layer. Every step doubles the number of samples and

halves the number of the feature channels. The expansive pathway combines the feature and spatial information through a sequence of up-convolutions and concatenations with high-resolution features from the contracting path [33]. The name U-Net is due to the resulting shape of its architecture, as it is possible to see in Figure 1.3.

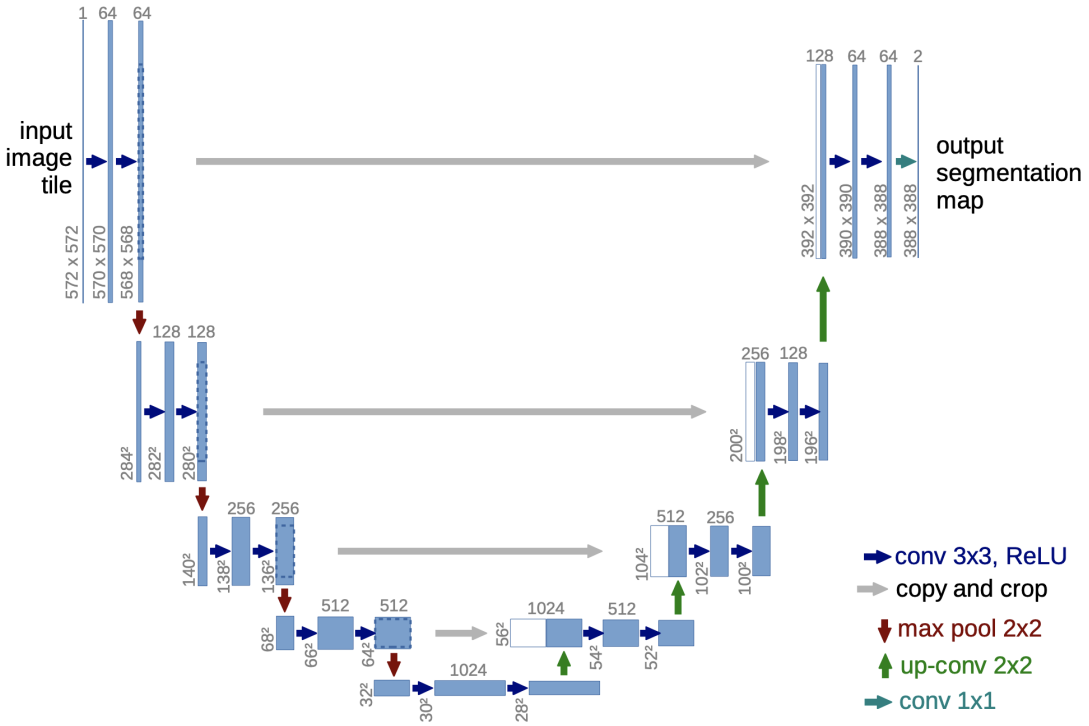


Figure 1.3: *U-Net architecture example. Each blue box corresponds to a multi-channel feature map. White boxes represent copied feature maps. The arrows denote the different operations [33].*

1.4.1.4 Atlas based Segmentation

Another technique consists in have a sort of template already segmented (the atlas) and registering it on the image that is intended to be segmentated. This could lead to a transfer of the labels found on the atlas to the image [25].

An atlas is a segment image of a composite head image formed from co-registered head images of several subjects [25]. In an atlas the pixel intensity could be the simple average of the pixel intensity of the co-registered images, or it can be normalized in some arbitrary way [34]. Another useful atlas can be tissue-type atlas, in which pixel value represents the probability of that pixel to belong to a determined tissue [35].

This technique consists in finding the right registration for the atlas that have to be already segmented for the classes searched. Once the atlas is registered is possible to apply the same transformation to the labels done on the atlas, in this way the labels will be transformed in the image space. As for the other techniques this one can be used standalone but have a great potentiality when pipelined with some others.

For example it is possible to register a probabilistic atlas over an unsegmented volume and using that to select the "a priori" probability for the statistical pattern recognition segmentation, as done in this work of thesis.

Finally it must be mentioned the use of probabilistic atlas had lead to the full segmentation of brain, including subcortical structures [25].

1.5 Performance Evaluation Metrics

Once a pipeline of segmentation is implemented it is recommended to evaluate its performance. In order to do that evaluation in this work of thesis a series of methods and metrics are used. In first place to compute a metric there is the need for understanding what is a *confusion matrix*. Next the main metrics for the classification are reported, such as *precision, recall, specificity* and *FPR*. Then more specific metrics such as *accuracy, dice coefficient, jaccard index, mcc* and *ROC AUC* are discussed.

Confusion Matrix As a first thing it is necessary to define the four cases that is possible to encounter during a binary classification:

- `True Positive (tp)`: an element that has been classified as positive while really being positive;
- `True Negative (tn)`: an element that has been classified as negative while really being negative;
- `False Positive (fp)`: an element that has been misclassified as positive while in reality is negative;
- `False Negative (fn)`: an element that has been misclassified as negative while in reality is positive.

Given those definitions is possible to define the confusion matrix, that is a table that permits the visualization of the performance of classifier. Each row of the matrix represents the instances in an actual class (i.e. the *ground truth*) while each column represents the instances in a predicted class. (In literature is also possible found the opposite, both variant are permitted [36, 37])

Precision is the rate between the true positives and the number of all the elements predicted as positive. Intuitively is the ability to don't misclassify a negative element as positive. In the subsequent Equation 1.2 there is the mathematic definition:

$$precision = \frac{tp}{tp + fp} \tag{1.2}$$

Recall , also known as *true positive rate (TPR)* or *sensitivity* is the rate between the true positives and all the elements that really are positive (Equation 1.3). It can be considered as the ability to correctly classify a positive element.

$$recall = \frac{tp}{tp + fn} \tag{1.3}$$

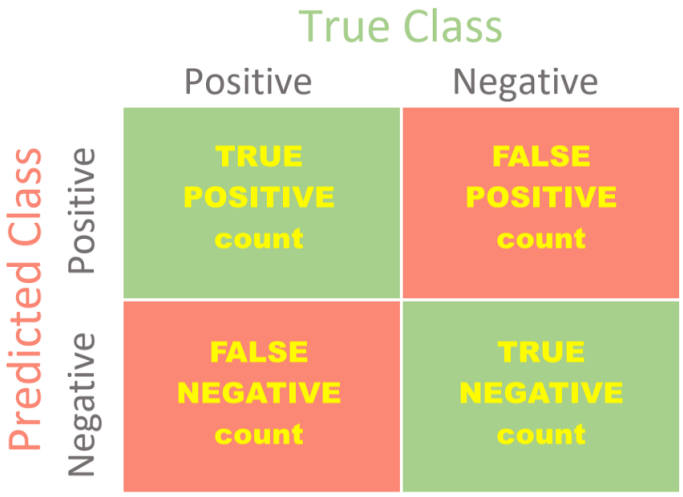


Figure 1.4: *Example of a binary confusion matrix. In this matrix we can find displaced all the outcomes of the classification used. In the rows are visualized the predicted classes and in the columns the actual classes of the incoming data. That permits to have a visual indication of the performance of a classificiation algorithm.*

Specificity is the rate between the true negatives and all the elements that really are negative (Equation 1.4). It can be considered as the ability to correctly classify a negative element.

$$specificity = \frac{tn}{tn + fp} \tag{1.4}$$

False Positive Rate (FPR) is the rate between false positives and the total negatives. As visible in Equation 1.5 the total number of real negatives is the sum the true negatives and the false positives.

$$FPR = \frac{fp}{tn + fp} \tag{1.5}$$

Accuracy and Balanced Accuracy The accuracy is the rate between the number of well estimated (true) outcomes with respects to the total outcomes (Equation 1.6) and it intuitively represents the fraction of well classificated elements with respect to the total number:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{1.6}$$

When the two classes (positive and negative) are not composed of roughly the same amount of data, usually referred as a situation of *unbalanced data*, the accuracy is not a good measure of the goodnes of a classification, due to the low reppresentation of a class in the total amount of the data.

To overcome this issue another metric was defined with the name of balanced accuracy, which is the mean of recall and specificity (Equation 1.7).

$$\text{balanced accuracy} = \frac{1}{2}\left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp}\right) \tag{1.7}$$

Dice Similarity Coefficient / F1 score is used to measure the similarity of two samples. In the case of two images the similarity can be thought as the capability of the two images to overlay one on the other. It is the double of the rate between the cardinality (number of elements in the set) of the intersection between two sets and the sum of the cardinalities of the single sets (Equation 1.8). It is intuitively that the best value is 1 (total overlapping) and the worst value is 0 (no overlapping). Given two sets named X and Y the following is the mathematical definition:

$$DSC = 2\frac{|X \cap Y|}{|X| + |Y|} \tag{1.8}$$

When applied to boolean data it is often referred as *F1 score* and is defined as the harmonic mean of precision and recall (Equation 1.9).

$$F1 = \frac{2}{precision^{-1} + recall^{-1}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2tp}{2tp + fp + fn} \tag{1.9}$$

While not immediately obvious at a first sight, the proof of the equality of those equations is reported in Appendix A.

Jaccard index , also known as Jaccard similarity coefficient, is a metric used to evaluate overlapping between two sets. It is defined as the rate between the cardinality of intersection of the sets and the cardinality of the union of the same sets (Equation 1.10). Given X and Y as the sets to evaluate:

$$J = \frac{|X \cap Y|}{|X \cup Y|} \tag{1.10}$$

Matthews correlation coefficient (MCC) , so called because was firstly used by Matthews in 1975 [38], is a value that ranges between -1 and $+1$ and can be applied to multiclass classification. Here it is discussed and defined (Equation 1.11) only the case of binary classification. The $+1$ result means a total agreement while a -1 means total disagreement. If the result is 0 than the prediction is totally random [39].

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{1.11}$$

Receiver Operating Characteristic (ROC) curve Considering a soft segmentation output in which the output is continue, to determine the classification (hard segmentation) it is necessary to apply a threshold to the outcomes of the segmentation and tp, tn, fp and fn will vary with the threshold. Usually there is a trade off between fp and fn produced by the algorithm. ROC curve visualize this result plotting the recall in function of the false positive rate while varying the threshold [39].

We can distinguish two extreme curves:

1. a line with 45 of inclination, representing a random classifier with no benefit;
2. a broken line rising from the origin of axis to the point (0,1) and then ending at (1,1) meaning a perfect classifier.

Precision-Recall Curve Similarly to the ROC curve, the precision-recall curve plot the trade off between the precision and the recall at the variation of the decision threshold. This curve returns a better visualization for the goodness of a classification when the data are unbalanced, i.e. one of the two classes is less represented in the considered sample than the other [40].

While the perfect classifier is represented by an horizontal line starting from the point $(1, 1)$, reaching the point $(1, 1)$ and then falling to the point $(1, 0)$, a random classifier is represented by an horizontal line at the height equal to the rate between the positive and negative class.

Chapter 2

Material and Methods

This work aims at the development and implementation of the pre-processing and post-processing stages of a segmentation pipeline of Silent Cerebral Infarctions in brain MRI of patients affected by Sickle Cell Disease.

The segmentation is carried out by a U-Net ensemble [5] that works on the single slices in the axial direction and it requires standardized images as input. The developed preprocessing phase aims the data normalization, with the batch effect removal. To perform this tasks the brain was isolated from the head images obtained and was next segmented into its main tissues.

It was observed that the U-Net ensemble tends to identify as SCI in some areas which in reality are false positives. The aim of the post-processing phase is the identification and removal of these areas.

This Chapter will firstly report a short review on the analysed dataset. Then the pipelines of pre-processing and post-processing will be discussed in details, explaining also how they were implemented.

2.1 Materials

The main dataset used for this work of thesis was provided by three different medical centers in italy and was acquired in the contest of the european project *Genomed4All*. Those centers was respectively located in the cities of Padua, Naples and Genoa. The dataset was composed of MRI scans, both T1W and FLAIR for each patient, in Nifti format, recorded in the time range from 2009 to 2020. The provided dataset has a high heterogeneity about acquisition systems used and spatial resolution as it possible to see in Table 2.1. While all the images have the same size (256×256) on the transverse plane (xy-plane) on the z-axis the size ranges from a minimum value of 22 to a maximum value of 512. Also the acquisition parameters have a wide range of values. The minimum *echo time* used is of $2.958ms$ and a maximum one of $344.578ms$. The *repetition time* instead ranges from $8.456ms$ to $11000.0ms$.

The images come from 57 patients of which 51 have evidences of SCIs and the other 6 without this kind of lesion. The volumes were manually labeled by expert clinicians from the University of Padua. The labels provided consist in a binary segmentation of the SCIs' volume.

Genua's and Padua's centers provided also general informations about the age and the sex of the patients and in Table 2.2 is possible to appreciate that the dataset consists mainly of underarged patients.

Axis	Size (pixel)		Spacing (mm)	
	Mean	Std. Dev.	Mean	Std. Dev.
x	256	0	0.85	0.12
y	256	0	0.81	0.14
z	90	132	4.34	1.93

Table 2.1: *In this table are reported the the mean and the standard deviation for the size and the spacing on the three main axis of the images of the entire data set. The data reported in this table concern the scans from all the centers (Naples, Genua and Padua).*

Medical Center of Provenience	Age		Sex		Number of Patients
	Mean	Standard Deviation	M	F	
Padova	11.18	5.78	16	23	39
Genova	39	14.42	4	3	7

Table 2.2: *Patient dataset description divided by center of provenience. It is possible to appreciate the distribution of ages, sex. The Naples center is omitted because no informations were provided.*

2.2 Pre-Processing

The starting point for the Silent Cerebral Infarcts' segmentation was the T1 weighted and FLAIR head MR scans. The approaches here explained are aimed to process the T1W images only. It is therefore possible to transfer all the masks obtained registering the T1W images over the FLAIR images using a rigid transformation in order to align the two images and to permit that every mask obtained can be transferred on the FLAIR images also. In Figure 2.1 is reported the full flowchart of the pre-processing pipeline described below.

2.2.1 Description

The first need was to remove from the images everything except the brain, in a step called *brain extraction* (or also *skull stripping*). Once this task was done the challenge has been to segment the different brain tissues (white matter, grey matter and cerebrospinal fluid), with special regard toward the white matter, because it is the tissue where SCIs can be found.

The techniques developed for both the brain extraction and the segmentation are atlas based and relies on the use of an already segmented atlas. The pipeline is designed to work with every atlas that contains at least a brain mask and a soft segmentation of the three tissues cited, but during the developing of this work of thesis was used in particular the ICBM MNI 152 non linear symmetric atlas 2009a which have isotropic voxels of size $1mm^3$ and a total size of $106 \times 232 \times 188$ [41].

2.2.1.1 Brain Extraction

The process of brain extraction consists in the separation of the brain from the non-brain tissues present in the head. This process is often called also *skull-stripping* even if the two meanings are not totally interchangeable [42], it is common to use

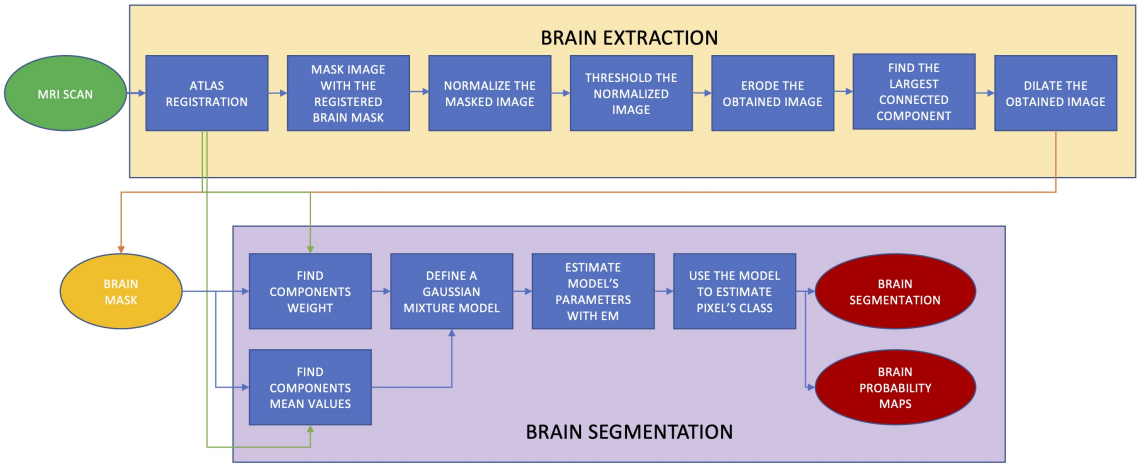


Figure 2.1: *Workflow of the developed preprocessing pipeline. In this flowchart are included both the brain extraction stage and the segmentation stage. The output from the first stage is linked as input of the second stage since only the extracted brain was used during the segmentation process. The atlas registration is linked to the segmentation stage because the probability maps of the atlas was registered in the brain space using the transformation parameters obtained during the registration of the atlas in the brain extraction step.*

both the terms to indicate the same process and in this work both the terms will be used.

This is a common pre-processing step in neuroimaging as it permits to focus only to the spatial volume where the researched features physically are. Due to high variability that is possible to find between each image, caused by differences in image acquisition or inter-patient variability the automatic brain extraction is not a easy task.

The technique here proposed is intended to be totally automatic and computationally fast even on a domestic pc. It relies on the use of an atlas and the corresponding binary brain mask. In that paragraph all the steps that brought to a proper brain extraction are explained.

Registration: MNI152 atlas was registered onto the T1W image using a multiodal and multiresolution registration. The first transformation used was a *rigid* transformation to roughly align the atlas on the T1 image. The so obtained image was then transformed with an *affine* transformation to permit to the moving image to scale and skew to reach a better alignment with the fixed image. In the last step an elastic *BSpline* transformation was applied, to permit the deformation necessary to make the two images as overlapping as possible. In all those three transformations the metric to minimize was the *mutual information* which allows fast computation of the metric value [43] and permits to match the images even when the values of the matched pixel are not the same. Morevoer, according with the last applied transformation, the chosen interpolator was a BSpline function in order to have a higher precision in the final pixel values, since the computation time is not affected in a significant way. The final result is visible in Figure 2.2.

Masking: The mask obtained from the registration was then thresholded and dilated with a spherical kernel of radius 1. The brain mask obtained was then used to mask the image in order to have a first roughly extracted brain. The dilating process was necessary in order to obtain a more conservative extraction of the brain and assuring that all the regions of interest were conserved.

Thresholding: That allows to reach a rough head segmentation in which only the brain and part of the skull are visible. The image obtained in such way was then normalized. The normalization was done calculating the mean value an the standard deviation for the pixels under the mask obtained in the precedent step and then shifting the pixel values in order to have the mean value at 0 and scaling the values of 1σ. The normalized image was then thresholded with a multi-otsu algorithm in order to exclude the background and obtain a binary image. The thresholded image was used as a mask on the original head image. The obtained masked image was then normalized and thresholded again with a fixed threshold. Those two thresholds were necessary to have a better control in removing the unwanted regions. The normalization process was necessary in order to have the possibility of use a fixed threshold.

Finding the Largest Connected Region: The result from the last step was a binary image in which the gap between the brain and the skull was partially considered background. In order to have a proper separation of the skull from the brain an eroding filter with a spherical kernel of radius 1 was applied. At this point, having the brain totally divided from the skull, the largest connected region in the image was found, resulting in the brain.

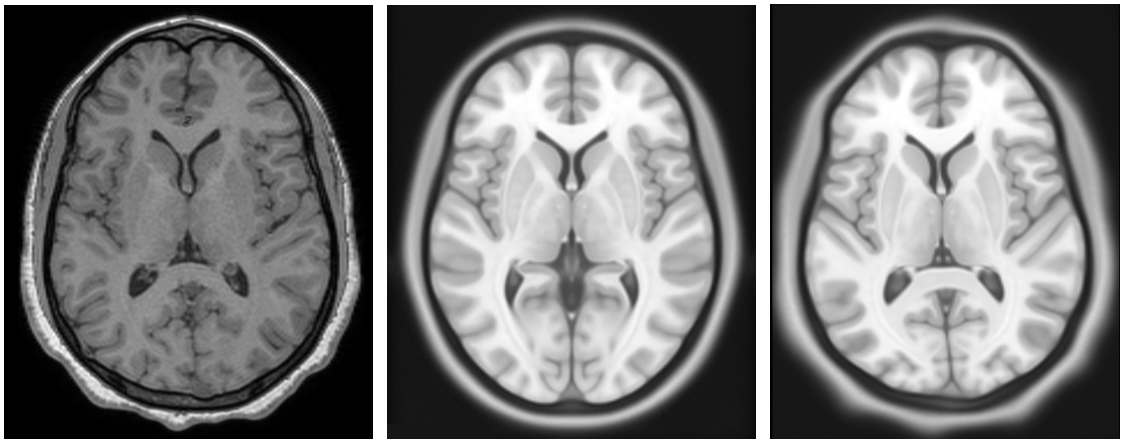
Refinition: The so obtained mask was finally dilated with a spherical kernel of radius 2, in order to regain the loss volume in the eroding applied for the purpose of fully divide the skull from the brain.

In this way was possible to obtain an algorithm that compute the brain mask for a T1W MRI scan like the ones visible in Figure 2.3 and Figure 2.4 with times in the order of few minutes, even using a normal domestic pc.

2.2.1.2 Tissue Segmentation

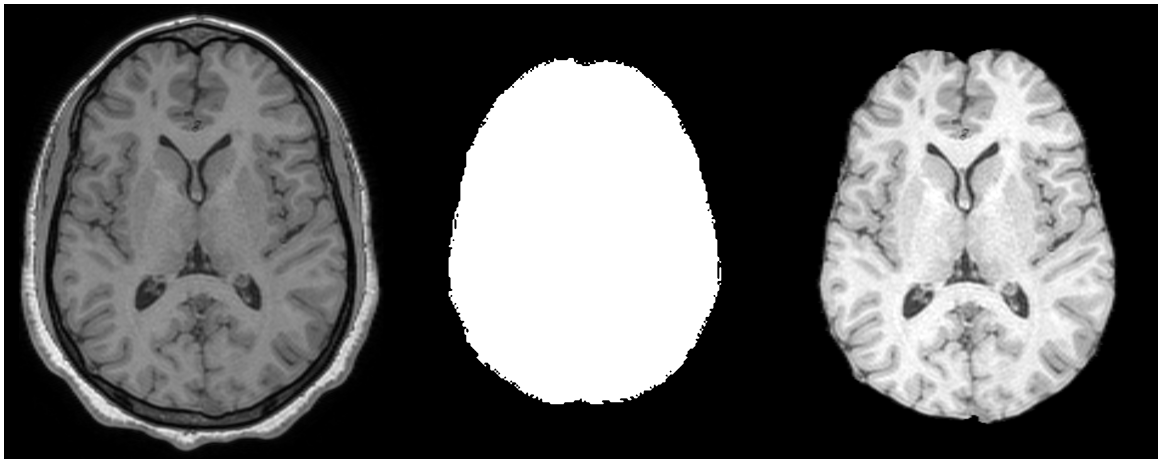
Since the SCIs are found only in the white matter, a proper segmentation of the brain tissues has a great value for their segmentation. For the aims of this work of thesis the algorithm is intended to work with relatively short time even on a normal domestic personal computer.

The main idea behind the approach here proposed was to consider the pixel values of the brain distributed as a mixture of Gaussians distributions, as visible in Figure2.5 and then use an Expectation Maximization (EM) algorithm to estimate the parameters for those distribution functions. The EM algorithm requires some parameters for it initialization, such as the number of classes, the estimated mean value for each class and the weight of each class on the total image. The potentiality of having the brain already extracted was to have the background already segmented and therefore the number of classes was set to 3 as the algorithm is intended to segment the main tissues of the brain (white matter, grey matter, cerebrospinal fluid).



(a) *Fixed Image* **(b)** *Original Moving Image* **(c)** *Registered Moving Image*

Figure 2.2: *T1W atlas registration over a T1W image. Figure a is the original head scan. Figure b is the MNI 152 [41] atlas before the registration process. Figure c is the atlas scan after the registration and is possible to see how the atlas was deformed by the algorithm to match the fixed image.*



(a) *Original Image* **(b)** *Computed brain mask* **(c)** *Extracted brain*

Figure 2.3: *In these images is possible to see the comparison between an original head image (a), the brain mask computed (b) and the brain extracted with that mask (c).*

The other parameters were estimated relying on the registration of the probability maps for the tissue segmentation of the atlas on the images. This approach permits an initialization more accurate and focused for each analysed volume.

In the first place the probabilistic tissue maps of the atlas were transformed using the same transformation obtained for the atlas registration during the skull stripping. Once the map were transformed each pixel was assigned to the class with the higher probability. This process permitted to have a rough preliminary segmentation of the tissues and this information was used to compute the estimate for the class weight and the mean value of the pixels of every tissue.

It is to mention that the expectation maximization algorithm lead to a soft segmentation, giving a probability map for every class in output. To compute the hard segmentation was used the same approach as before: every pixel was assigned

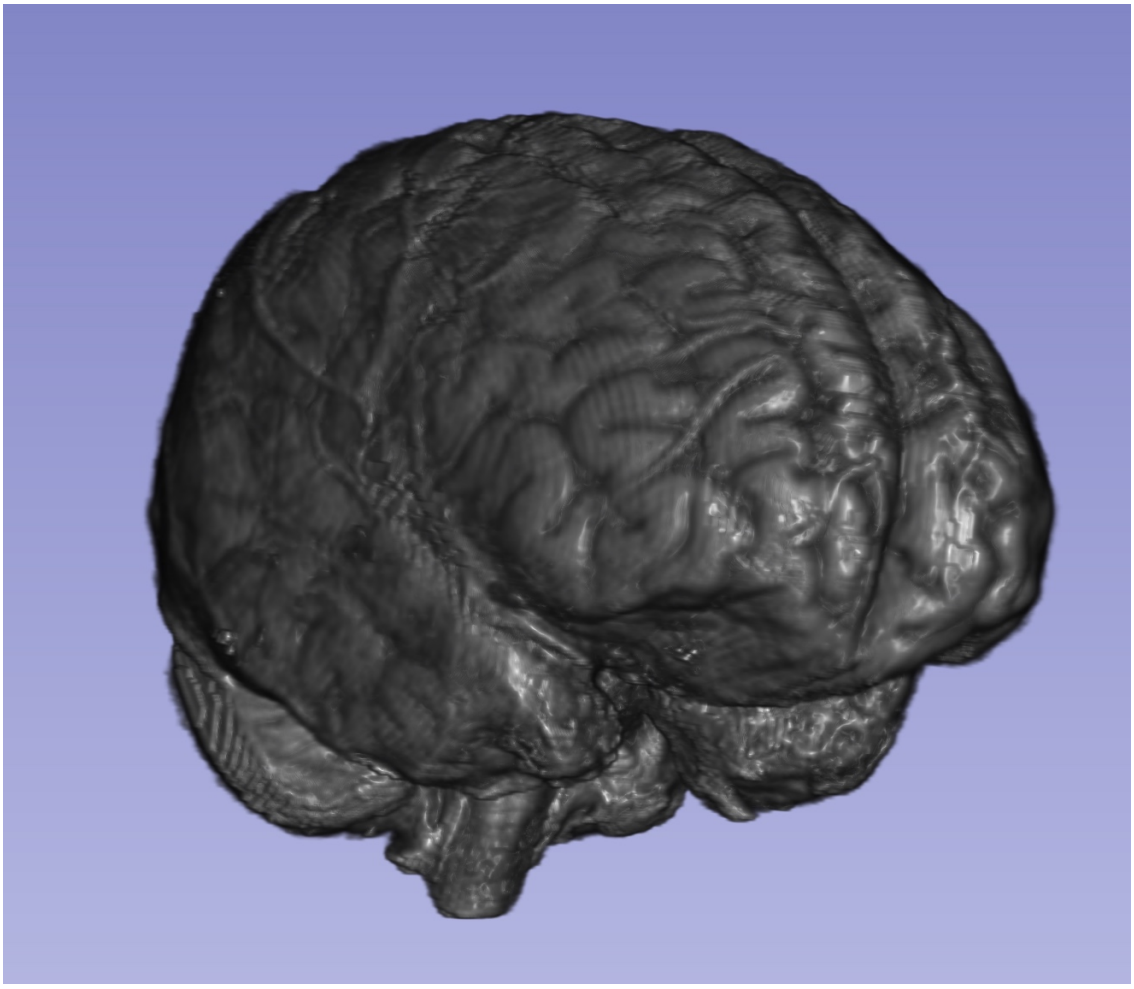


Figure 2.4: *Using specialized applications is possible to render a 3D image of the volumes permitting to have a better looking to the external structures of the obtained object. The one here proposed let see the extracted brain with the proposed algorithm. The image is the same of Figure 2.3. This 3D rendering is computed with the application 3DSlicer [44].*

to the class of greater probability. The results of the segmentation are visible in Figure2.6.

2.2.2 Implementation

The two pipelines described before was implemented using Python, which is an high level object oriented programming language. To perform all the essential operations (input/output, filtering, morphological operations, etc.) two main libraries were used: ITK [45] and Numpy [46]. To perform the registration and to apply the obtained transformations the ITK-Elastix [47] library was used, which is an ITK Python interface to Elastix [22]. To implement the brain segmentation the Scikit-Learn[48] library was used.

The whole code is opensource and freely available con GitHub [49] and the installation is managed by a setup.py file which also provides the full list of dependencies. During the developement, the installation was automatically tested on Ubuntu for different python versions. This continuos integration process was carried out using

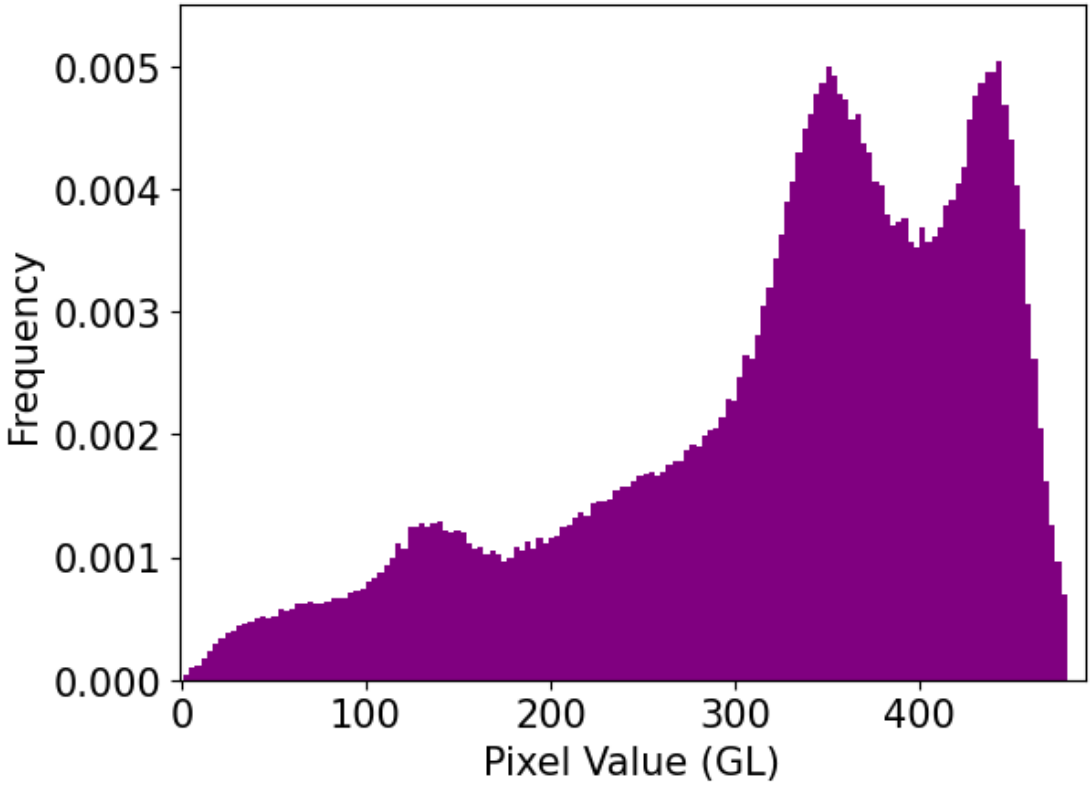
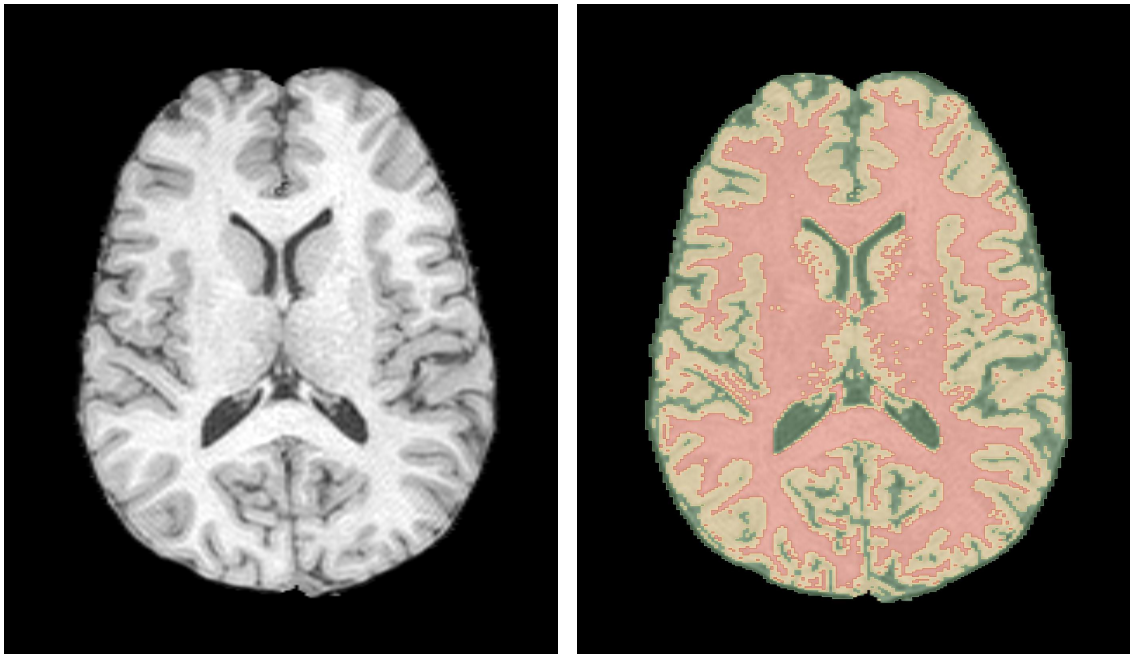


Figure 2.5: *In the figure is possible to see the histogram of an extracted brain from a T1W scan. Looking at the pixel value distribution (expressed as Grey Level since it is a monochromatic image) is possible to appreciate how to model the pixel distribution as a mixture of Gaussians with different means (identifiable with the three observable peaks) and different standard deviations.*



(a) *Brain extracted from a T1W scan.* **(b)** *Brain segmentation result.*

Figure 2.6: *Segmentation process results on an already extracted brain. Figure a is the brain extracted from the scan. In Figure b is is possible to observe the result of the segmentation process in which the white matter is labeled in red, the grey matter in yellow and the cerebrospinal fluid in green. The masks are obtained from a Gaussian mixture model with three classes which parameters were estimated using an expectation maximization algorithm.*

the github actions.

The whole code is subdivided in three modules, respectively dedicated to registration, skull stripping and segmentation and is intended to work with Nifti images.

2.2.2.1 Brain Extraction

In the last years many methods were proposed. Some of the more widely used examples of brain extraction methods are *Brain Extraction Tool (BET)* from the FSL library, *Brain Surface Extraction (BSE)* which is part of BrainSuite and *Robust Learning-based Brain Extraction System (ROBEX)* and many others [42]. Each of the cited methods have a different approach to the skull stripping problem, ranging from low level morphological operations to the classification using pattern recognition.

The brain extraction process proposed in this work of thesis relies on a registration of an atlas over the image as explained before, so both the registration module and the skull stripping module are involved in this part of the implementation.

The registration was done using this function based on Elastix library. The images were all read with the ITK I/O functions with floating pixels as required by Elastix.

Listing 2.1: Multimap Registration Function

```
1 import itk
2
```

```
def elastix_multimap_registration(fixed_image, moving_image):

    # Setting our number of resolutions
    parameter_object = itk.ParameterObject.New()
    resolutions = 4

    #Import RIGID parameter map
    parameter_map_rigid = parameter_object.GetDefaultParameterMap(
    'rigid', resolutions)
    parameter_map_rigid['Metric']        = [
                'AdvancedMattesMutualInformation']
    parameter_map_rigid['Interpolator'] = [
    'BSplineInterpolatorFloat']
    parameter_object.AddParameterMap(parameter_map_rigid)

    #Adding an AFFINE parameter map
    parameter_map_affine = parameter_object.GetDefaultParameterMap(
    "affine", resolutions)
    parameter_map_affine['Metric']        = [
    'AdvancedMattesMutualInformation']
    parameter_map_affine['Interpolator'] = [
    'BSplineInterpolatorFloat']
    parameter_object.AddParameterMap(parameter_map_affine)

    #Adding a NON-RIGID parameter map
    parameter_map_bspline = parameter_object.GetDefaultParameterMap
    (
    "bspline", resolutions)
    parameter_map_bspline['Interpolator'] = [
    'BSplineInterpolatorFloat']

    parameter_object.AddParameterMap(parameter_map_bspline)

    #Set the registration
    elastix_object = itk.ElastixRegistrationMethod.New(
                                            fixed_image,
    moving_image)
    elastix_object.SetParameterObject(parameter_object)

    #Start the registration
    elastix_object.UpdateLargestPossibleRegion()

    return elastix_object
```

At the end of the registration and object containing both the registered image and the transformation applied was returned. This function is then inserted in the the main brain extraction pipeline.

The implemented pipeline use different functions that are aimed to perform the different steps described in the previous sections. The first needs encountered was to have a simple interface for the application of the used ITK filters, as threshold and morphological operations and their name is always self explanatory.

One of the most important step in this brain extraction pipeline is the rough masking done with the registered atlas' brain mask. Due to the ITK installation on python, in which not every filter are wrapped from the original C++ language, the masking functions was not always available for the used types. To bypass the problem a dedicated function that explicitly set a 0 value for every pixel outside the

mask was implemented with the name of *negative_3d_masking*.

One of the important processes needed to explain in this section is the normalization used to permit the using of a fixed threshold. It permits the normalization of the pixels values masked by a certain mask which in this case is the brain mask obtained in the first steps, permitting to not have the background pixel's value to account for the normalization process.

The last step in the brain extraction is the function that permits to find the largest connected region in the image. This is the step in which the proper brain is extracted. The function implemented simply found every connected region in the image and then sort them by their dimension, finally keeping only the greater ones.

Listing 2.2: Largest Connected Region Finder

```
import itk

def find_largest_connected_region (image):

    #searching for connected regions in the images
    connected_filter = itk.ConnectedComponentImageFilter[OutputType
    , OutputType].New()
    connected_filter.SetInput(cast_filter.GetOutput())
    connected_filter.Update()

    #relabeling the images in increasing order for decreasing
    dimension
    label_filter = itk.RelabelComponentImageFilter[OutputType,
    OutputType].New()
    label_filter.SetInput(connected_filter.GetOutput())
    label_filter.Update()

    #tengo solo la regione connessa pi  grande
    thresholdFilter = itk.ThresholdImageFilter[OutputType].New()
    thresholdFilter.SetInput(label_filter.GetOutput())
    thresholdFilter.ThresholdOutside (1,1)
    thresholdFilter.SetOutsideValue(0)
    thresholdFilter.Update()

    return thresholdFilter.GetOutput()
```

The pipeline for the brain extraction is then implemented as follows.

Listing 2.3: Brain Extraction Function

```
 import itk
 import numpy as np

 def skull_stripping_mask (image, atlas, mask,
  transformation_return = False):

   reg_atlas_obj = elastix_multimap_registration( image, atlas )

   #apply the transformation to the mask and then make it again a
   binary image
   reg_mask = itk.transformix_filter( mask, reg_atlas_obj.
   GetTransformParameterObject() )
   bin_reg_mask = binarize( reg_mask )

   #dilating the mask to be more conservative
```

```
13      bin_reg_mask = binary_dilating(bin_reg_mask)
14
15      #do a first skull stripping
16      first_brain = negative_3d_masking( image, bin_reg_mask )
17
18      #binarize the first_brain to obtain a mask useful for the
        normalization
19      first_brain_mask = binarize( first_brain )
20
21      #Casting the first_brain_mask to be used by the normalization
        filter
22      OutputType = itk.Image[itk.UC, 3]
23      cast_filter = itk.CastImageFilter[type(first_brain_mask),
        OutputType].New()
24      cast_filter.SetInput(first_brain_mask)
25      cast_filter.Update()
26      first_brain_mask = cast_filter.GetOutput()
27
28      #normalize the first obtained brain
29      normalized_first_brain = itk_gaussian_normalization(
        first_brain, first_brain_mask )
30      normalized_first_brain.Update()
31
32      #Normalized image must be casted
33      OutputType = itk.Image[itk.D, 3]
34      cast_filter = itk.CastImageFilter[type(normalized_first_brain.
        GetOutput() ), OutputType].New()
35      cast_filter.SetInput(normalized_first_brain)
36      normalized_first_brain = cast_filter.GetOutput()
37
38      #MultiOtsu thresholding the normalized_brain
39      InputType = type(normalized_first_brain)
40      motsu = itk.OtsuMultipleThresholdsImageFilter[InputType,
        InputType].New()
41      motsu.SetInput(normalized_first_brain)
42      motsu.SetNumberOfThresholds(3)
43      motsu.Update()
44      thresholded_first_brain = binarize(motsu.GetOutput(), hi_value
        = 3 )
45
46      #creating a second brain
47      second_brain = negative_3d_masking( image,
        thresholded_first_brain )
48
49      #Casting the thresholded_first_brain to be used by the
        normalization filter
50      OutputType = itk.Image[itk.UC, 3]
51      cast_filter = itk.CastImageFilter[type(thresholded_first_brain)
        , OutputType].New()
52      cast_filter.SetInput(thresholded_first_brain)
53      cast_filter.Update()
54
55      #normalize the second obtained brain
56      normalized_second_brain = itk_gaussian_normalization(
        second_brain, cast_filter.GetOutput() )
57      normalized_second_brain.Update()
58
59      thresholded_second_brain = binarize( normalized_second_brain.
```

```
    GetOutput(), -3, 1.5 )
60
61      #eroding the mask to better find the largest connected region
62      eroded_mask = binary_eroding( thresholded_second_brain )
63
64      #find the largest connected region
65      first_mask = find_largest_connected_region( eroded_mask )
66
67      #apply a dilation to the mask and a hole filler
68      final_mask = hole_filler( binary_dilating(first_mask, 2) )
69
70      print('Your brain mask is ready!')
71
72      if transformation_return == False:
73          return final_mask
74      else:
75          return final_mask, reg_atlas_obj.
    GetTransformParameterObject()
```

2.2.2.2 Tissue Segmentation

As for the brain extraction there many software that permits the segmentation of the brain tissues. The approach here proposed to segment the brain tissues the code was based on the use of a Gaussian Mixture function of the Scikit-learn library. The image has to be given in input to the Gaussian Mixture function as a flatten array, so there were the need to flatten the image having the possibility to rebuild it once it was segmented. This need of giving to the Scikit-learn function a flatten array arises because it takes in input an array of dimension $Nsamples \times Nfeatures$ in which, for our application, the samples are the pixels and the only feature given was the pixel value.

In order to do this flattering two functions were implemented, the first one to flatten the image and the second to rebuild the image once it was segmented. The main idea was to create an array with only the pixels under the brain mask. This permits to limit the number of pixels that the algorithm had to classify and to don't have to model the background pixels. This lead to the necessity of save the original index of every pixel to have a correct reconstruction at the end of the segmentation.

Listing 2.4: Indexing functions

```
1  import itk
2
3  def indexing (image, mask):
4      Dimension = 3
5      index = itk.Index[Dimension]()
6
7      image_array = [] #The array with the grey values of the masked
       pixels
8      index_array = [] #The array with the itk indexes of the pixels
9
10     for index[0] in range( image.GetLargestPossibleRegion().GetSize
       ()[0] ):
11         for index[1] in range( image.GetLargestPossibleRegion().
       GetSize()[1] ):
12             for index[2] in range( image.GetLargestPossibleRegion()
       .GetSize()[2] ):
```

```
    #Only if the pixel is under the mask then the function will take that pixel
                if mask.GetPixel(index) != 0:
                    image_array.append( image.GetPixel(index) )
                    index_array.append( [ index[0], index[1], index[2] ] )

    return image_array, index_array



def label_de_indexing (image_array, index_array, reference_image, first_label_value = 0 ):
    Dimension = 3
    ImageType = itk.template(reference_image)[1]

    #Creation of the new itk image
    image = itk.Image[ImageType].New()

    #Creation of the itk Index object
    index = itk.Index[Dimension]()

    #Setting the new image space as the one of the original (reference) image.
    image.SetRegions( reference_image.GetLargestPossibleRegion() )
    image.SetSpacing( reference_image.GetSpacing() )
    image.SetOrigin( reference_image.GetOrigin() )
    image.SetDirection( reference_image.GetDirection() )
    image.Allocate()

    #create a black image
    for index[0] in range( image.GetLargestPossibleRegion().GetSize()[0] ):
        for index[1] in range( image.GetLargestPossibleRegion().GetSize()[1] ):
            for index[2] in range( image.GetLargestPossibleRegion().GetSize()[2] ):
                image.SetPixel(index, 0)

    for i in range(len(index_array)):
        #Set the itk index as the i_th index of the index_array
            index[0] = int( index_array[i][0] )
            index[1] = int( index_array[i][1] )
            index[2] = int( index_array[i][2] )

        #Set the Pixel value of the image as the one in the array
            image.SetPixel( index, image_array[i] + first_label_value )

    return image
```

As previously discussed the aim of the proposed pipeline was to automatically segment the image, this result in the need of functions that can estimate the initial parameters of the Gaussian Mixture. The estimation of those parameters was done using the probability maps of the used atlas registered in the space of the image. Two functions were then implemented to permit to find the relative weight of each mask and the mean pixel value for each class.

Listing 2.5: Parameters Estimation

```
import itk
import numpy as np

def find_prob_weights (wm_mask, gm_mask, csf_mask):

    #Getting arrays from masks
    wm_array = itk.GetArrayFromImage(wm_mask)
    gm_array = itk.GetArrayFromImage(gm_mask)
    csf_array = itk.GetArrayFromImage(csf_mask)

    tot_array = wm_array + gm_array + csf_array

    four_dim_array = [wm_array, gm_array, csf_array]

    #finding for every pixel which is its most probable type
    prob_array = np.argmax(four_dim_array, 0)

    #finding number of pixels for gm and csf
    gm_pixels = np.count_nonzero(prob_array == 1)
    csf_pixels = np.count_nonzero(prob_array == 2)

    #finding the total number of pixels
    tot_pixel = np.count_nonzero( tot_array )

    if tot_pixel == 0 :
        #In this case the masks are empty images.
        return [0,0,0]

    wm_pixels = tot_pixel - gm_pixels - csf_pixels

    #finding weights for the masks
    wm_weight = wm_pixels/tot_pixel
    gm_weight = gm_pixels/tot_pixel
    csf_weight = csf_pixels/tot_pixel

    #creating a list with all the weights.
    weights = [wm_weight, gm_weight, csf_weight]

    return weights



def find_means (brain, brain_mask, csf_mask, gm_mask, wm_mask):

    #converting everything in numpy array. This is done because ITK
    functions in python are not always wrapped for every type.
    brain_array = itk.GetArrayFromImage(brain)
    wm_array = itk.GetArrayFromImage(wm_mask)
    gm_array = itk.GetArrayFromImage(gm_mask)
    csf_array = itk.GetArrayFromImage(csf_mask)
    brain_mask_array = itk.GetArrayFromImage(brain_mask)

    #setting a greater value for bg in order to have it correctly
   selected in prob array
    background = np.where(brain_mask_array == 0, 2, 0)

    #finding for each pixel what is its more probable class (bg =
```

```
    0, wm = 3, gm = 2, csf = 1)
56      four_dim_array = [background, csf_array, gm_array, wm_array]
57      prob_array = np.argmax(four_dim_array, 0)
58
59      #recreating the pixels array
60      wm_array = np.where(prob_array == 3, 1, 0)
61      gm_array = np.where(prob_array == 2, 1, 0)
62      csf_array = np.where(prob_array == 1, 1, 0)
63
64      #creating arrays in which there are only pixels of the brain
    where those pixels are of that class
65      wm_brain = np.where (wm_array == 1, brain_array, 0)
66      gm_brain = np.where (gm_array == 1, brain_array, 0)
67      csf_brain = np.where (csf_array == 1, brain_array, 0)
68
69      #find means
70      wm_mean = np.sum(wm_brain)/np.count_nonzero(wm_array)
71      gm_mean = np.sum(gm_brain)/np.count_nonzero(gm_array)
72      csf_mean = np.sum(csf_brain)/np.count_nonzero(csf_array)
73
74      means = [csf_mean, gm_mean, wm_mean]
75
76      return means
```

The implementation of the brain segmentation is then provided. It is to mention that the provided function permits to return both a single image with the labels for each tissue or a set of three images containing the probability map of each tissue.

Listing 2.6: Segmentation pipeline

```
1  import numpy as np
2  from sklearn.mixture import GaussianMixture
3
4  def brain_segmentation ( brain, brain_mask, wm_mask, gm_mask,
    csf_mask, auto_mean = False, undefined = False, proba = False ):
5
6       #casting of the mask for the normalization.
7       OutputType = itk.Image[itk.UC, 3]
8       cast_filter = itk.CastImageFilter[type(brain_mask), OutputType
    ].New()
9       cast_filter.SetInput(brain_mask)
10      cast_filter.Update()
11      brain_mask = cast_filter.GetOutput()
12
13      #brain normalization
14      norm_brain_filter = itk_gaussian_normalization (brain,
    brain_mask)
15
16      #I change the physical space because the normalization have
    changed it.
17      matching_filter = match_physical_spaces(norm_brain_filter.
    GetOutput(), brain)
18      matching_filter.Update()
19      brain = matching_filter.GetOutput()
20
21      #mask back to float
22      cast_filter = itk.CastImageFilter[type(brain_mask), OutputType
    ].New()
23      cast_filter.SetInput(brain_mask)
```

```
    cast_filter.Update()
    brain_mask = cast_filter.GetOutput()


    #linearize and indexing the brain obtaining the image array and
    the index array
    #(the index array is useful to build the itk label image)
    brain_array, index_array = indexing (brain, brain_mask)


    #Masking also the Probability maps
    wm_mask = negative_3d_masking(wm_mask, brain_mask)
    gm_mask = negative_3d_masking(gm_mask, brain_mask)
    csf_mask = negative_3d_masking(csf_mask, brain_mask)


    #INITIALIZING THE MODELS PARAMETERS
    if  auto_mean : #if automean the proposed function will be used
    to find the means values

        if undefined :
              means = np.reshape(find_4_means(brain, csf_mask,
   gm_mask, wm_mask), (-1,1))

        else: means = np.reshape(find_means(brain, brain_mask,
   csf_mask, gm_mask, wm_mask), (-1,1))

        #Adding a check for the mean values.
        #We should obtain mean values sufficiently different and in
     order csf < gm < wm.
        #if not then the mean values will be initialized by the k-
   means++ algorithm of Scikit-Learn
        if ( (means[0]+0.1) >= means[1] ) or ( (means[1] + 0.1) >=
   means[2] ) :
             means = None

    else: means = None #The k-means++ algorithm of Scikit-Learn
   will be used to initialize the means

    if undefined:
        n_classes = 4
        weights = find_prob_4_weights(csf_mask, gm_mask, wm_mask)
    else:
        n_classes = 3
        weights = find_prob_weights(csf_mask, gm_mask, wm_mask)

    #Definition of the models
    model = GaussianMixture(
                n_components = n_classes,
                covariance_type = 'full',
                tol = 0.01,
                max_iter = 10000,
                init_params = 'k-means++',
                means_init = means,
                weights_init = weights
                )

    #updating the funtion to find the labels
```

```
74      model.fit( np.reshape( brain_array, (-1,1) ) )
75
76      #Return 3 probability masks
77      if proba:
78          label_array = model.predict_proba( np.reshape( brain_array,
        (-1,1) ) )
79          wm = label_de_indexing (label_array[:,2], index_array,
        brain)
80          gm  = label_de_indexing (label_array[:,1], index_array,
        brain)
81          csf  = label_de_indexing (label_array[:,0], index_array,
        brain)
82
83          return wm, gm, csf
84
85      else:
86          label_array = model.predict( np.reshape( brain_array,
        (-1,1) ))
87          label_image = label_de_indexing (label_array, index_array,
        brain, 1.)
88          return label_image
```

2.3 Post-Processing

Post processing stage aims to improve the goodness of the SCIs segmentation obtained with the U-Net implemented, optimizing the full pipeline. In the processing step it was discovered that the implemented neural network has the tendency to obtain false positives, i.e. to find many false lesions. This use to happen in the regions near the brain ventricles because their wall can appear as hyperintense in FLAIR images or in some areas in the grey matter. The proposed pipeline is therefore intended to remove the wrong segmented regions and its structure is summarized in Figure 2.7.

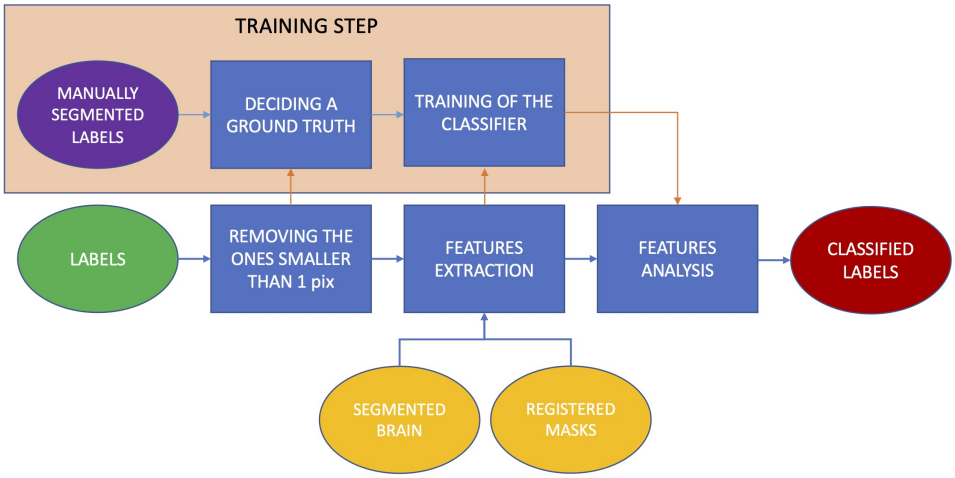


Figure 2.7: *Workflow of the developed post-processing pipeline. In the flowchart is reported also the training of the classifiers which will be necessary only in the developing step. Once the classifiers are trained they can be used straightly on the new scans proposed.*

2.3.1 Description

The post processing pipeline has to classify the labels obtained by the U-Net ensemble as true or false.

Traditionally, SCI must be greater that $3mm$ in greatest linear dimension [4] and visible in at least two planes [50]. Since the MRI has been improved and the 3D imaging became more often used this definition no longer holds strictly, but assure that, on the time this work of thesis was done, SCI's label can't be too small. For those reason the first operation done on the results of the automatic segmentation was to remove all the labels smaller than two voxels. Even if the scans of the data set has an high variability in term of spacing, it was chosen to consider the number of voxels of the lesion instead of the physical space because the U-Net ensemble only consider single slices as 2D images.

The next step was the training of a classificator to decide if a segmented SCI has to be considered true based on features that will be discussed further on in this section. Different classifiers was attempted to reach the best possible results, as it will showed in the next Chapter. Both the training and the testing of the classifier was possible thanks to the provided manual segmentation of the SCIs, used as a ground truth.

2.3.1.1 Ground Truth

The first task to perform in order to obtain a good classifier was to decide how the automatically segmented lesions has to be considered true or false. That is needed since even the labels for the corrected segmented lesions often wasn't perfectly superimposable on the ground truth label. This need was satisfied using a metric which is a modified version of the Jaccard index which is capable to take in account for the cases in which a single label obtained with the U-Net ensemble overlaps two or more labels of the manual segmentation. The mathematically expression of the used index in reported in equation 2.1.

$$score(X) = \frac{|X \cap (Y_1 \cup Y_2 \cup ... \cup Y_n)|}{|X| + \sum_{i=0}^{n} |Y_i|} \tag{2.1}$$

Where X represents the set of pixel of the automatically segmented label, Y_i represents one of the ground truth label that overlaps X and n is the number of the overlapped ground truth labels. To consider the *a priori* truth value of a single label obtained with the U-Net were manually compared the automatic segmentation of the SCIs with the ground truth and were chosen to keep a value of 0.25 for the modified Jaccard index as a minimum threshold to consider the label true.

2.3.1.2 Feature Extraction

The next step was to choose which features to extract for the automatically segmented labels. The features extracted where the followings:

- The mean probability value of a label given by the U-Net;
- The total volume of the label measured in mm^3;

- The overlapping of a segmented label with a determined mask in which the probability of finding SCI is higher. The overlapping was measured computing the fraction of the pixels of the label that overlaps the selected mask over the total number of pixels of the label. It was again chosen to use the number of pixel because the U-Net ensemble only consider one slice per time;
- The overlapping of a segmented label with a determined mask in which the probability of finding SCI is lower. The overlaping was measured in the same way as for the previous point;
- The probability of a label to be in the white matter considering the partial volume map of the atlas. This probability was evaluated averaging the probability of each pixel of the label over the total number of pixels of the lesion;
- The probability of a label to be in the grey matter;
- The probability of a label to be in the cerebrospinal fluid;

The first feature comes from the output of the U-Net. The segmentation neural network returns a probability for each pixel to be part of a lesion. The mean value for the pixels of each label was taken.

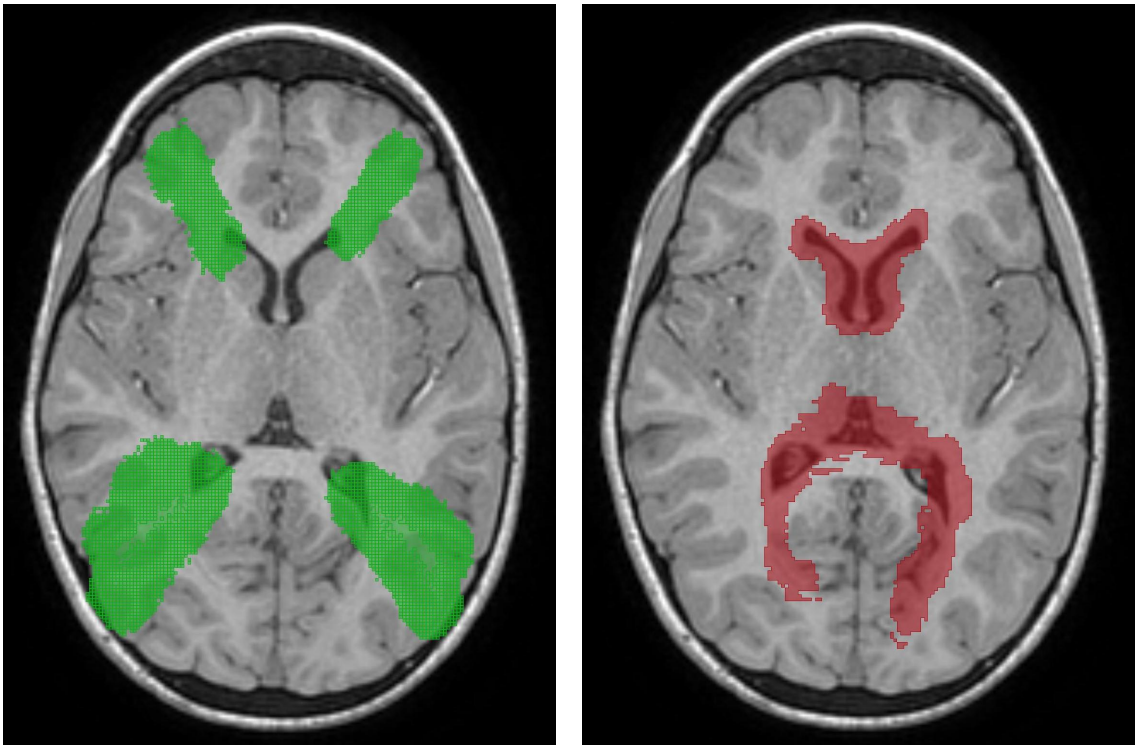
The volume of a lesion was taken because, since the nature and the given definition of a SCI, the probability of finding very large or very small lesion is low.

Two binary masks were provided by expert clinicians from the University of Padua. The first one is a mask of the boundary region of brain arteries and the cortical zone that in this work thesis will be called *boundary region*. This is the region where the SCIs are more localized since the global reduction in arterial oxygen content and the cerebral hemodynamic factors seems to drive the ischemic mechanism [51]. An example of the applied mask can be seen in Figure 2.8a The second is a mask of the ventricular and periventricular regions in the brain. Those regions have a great probability to lead to false positive due to their structure which often lead to high intensity zone in a FLAIR scan for normal anatomical factors. This mask will be called *exclusion region* and an example can be seen in Figure 2.8b. Those mask were manually segmented on the MNI 152 atlas and then registered on each image analysed. The overlapping between a label and a certain mask was calculated averaging the number of pixels in the mask over the total number of pixels in the label, as showed in the equation 2.2.

$$P(X \in Y) = \frac{|X \cap Y|}{|X|} \tag{2.2}$$

Where X is the set of pixels of a label and Y is the set of pixels in the mask.

The probability of a label of being localized in a determined tissue therefore was obtained using the maps of the partial volume effect obtained in the pre-processing stage, having in this way a feature calculation which is based on the proper anatomy of each brain. This features were extracted in order to keeping in account both the medical evidence of the SCI to be more localized into the deep white matter [6], and the working proper of the U-Net ensemble, which consider only one slice per time and therefore could suffer the partial volume effect. For each tissue the probability of a label to belong to that tissue was calculated averaging the probability of each pixel over the total number of pixel in a label as the equation 2.3 shows.



(a) *Boundary Zone* **(b)** *Exclusion Zone*

Figure 2.8: *In Figure are reported an example of the masks used to evaluate the localization of a label registered over a patien volume. In Figure a is possible to see the region where the SCIs are usually more localized due to the anatomy of the human brain [6], and in Figure b a zone which tends to naturally give hyperintense zones in the FLAIR image due to the walls of the ventricles. Both the masks were manually segmented by two expert clinicians of the University of Padua on the MNI152 atlas [41] and then registered on each valued image.*

$$P(X \in A) = \frac{\sum_i P(x_i \in A)}{|X|} \tag{2.3}$$

Where A is the considered tissue (WM, GM or CSF), X the set of pixels of a label and x_i is a pixel of the label. The probability $P(x_i \in A)$ is given by the probability maps returned by the pre-processing segmentation.

2.3.1.3 Training

A set of two classifiers were then trained taking in input both a 2 dimensional array of features ($Nsamples \times Nfeatures = Nsamples \times 7$) and a single dimensional array reporting the ground truth classification of each label, obtained starting from the manual segmentation and calculated as explained above. The classifiers chosen were a *Decision Tree* a *Random Forest* and a *Logistic Regression* classifiers. The Decision Tree was chosen because of its interpretability and its capability to mimic the human decision process. The Random Forest was attempt because of its higher resilience to overtraining and its usually highly performances with respect to decision tree. The Logistic Regression was implemented due to the possibility of return a probability for each classification.

The complete data set was divided in two independent subsets, the first one, composed of 51 patient's scans, was used during the training stage, the second one, composed of 6 patients' scan was used for the test stage. In order to keep in account of all the possible variables the full data set was divided in order to obtain both the training and the test data set heterogeneous in terms of center or provenience, physical size of the automatically segmented labels and possible outcomes of the ground truth (i.e. possibility for a patient to be healthy and really not having SCI's evidences).

The training set was composed of the images of 51 patients and totally counted 5581 automatically obtained labels. This set was found to be highly unbalanced since approximately only the 3.7% was labels classified as true. To obtain a better training two resampling algorithms where used that permitted to obtain a better balancing for the two classes.

Firstly a *SMOTE* [52] algorithm where used to add data to the less populated class. The SMOTE is a resampler algorithm that permits to artificially generate new data starting from the existing ones. The more populated class was then undersampled with a random resampler algorithm, which randomly take out data from the selected class.

The parameters of the implemented classifiers were found using a *Grid Search Cross Validation*, a process that permits to find the best parameters combination automatically maximizing a predetermined metric in a *cross validation* test. The cross validation process consists in the division of the given set (in this case the training set in to n subsets, $n-1$ of those subsets are used for training the classifier and the remaining one is used for test, measuring a predetermined metric, then the operation is repeated changing every time the test subset and finally averaging the obtained score. The grid search operation consists in repeating the cross validation test for every possible parameters combination, permitting to find the one that obtain the maximum value for the chosen metric. In this case also the parameters for the resamplers was found using the Grid Search, but generally for the SMOTE the output ratio between the minority and majority class ranged between 0.2 and 0.3 and for the random undersampler the output ratio found was between 0.6 and 0.7 for each classifier.

2.3.2 Implementation

The post processing pipeline was implemented using Python. To perform the essential operations the same libraries used on the pre-processing stage where used: ITK [45] and NUMPY [46]. To perform the classification the automatically segmented label two libraries were used: SCIKIT-LEARN [48] and IMBALANCED-LEARN [53]. The whole code is opensource and freely available on GitHub [49]. During the develpement, the installation was automatically tested on Ubuntu for different python versions. This continuos integration process was carried out using the github actions.

2.3.2.1 Feature Extraction

The first need was to distinguish and enumerate the labels for both the ground truth and the output labels uniquely. This was obtained implementing a function that finds the connected regions on an image, which relies on an ITK filter already existing. The name of the function is self-explanatory.

Then the need was to quantify the agreement of automatically obtained label of the training set with the provided ground truth. This was obtained implementing a function that measure the modified Jaccard index as shown in equation2.1 and then selecting only those lables that reaches a score higher than 0.25.

Listing 2.7: Modified Jaccard Index

```
import itk
import numoy as np

def find_Jaccard_truth_value (counted_label, gnd_label, threshold = 0.25):

    #FINDING NUMBER OF LABELS
    maximum_filter = itk.MinimumMaximumImageCalculator[type(counted_label)].New()
    maximum_filter.SetImage(counted_label)
    maximum_filter.ComputeMaximum()
    num_labels = maximum_filter.GetMaximum()

    count_labels = np.array([0] * num_labels) #total number of pixels for lesion
    value_array = np.array( [False] * num_labels ) #array that reports what label is true

    #FINDING NUMBER OF GND TRUTH LABELS
    counted_gnd = find_connected_regions(gnd_label)
    maximum_filter = itk.MinimumMaximumImageCalculator[type(counted_gnd)].New()
    maximum_filter.SetImage(counted_gnd)
    maximum_filter.ComputeMaximum()
    num_gnd = maximum_filter.GetMaximum()

    count_gnd = np.array( [0] * num_gnd ) #total number of pixels for lesion

    overlapping_matrix = np.array( [[0]*num_gnd]*num_labels)
    jaccard_index = np.array([0.]*num_labels)
    index = itk.Index[3]()

    for index[0] in range( counted_label.GetLargestPossibleRegion().GetSize()[0] ):
            for index[1] in range( counted_label.GetLargestPossibleRegion().GetSize()[1] ):
                for index[2] in range( counted_label.GetLargestPossibleRegion().GetSize()[2] ):
                    if counted_gnd.GetPixel(index) != 0 :
                        count_gnd[ counted_gnd.GetPixel(index)-1] +=1
                    if counted_label.GetPixel(index) != 0 :
                        count_labels[ counted_label.GetPixel(index) -1] +=1
                        if counted_gnd.GetPixel(index) != 0 :
                            overlapping_matrix[counted_label.GetPixel(index) - 1][counted_gnd.GetPixel(index)-1] += 1

    num = np.array([0]*num_labels)
    den = np.array([0]*num_labels)
```

```
40
41      for i in range(num_labels):
42          for j in range(num_gnd):
43              num[i] += overlapping_matrix[i][j]
44              if overlapping_matrix[i][j]!= 0 : den[i] += count_gnd[j
    ]
45          den[i] += count_labels[i] - num[i]
46
47          jaccard_index[i] = num[i]/den[i]
48
49          value_array[i] = (jaccard_index[i] >= threshold)
50
51      return value_array, jaccard_index
```

Once the labels of the training set have been classified as true or false the subsequent step was to properly extract the cited features.

The features extracted was the mean of the probability values given by the U-Net for the pixels in a label; the volume of the label, obtained multiplying the number of the voxels in a label for the voxel dimension; the mean of overlapping with two binary masks (boundary region and exclusion region) and the mean value for the overlapping with three probability masks (WM, GM, CSF). Since the boundary and exclusion region are binary the overlapping has been taken in the same way the mean probability of belonging to a determined tissue was calculated. This permitted to implement a function that take as input only two parameters: the image with the labels and a list of masks.

The feature extracted therefore had different magnitude and also there was the possibility that some features was an outlier. Outliers are defined as an observation which deviates a lot from other observations to arouse suspicion that was generated by a different mechanism in a data set [54]. To have a better distribution of the data the obtained features was then applied a scaler, i.e. an algorithm capable to normalize the data in order to have all the same order of magnitude. The applied algorithm was the function *Robust Scaler* from the library Scikit-Learn [48], which computes the median and the interquartile range (IQR) for each feature of the data set proposed and then remove the medians for each feature and scales them according to the IQR [55]. Usually features scaling the mean and the variance are used, but those metrics tend to be badly affected by the outliers, this approach is more robust toward outliers and permits better results.

Listing 2.8: Feature Extracion

```
1  import itk
2  import numpy as np
3  from sklearn.preprocessing import RobustScaler
4
5  def feature_scoring (label_img, masks_list):
6      #binarize the labels
7      bin_label = binarize (label_img, 0.5)
8
9      #differentiating the labels
10     counted_label = find_connected_regions (bin_label)
11     #Calculating the volume of a voxel
12     voxel_volume = label_img.GetSpacing()[0] * label_img.GetSpacing
    ()[1] * label_img.GetSpacing()[2]
13
14     #FINDING NUMBER OF LABELS
```

```
    maximum_filter = itk.MinimumMaximumImageCalculator[type(counted_label)].New()
    maximum_filter.SetImage(counted_label)
    maximum_filter.ComputeMaximum()

    index = itk.Index[3]()
    pounded_score = np.array( [[0.] * (len(masks_list) + 2)] * maximum_filter.GetMaximum() ) #score for lesions divided pixels per lesion
    score = np.array( [[0.] * (len(masks_list) + 2)] * maximum_filter.GetMaximum() ) #total scores for each lesion
    count = np.array( [0.] * maximum_filter.GetMaximum() ) #total number of pixels for lesion

    for index[0] in range( label_img.GetLargestPossibleRegion().GetSize()[0] ):
        for index[1] in range( label_img.GetLargestPossibleRegion().GetSize()[1] ):
            for index[2] in range( label_img.GetLargestPossibleRegion().GetSize()[2] ):
                if label_img.GetPixel(index) != 0 :
                    count[counted_label.GetPixel(index) - 1] += 1
                    score[counted_label.GetPixel(index) - 1][0] += label_img.GetPixel(index)
                    for i in range(len(masks_list)):
                        score[counted_label.GetPixel(index) - 1][i + 2] += masks_list[i].GetPixel(index)

    for i in range(len(score)):
        pounded_score[i][0] = score[i][0] / count[i]
        pounded_score[i][1] = voxel_volume * count[i]

        #everything in the mask must be averaged
        for j in range( 2, len(score[i]) ):
            pounded_score[i][j] = score[i][j] / count[i]

    return pounded_score, counted_label

def robust_scaling(data, transformer = None):

    if transformer == None:
        transformer = RobustScaler().fit(data)

        new_data = transformer.transform(data)
    return new_data, transformer

```

2.3.2.2 Training

Once the features were extracted for the labels of the training set, then it was possible to define and train the classifiers. For both classifiers the parameter *class_weight* was set as *'balanced'* since, in this way, the fitting algorithm would have taken in account the unbalancing of the training set. All the other parameters were automatically chosen using a *grid search* algorithm. The metric that where chose to maximize was the AUC precision-recall score, which keeps in account both the precision and recall and is usually a valid choice to estimate goodness of unbalanced

data classification.

Listing 2.9: Training

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.ensemble import BalancedRandomForestClassifier
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler

dtc = Pipeline([
    ('SMOTE', SMOTE(sampling_strategy = 0.2, random_state = 42)),
    ('UnderSampler', RandomUnderSampler(sampling_strategy = 0.7,
    random_state = 42)),
    ('classification', DecisionTreeClassifier(random_state = 42,
    max_depth = 6, class_weight = 'balanced', criterion = 'entropy')
    ) ])
dtc.fit(train_score, train_truth)

lr = Pipeline([
    ('SMOTE', SMOTE(sampling_strategy = 0.3, random_state = 42)),
    ('UnderSampler', RandomUnderSampler(sampling_strategy = 0.7,
    random_state = 42)),
    ('classification', LogisticRegression(random_state = 42,
    class_weight = 'balanced', penalty = None, solver = 'lbfgs',
    max_iter = 10000)) ])
lr.fit(train_score, train_truth)

brfc = Pipeline([
    ('SMOTE', SMOTE(sampling_strategy = 0.2, random_state = 42)),
    ('UnderSampler', RandomUnderSampler(sampling_strategy = 0.6,
    random_state = 42)),
    ('classification', BalancedRandomForestClassifier(
        random_state = 42,
        class_weight = 'balanced',
        criterion = 'entropy',
        max_depth = 8,
        max_features = None,
        n_estimators = 1000,
        n_jobs = 10)) ])
brfc.fit(train_score, train_truth)
```

Chapter 3

Results

The proposed pipeline was developed and applied on the data set described in Chapter 2.1. The main method for the evaluation of the pre-processing pipeline, i.e. the skull stripping and the tissue segmentation, was a quantitative comparison with the results obtained using the software *FSL* [56], which is a library of functions developed specifically for the analysis of the MRI brain images that became a standard in literature for MR analysis during the last couple of decades [56]. The post-processing pipeline instead was evaluated comparing quantitatively the labels with the manual segmentation of the SCIs provided.

3.1 Pre-Processing

The FMRIB Software Library (FSL) became a standard for the analysis of MRI images and then it was decided to use the brain mask obtained with the *BET* function and the segmentation masks (one for each tissue class) obtained using the *FAST* function to use as references for the pre-processing tools developed during this work of thesis. Another aspect of the proposed pipeline during its development was the execution time that had to be reasonably low in order to give the possibility to use the implemented algorithms also on personal computers.

3.1.1 Timing

One of the main researched feature for the implemented pipeline was the short execution time required to obtain a full segmentation. It was therefore pre-processed the full data set of 57 patients, calculating the computation time for each volume. Nevertheless the computation time depends on the input size and, as it was shown in the previous chapter, the data set is highly heterogeneous. Since the main difference between images is the size and the spacing on the z-axis, the time results are plotted as a function of the number of slices (i.e. images in planes perpendicular to the z-axis) as it is possible to see in Figure 3.1. To obtain the provided data the whole pre-proessing pipeline was performed 8 times on the entire data set. Of the analysed images 44 have 30 slices or less and for those images the mean execution times was smaller than 1 minute, 10 images have a number of slices between 224 and 288 and for those images the execution time ranged from 178 s to 216 s. For the remaining three images with 512 slices the computation time was about 6 min.

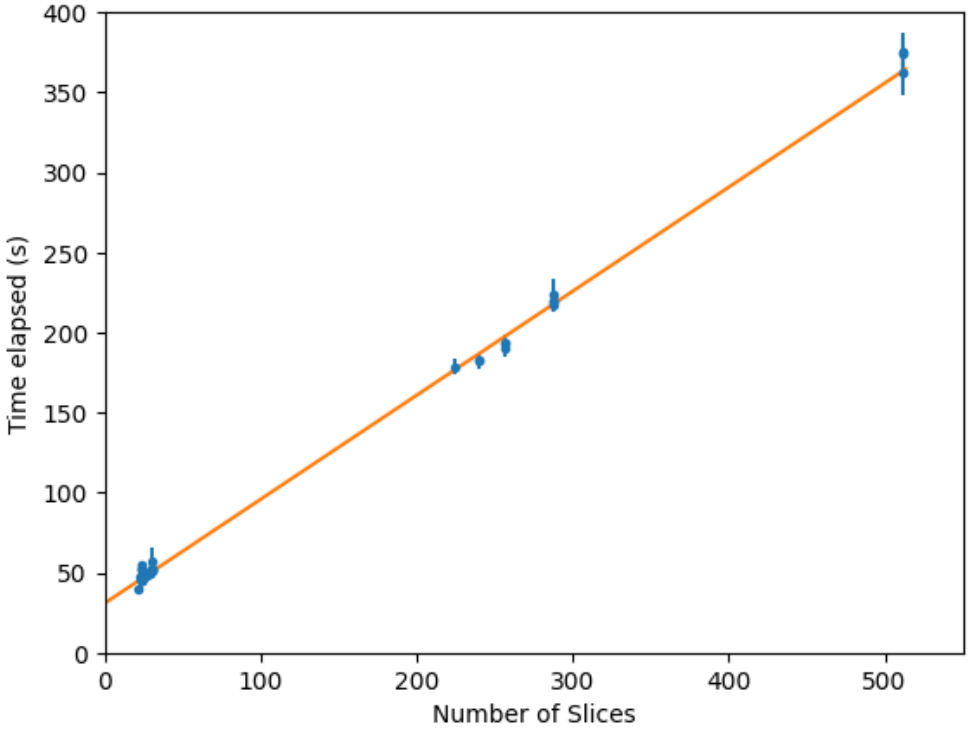


Figure 3.1: *In the Figure is plotted the execution time in function of the number of slices that composed the image. It is possible to see how the execution time increases linearly with the number of slices. In general, for images with 30 slices or fewer the execution time was less than a minute. The maximum values were reached for the images that have 512 slices and is of about* 375 *s.*

The results plotted are obtained running the pipeline on one of the server of the Department of Physics and Astronomy of the University of Bologna, a computer with 32 CPUs and 126Gb of RAM.

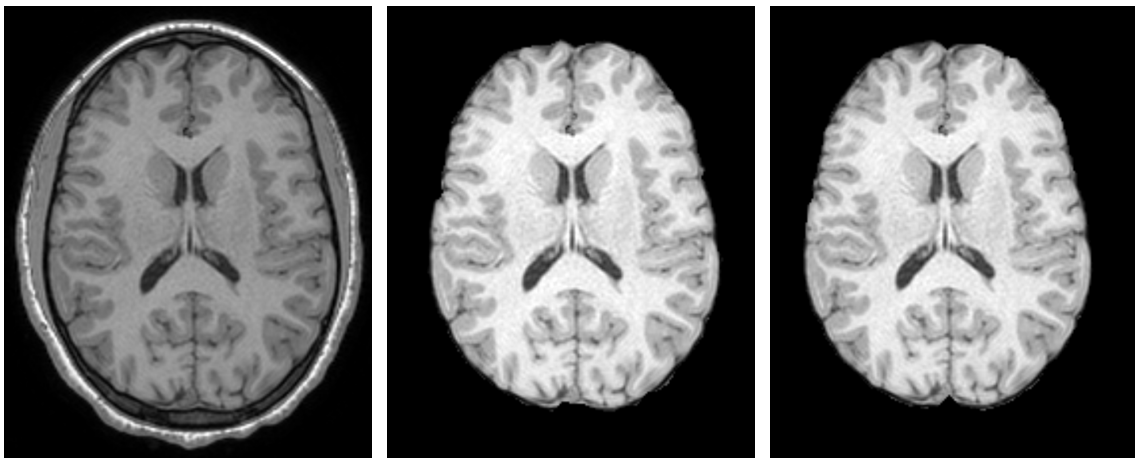
3.1.2 Segmentation Performances

This sections describes the comparison between the presented pre-processing pipeline and the segmentation obtained using FSL. The comparison was carried out using the dice coefficient presented in equation 1.8. As described before, this metric provides an overlapping measure; it's range is in [0, 1] where 0 means no overlapping and 1 a perfect overlapping of two binary volumes. To provide that measure was chosen to compare the obtained masks (brain mask, white matter, grey matter and cerebrospinal fluid) with the ones obtained with FSL. The comparison was done on a subset of 56 over the total data set of 57 patients. This was because one of the volumes has a incredibly poor resolution the FSL FAST software totally fails in its segmentation.

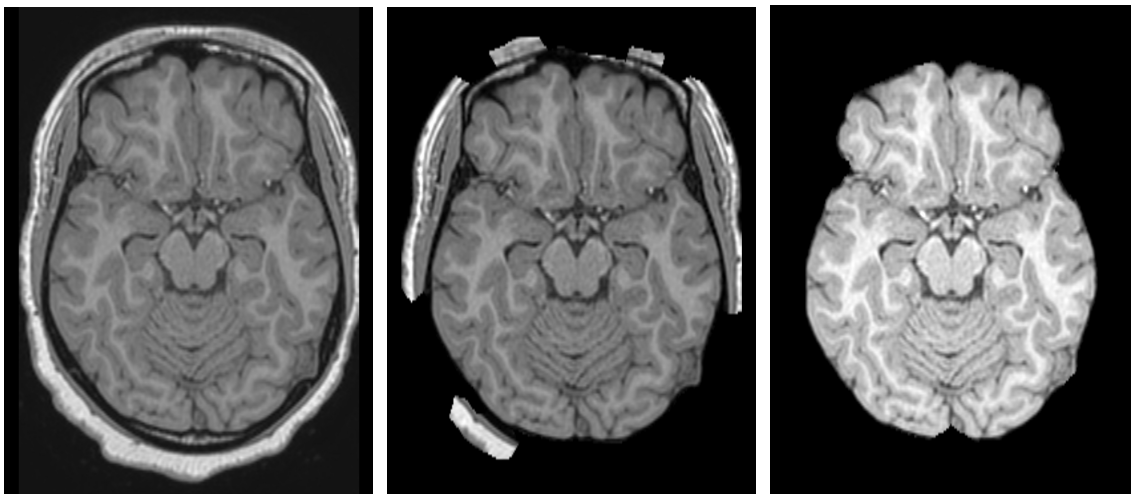
Brain Extraction: The first process evaluated was the brain extraction. It was measured calculating the Dice coefficient of the overlapping between the brain masks obtained with the proposed pipeline and the ones obtained with the FSL BET function. The obtained DSC ranges from a minimum value of 0.47 to a maximum

of 0.96. As visible in Table 3.1, the mean DSC was 0.87 ± 0.12 where the error was evaluated as the standard deviation of the analysed volumes. The median value for the DSC was 0.93.

To have a better comprehension of the obtained Dice coefficient values is however necessary to do also a qualitative evaluation of the goodness of the results of FSL. In Figure 3.2 are reported the results of brains extracted with both FSL and the proposed pipeline for a volume which the two software are in agreement leading to a high value for the Dice coefficient, and for a volume in which the FSL BET segmentation partially fails while the proposed pipeline manage in a good segmentation, leading to a low value for the measured metric. It is therefore necessary to underline that the FSL function result can be optimized by manually change the extraction parameters while the goal researched for the proposed pipeline was to work fully automatically.



(a) *Original scan* **(b)** *Brain extracted with FSL* **(c)** *Implemented pipeline*



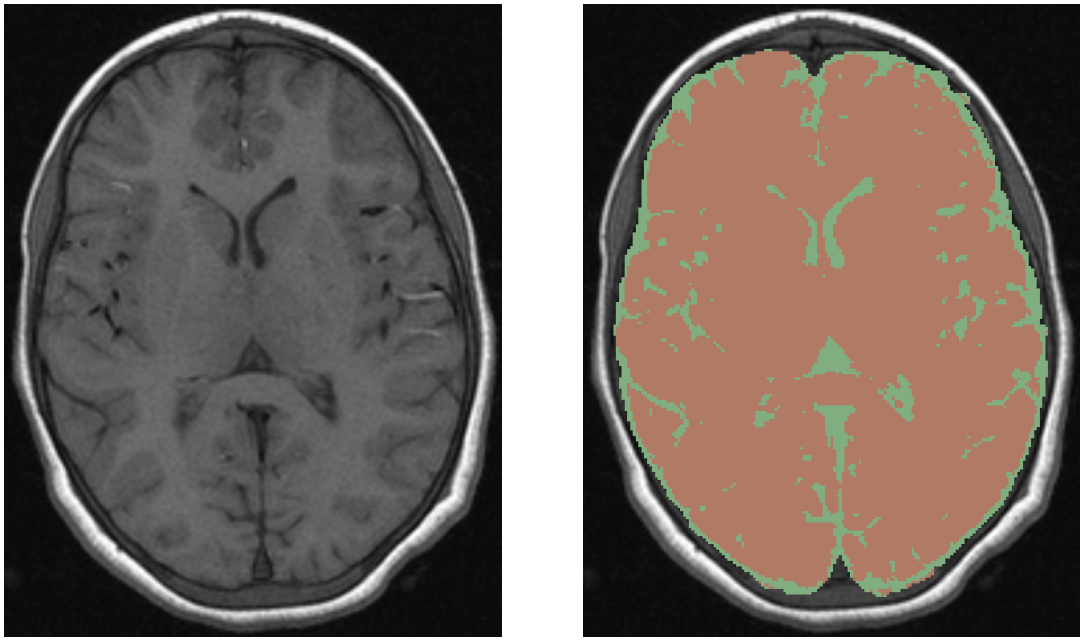
(d) *Original scan* **(e)** *Brain extracted with FSL* **(f)** *Implemented pipeline*

Figure 3.2: *In the figure is reported the visual comparison of the brain extracted with FSL and the brain extracted with the proposed pipeline in two different cases. The first row (Figures a, b, c) show a brain extraction that comparing the obtained brain mask with the one of FSL obtained a Dice coefficient of 0.95. The second row (images d, e, f) refers to a brain extraction that receive a Dice coefficient of* 0.59.

The Dice coefficient is a metric to measure the overlapping between two images,

then in the cases in which the FSL brain extraction is not the most desired one, as it is possible to see in the Figure 3.2e, the measured metric could be low even when the implemented pipeline gives a correct result.

Tissue Segmentation: The tissue segmentation was evaluated mainly comparing the binary masks obtained by the proposed pipeline to the ones obtained by the FSL FAST algorithm, after a first evaluation by eyes, checking the eventually presence of visible errors. In this way was found that on a minority of the volumes the algorithm used to merge two classes together, usually the white matter and the grey matter. This kind of issue used to happen in images with a lower contrast as it is to see in Figure 3.3. Looking at the probability maps obtained emerged that the lost class was correctly found but with a probability lower than the majority class (usually the white matter).



(a) *Original Scan* **(b)** *Segmentation results*

Figure 3.3: *In Figure is reported a T1W scan and the result of a segmentation with the proposed algorithm. In this case is possible to observe how the low contrast of the original image lead to an erroneous segmentation in which the grey matter and the white matter are both classified in the same class (red) and only the cerebrospianl fluid is correctly classified (green). It is therefore important to say that the grey matter was correctly found in the partial volume maps but the associated probability was lower than the one in the white matter map.*

Once the volumes were manually evaluated, a comparison between the masks obtained with FSL and the ones obtained with the proposed pipeline were computed measuring the obtained Dice coefficient. For the images in which two classes were merged for the missing class was kept a Dice coefficient of 0. The obtained scores are reported in Table 3.1.

It is important to mention that the FSL algorithm failed in some cases as is possible to see in Figure 3.4. This usually happened in those volumes where also the brain extraction failed. Thus the Dice coefficient isn't a direct measure of the reliability

of the proposed pipeline but an overlapping metric between the obtained masks and the masks computed by FSL, in the images in which FSL fails the obtained DSC is low even if the proposed pipeline manage to obtain a good segmentation.

	Mean	**Std. Dev.**	**Median**	**IQR**
Brain Mask	0.87	0.12	0.93	0.11
WM	0.78	0.19	0.83	0.19
GM	0.67	0.30	0.80	0.28
CSF	0.66	0.24	0.79	0.22

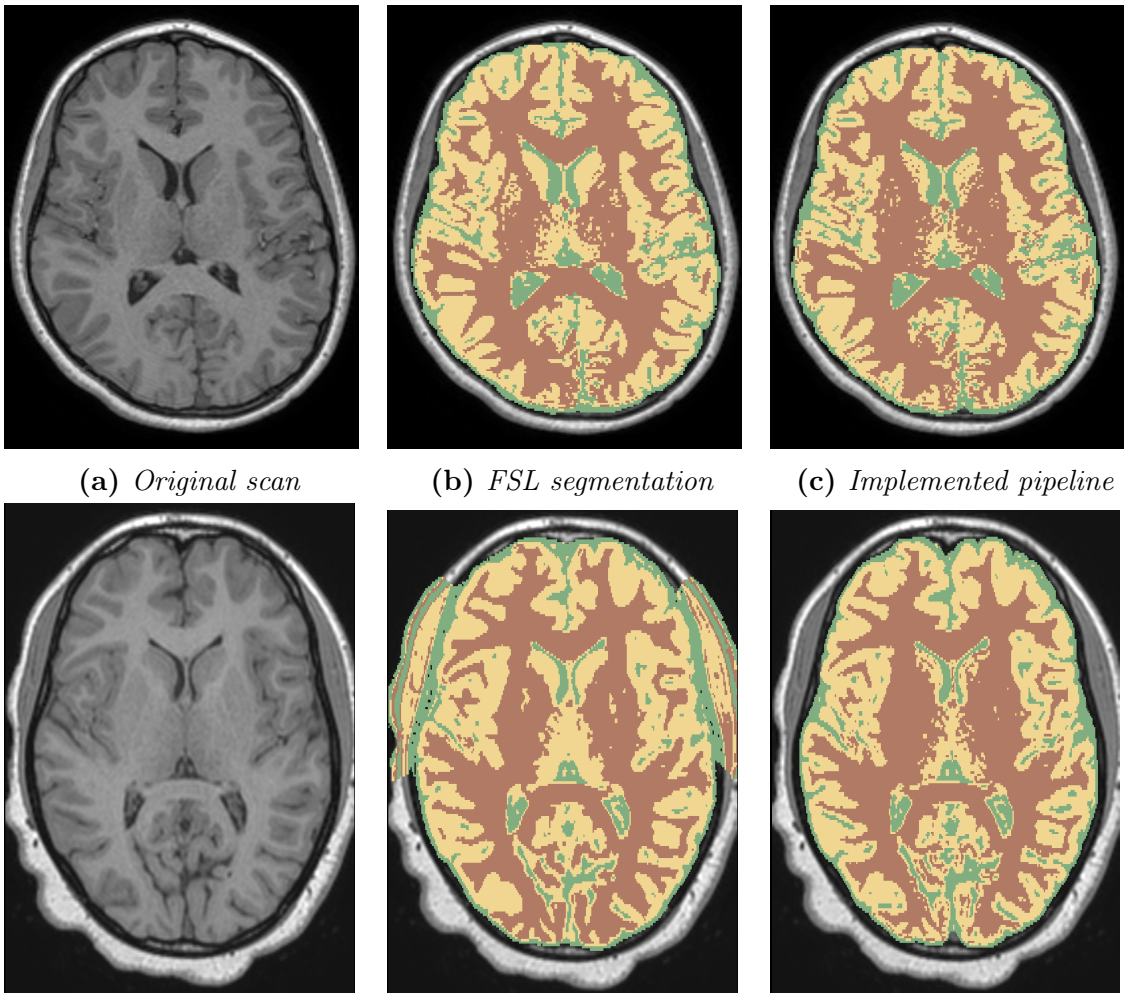
Table 3.1: *In the table are reported the mean values, the standard deviation, the median value and the interquartile range of the Dice coefficient for the comparison between the binary masks for the brain, the white matter (WM), the grey matter (GM) and the cerebrospinal fluid (CSF) obtained with the proposed algorithm and the ones obtained with FSL BET (for the brain mask) and FAST (for the tissue segmentation). As it is possible to see for the tissue segmentation the median is significantly higher than the mean. This happened because the outliers of both the proposed pipeline segmentation (where two classes were merged) and the outliers of the FSL segmentation (in which the brain extraction failed, leading to a failing segmentation) were kept in account.*

On the images taken with an higher resolution it is possible to appreciate how the proposed pipeline correctly extract the brain and segment the image in its entire volume, as visible in Figure 3.5.

3.2 Post-Processing

The provided data set was divided in two subsets. The first one is comprehensive of 51 images, in which the U-Net ensemble found a total of 5581 lesions, and constitute the *training set*, used to train the chosen classifiers. The remaining 6 images containing 425 total labels of which 99 are considered true, constitute an independent *test set*, used to evaluate the classification performances.

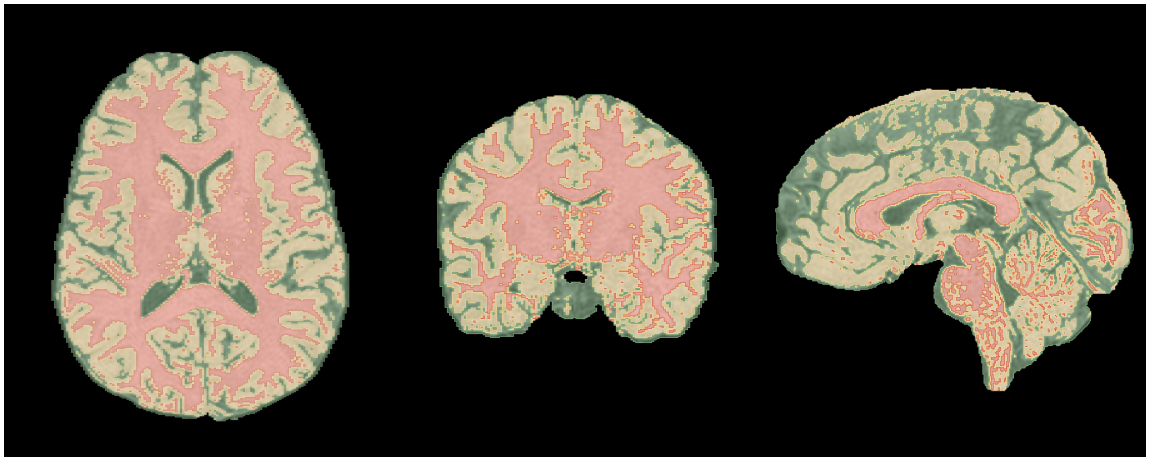
The goodness of the classifiers was evaluated comparing the labels classified by the three proposed approach with the ones classified "a priori", measuring the modified Jaccard score for overlapping with the manual segmentation, as shown in equation 2.1. To have a quantitative comparison were used different metrics in order to take in account of all the possible difference between the three implemented classifiers. The results, for each classifiers are reported in Table 3.2. The first researched characteristic for the three classifiers was to have an high recall value, in order to be more conservative toward the lesions correctly found by the U-Net ensemble. This characteristic was satisfied by all the implemented classifier. The classifier which gained the highest values for all the metrics was the random forest classifier, which however is the less explainable one. For this reason also the other two classifiers were kept during the work of thesis. It is interesting to compare the decision tree and the logistic regression because, even if the decision tree classifier have greater balance accuracy and recall, the logistic regression classifier gains an higher AUC ROC score and especially an higher AUC precision-recall score, a metric that is particularly suited for unbalanced data [40]. It is important to compare



(a) *Original scan* **(b)** *FSL segmentation* **(c)** *Implemented pipeline*

(d) *Original scan* **(e)** *FSL segmentation* **(f)** *Implemented pipeline*

Figure 3.4: *In the figure is reported the visual comparison of the brain segmented with FSL and the brain segmented with the proposed pipeline. The color scheme here used is red for white matter, yellow for grey matter and green for cerebrospinal fluid. The head scan for the second row (Figures d, e, f) is the same of Figure 3.2 and in it is possible to see how a low quality segmentation lead not fully correct segmentation, concurring to obtain a lower Dice coefficient.*



(a) *Axial view.* **(b)** *Frontal view.* **(c)** *Lateral view.*

Figure 3.5: *3D visualization of the segmentation of an extracted brain. Looking at the three different points of view of the captured volume (axial view, frontal view and lateral view) is possible to appreciate how the full brain volume was extracted from the unwanted head structures and segmented in white matter (red), grey matter (yellow) and cerebrospinal fluid (green).*

the result obtained for the area under precision-recall curve with the ones that a random classifier would have gained with the proposed set of data: since the data are unbalanced with a positives ratio of about 0.23 the expected score for a random classifier is of 0.23.

	Bal. Acc.	**AUC ROC**	**AUC PR**	**Precision**	**Recall**
Decision Tree	0.74	0.81	0.45	0.38	0.96
Logistic regression	0.68	0.83	0.61	0.32	0.96
Random Forest	**0.74**	**0.90**	**0.74**	**0.38**	**0.97**

Table 3.2: *In the table are reported the results of the five evaluated metrics (balanced accuracy, area under ROC curve, area under precision-recall curve, precision and recall) comparing the classified labels with the ones 'a priori' classified for all the three implemented classifiers (decision tree, logistic regression and random forest). As it is possible to appreciate all the classifiers obtain a very high value for the recall, which means that very few classified data are found to be* false negative*. The Random Forest classifier obtains the higher value in all the remaining metrics and therefore is the one that also classify correctly the greatest number of negative lesions.*

In the end was also interesting to look at the importance of the features for the decision tree and the random forest classifiers and to look at the coefficient given at each feature by the logistic regression. This gave us important informations about the functionality of the U-Net ensemble. The results obtained are reported in Table 3.3. It was discovered that both the random forest classifier and the decision tree didn't took in account at all for the probability given as output from the U-Net ensemble and the logistic regression gave it a strong negative value. As a check this feature was removed from all the classifier and the results obtained for the measures of the different metrics evaluated didn't change significantly. Another interesting discovery was the high relevance given to the white matter probability map, meaning

that the U-Net ensemble have the tendency to find lesions also outside the white matter, probably due to the working characteristic of the U-Net to consider one slice per time, and then suffer the possible partial volume effect.

	Prob.	**Size**	**Boundary**	**Exclusion**	**WM**	**GM**	**CSF**
Dec. Tree	0.000	0.0133	0.032	0.034	0.659	0.042	0.098
Log. reg.	0.00	8.83×10^{-2}	0.236	-1.50	1.07×10^{5}	4.58×10^{4}	7.82×10^{3}
Rnd F.	0.000	0.146	0.027	0.029	0.615	0.067	0.116

Table 3.3: *Table reporting the importance of the features for the decision tree and random forest classifiers. The features indicated are the probability in output from the U-Net, the physical size of the lesion, the overlapping with a manually defined boundary and exclusion regions and the probability to be in the white matter, grey matter or cerebrospinal fluid. For the logistic regression classifier the coefficients associated at the features are reported. It can be seen how the classifiers take in account the different features and for example evaluating which feature can be removed.*

Conclusions

In this work of thesis I have developed, implemented and tested an automatic pipeline for the brain extraction and tissue segmentation for MRI head scans, and a pipeline for the refinement of the results of an automated process of labelling Silent Cerebral Infarcts in patients with Sickle Cell Disease. The data set used in this work of thesis consists of a set of 57 FLAIR and T1W MRI head scans provided by different medical centers in Italy. The images of the data set were already manually labelled by expert clinicians at the University of Padua.

As a first step I have performed a brain extraction. This step involves the usage of a brain mask of an already segmented head atlas which permitted first rough extraction for the brain, followed by a sequence of application of different morphological operations in combination with thresholing operations. This step is then followed by a brain tissues segmentation implemented by the usage of an expectation maximization algorithm applied to a Gaussian mixture model whose parameters were estimated using the maps of the partial volume effect provided by the ICBM MNI 152 atlas. The first step have permitted to obtain standardized images that the already implemented SCIs' segmentation pipeline, a U-Net ensemble, could more easily elaborate. The second step has led to a soft segmentation of the brain, resulting in three partial volume maps for each image and this has been useful for the features extraction of the post-processing stage.

The last step has been the classification of the labels obtained by the U-Net ensemble in order to refine the results by removing the false positive outcomes of the U-Net ensemble itself. To perform this step, different features have been extracted from the labels considering both the proper functionality of the labelling ensemble and the known anatomical characteristics of the SCIs. A set of three classifiers has then been trained on a subset of the whole provided data set using the features extracted measuring the overlapping of the labels with the probability maps in output from the segmentation step, the physical size of the labels and the overlapping with masks manually provided by expert clinicians that delimits brain regions in which usually are localized the SCIs.

The pre-processing pipeline has been tested comparing the results obtained for both the brain extraction and the tissue segmentation to the segmentation obtained using the functions of the software FSL. For the whole data set has been measured the Dice similarity coefficient (DSC) of the overlapping between the obtained masks.

The mean dice similarity coefficient of the proposed brain extraction technique is 0.87 ± 0.12 and on some images performed better than the software using for comparison. The tissue segmentation pipeline has shown some issues on the images with poor contrast but have obtained however a mean DSC value of 0.78 ± 0.19 for the white matter segmentation, 0.67 ± 0.30 for the grey matter and 0.66 ± 0.24 for the cerebrospinal fluid. Even in this case on some volumes the proposed technique

reach better results than the software used for the comparison.

The post-processing pipeline has been tested on a subset of the whole data set that has not been used during the training stage. The test set has been chosen to well represents the heterogeneity of the full data set. The test has been performed measuring different metrics to correctly evaluate the performance. All the classifiers have shown a recall next to the maximum value (0.97), meaning the almost absence of false negative outcomes and a good balanced accuracy, always higher than 0.67. The classifier which reaches the best performances has been a random forest classifier that scores an AUC precision recall near 0.74 and balanced accuracy of 0.74.

The pipeline has been implemented using Python and is part of an open-source project freely available on the platform GitHub.

Further developing of this project is possible, like increasing the performances of the tissue segmentation step considering different approaches in the way the initial parameters are estimated, or to develop the pipeline of the post-processing stage to permit a refinement on the boundary and the size of the labels found by the U-Net ensemble.

In the end this project provided a suitable approach with satisfactory results for the extraction and the segmentation of the brain from head T1W MRI scans and an approach capable to improve the segmentation of the SCIs obtained with a U-Net ensemble.

Appendix A

Proof of equality between Dice Coefficient and F1 Score

Using the four definitions given for the confusion matrix in Chapter 1 it is possible to proof the equalities of those two equations.

Recalling 1.2 and 1.3:

$$\begin{aligned} F1 &= \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2}{\frac{tp+fp}{tp} + \frac{tp+fn}{tp}} = \\ &= \frac{2}{\frac{2tp+fp+fn}{tp}} = \frac{2tp}{2tp + fp + fn} \end{aligned} \tag{A.1}$$

Now the two classes to be considered in the Dice Similarity Coefficient 1.8 are the *Predicted Positive* as X and the *True Positive* as Y, so we find that $X = tp + fp$ and $Y = tp + fn$

Applying those definitions we find that:

$$\begin{aligned} |X \cap Y| &= tp \\ |X| + |Y| &= tp + fp + tp + fn = 2tp + fp + fn \end{aligned} \tag{A.2}$$

The consequence is that:

$$DSC = 2\frac{|X \cap Y|}{|X| + |Y|} = 2\frac{tp}{2tp + fp + fn} = F1 \tag{A.3}$$

Bibliography

[1] *What is Sickle Cell Disease?* https://www.cdc.gov/ncbddd/sicklecell/facts.html. Accessed: 19/02/2023.

[2] *NHS. Sickle Cell Disease: Symptoms.* https://www.nhs.uk/conditions/sickle-cell-disease/symptoms/. Accessed: 24/02/2023.

[3] Charles H. Pegelow et al. "Silent Infarcts in Children With Sickle Cell Anemia and Abnormal Cerebral Artery Velocity". In: *Archives of Neurology* 58.12 (Dec. 2001), pp. 2017–2021. DOI: 10.1001/archneur.58.12.2017.

[4] M.R. DeBaun et al. "Silent cerebral infarcts: a review on a prevalent and progressive cause of neurologic injury in sickle cell anemia." In: *Blood* 119-120 (May 2012), pp. 4587–96. DOI: 10.1182/blood-2011-02-272682.

[5] M. P. Boaro et al. "S265: RADIOMICS AND ARTIFICIAL INTELLIGENCE FOR IDENTIFICATION AND MONITORING OF SILENT CEREBRAL INFARCTS IN SICKLE CELL DISEASE: FIRST ANALYSIS FROM THE GENOMED4ALL EUROPEAN PROJECT". In: *HemaSphere* 6 (2022), pp. 166–167. DOI: 10.1097/01.HS9.0000843952.59228.1d.

[6] M.E. Houwing et al. "Silent cerebral infarcts in patients with sickle cell disease: a systematic review and meta-analysis". In: *BMC Medicine* 18 (2020), p. 393. DOI: 10.1186/s12916-020-01864-8.

[7] *GENOMED4ALL: Genomics For Next Generation Healthcare.* http://genomed4all.eu/. Accessed: 19/02/2023.

[8] *GenoMed4All: il progetto Humanitas selezionato come European AI Excellence all'EXPO Dubai 2022.* Accessed: 24/02/2023. URL: https://www.humanitas.it/news/genomed4all-il-progetto-humanitas-selezionato-come-european-ai-excellence-all-expo-dubai-2022/.

[9] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. "Current Methods in Medical Image Segmentation". In: *Annual Review of Biomedical Engineering* 2.1 (2000). PMID: 11701515, pp. 315–337. DOI: 10.1146/annurev.bioeng.2.1.315. eprint: https://doi.org/10.1146/annurev.bioeng.2.1.315. URL: https://doi.org/10.1146/annurev.bioeng.2.1.315.

[10] Giuseppe Filitto. "Implementation of an automated pipeline to predict the response to neoadjuvant chemo-radiotherapy of patients affected by colorectal cancer". MA thesis. Alma Mater Studiorum - Università di Bologna, School of Science, 2021.

[11] Riccardo Biondi. "Implementation of an Authomated Pipeline for the Identification of Ground Glass Opacities in Chest CT Scans of Patient Affected by COVID-19". MA thesis. riccardo.biondi7@unibo.it: Alma Mater Studiorum - Università di Bologna, School of Science, Dec. 2020.

[12] Michele Larobina and Loredana Murino. "Medical Image File Formats". In: *Journal of Digital Imaging* 27 (2 2014). ISSN: 27. DOI: 10.1007/s10278-013-9657-9.

[13] *The Encyclopedia of Graphics File Formats.* http://www.fileformat.info/mirror/egff/. Accessed: 22/02/2023.

[14] F. Bloch, W. W. Hansen, and Martin Packard. "Nuclear Induction". In: *Phys. Rev.* 69 (3-4 Feb. 1946), pp. 127–127. DOI: 10.1103/PhysRev.69.127. URL: https://link.aps.org/doi/10.1103/PhysRev.69.127.

[15] M. Niknejad, Y. Baba, and F. Deng. *T1 weighted image.* Accessed: 23/02/2023. DOI: https://doi.org/10.53347/rID-21760.

[16] J. Jones, A. Haouimi, and S. Vadera. *T2 weighted image.* Accessed: 23/02/2023. DOI: https://doi.org/10.53347/rID-6345.

[17] D. Finkelstein et al. "Differential diagnosis of T2 hypointense masses in musculoskeletal MRI". In: *Skeletal Radiology* 50 (10 Jan. 2021). DOI: https://doi.org/10.1007/s00256-021-03711-0.

[18] J Jones, Y. Baba, and S. Vadera. *Fluid attenuated inversion recovery.* Accessed: 23/02/2023. DOI: https://doi.org/10.53347/rID-5852.

[19] C. Li et al. "Early detection of secondary damage in ipsilateral thalamus after acute infarction at unilateral corona radiata by diffusion tensor imaging and magnetic resonance spectroscopy." In: *BMC neurology* (May 2011). DOI: https://doi.org/10.1186/1471-2377-11-49.

[20] Luis Martì-Bonmatì. "MR Image Acquisition: From 20 to 3D". In: *3D Image Processing: Techniques and Clinical Applications* (2002), pp. 21–31. DOI: 10.1007/978-3-642-59438-0.

[21] Junchi Bin et al. "The registration of visible and thermal images through multi-objective optimization". In: *Information Fusion* 95 (2023), pp. 186–198. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2023.02.020. URL: https://github.com/jb2849/MOIR..

[22] Stefan Klein et al. "elastix: A Toolbox for Intensity-Based Medical Image Registration". In: *IEEE Transactions on Medical Imaging* 29.1 (2010), pp. 196–205. DOI: 10.1109/TMI.2009.2035616.

[23] Derek L G Hill et al. "Medical image registration". In: *Physics in Medicine Biology* 46.3 (Mar. 2001), R1. DOI: 10.1088/0031-9155/46/3/201.

[24] *3dSlicer Wiki. Registration:Resampling.* Accessed: 26/02/2023. URL: https://www.slicer.org/wiki/Registration:Resampling.

[25] Daniel Withey and Zoltan J. Koles. "A Review of Medical Image Segmentation: Methods and Available Software". In: 2008.

[26] *Sci-kit Image. Thresholding.* Accessed: 24/02/2023. URL: https://scikit-image.org/docs/stable/auto_examples/applications/plot_thresholding.html.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

[28] Chaoxin Zheng and Da-Wen Sun. "2 - Image Segmentation Techniques". In: *Computer Vision Technology for Food Quality Evaluation*. Ed. by Da-Wen Sun. Food Science and Technology. Amsterdam: Academic Press, 2008, pp. 37–56. ISBN: 978-0-12-373642-0. DOI: https://doi.org/10.1016/B978-012373642-0.50005-3.

[29] "Data on MRI brain lesion segmentation using K-means and Gaussian Mixture Model-Expectation Maximization". In: *Data in Brief* 27 (2019), p. 104628. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2019.104628.

[30] Alexandra Lauric and Sarah Frisken. "Soft segmentation of CT brain data". In: *Tufts University Working Paper* (Jan. 2007).

[31] Shui-Yuan Huang et al. "Image classification and adversarial robustness analysis based on hybrid quantum–classical convolutional neural network". In: *Optics Communications* 533 (2023), p. 129287. ISSN: 0030-4018. DOI: https://doi.org/10.1016/j.optcom.2023.129287.

[32] Dongxian Wu et al. "Skip connections matter: On the transferability of adversarial examples generated with resnets". In: *arXiv preprint arXiv:2002.05990* (2020). DOI: https://doi.org/10.48550/arXiv.2002.05990.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). DOI: https://doi.org/10.48550/arXiv.1505.04597.

[34] Günther Grabner et al. "Symmetric Atlasing and Model Based Segmentation: An Application to the Hippocampus in Older Adults". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*. Ed. by Rasmus Larsen, Mads Nielsen, and Jon Sporring. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 58–66.

[35] D. Louis Collins et al. "ANIMAL+INSECT: Improved Cortical Structure Segmentation". In: *Information Processing in Medical Imaging*. Ed. by Attila Kuba, Martin Šáamal, and Andrew Todd-Pokropek. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 210–223.

[36] Amalia Luque et al. "The impact of class imbalance in classification performance metrics based on the binary confusion matrix". In: *Pattern Recognition* 91 (2019), pp. 216–231. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.02.023.

[37] David Powers. "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness Correlation". In: *Mach. Learn. Technol.* 2 (Jan. 2008).

[38] B.W. Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), pp. 442–451. ISSN: 0005-2795. DOI: https://doi.org/10.1016/0005-2795(75)90109-9.

[39] Pierre Baldi et al. "Assessing the accuracy of prediction algorithms for classification: an overview ". In: *Bioinformatics* 16.5 (May 2000), pp. 412–424. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/16.5.412.

[40] Brice Ozenne, Fabien Subtil, and Delphine Maucort-Boulch. "The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases". In: *Journal of Clinical Epidemiology* 68.8 (2015), pp. 855–859. ISSN: 0895-4356. DOI: https://doi.org/10.1016/j.jclinepi.2015.02.010.

[41] *ICBM 152 Nonlinear atlases version 2009.* Accessed: 27/02/2023. URL: http://www.bic.mni.mcgill.ca/ServicesAtlases/ICBM152NLin2009.

[42] Xu Han et al. "Brain extraction from normal and pathological images: A joint PCA/Image-Reconstruction approach". In: *NeuroImage* 176 (2018), pp. 431–445. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2018.04.073.

[43] Stefan Klein and Marius Staring. *Elastix. The Manual.* Oct. 2019. URL: https://github.com/SuperElastix/elastix/releases/download/5.0.0/elastix-5.0.0-manual.pdf.

[44] *3D Slicer image computing platform.* Accessed: 01/03/2023. URL: https://www.slicer.org.

[45] Hans J. Johnson et al. *The ITK Software Guide Third Edition - Updated for ITK version 4.5.* 2013. URL: http://itk.org/ItkSoftwareGuide.pdf.

[46] Travis E Oliphant. *A guide to NumPy.* Vol. 1. Trelgol Publishing USA, 2006.

[47] Insight Software Consortium. *ITK-Elastix.* URL: https://github.com/InsightSoftwareConsortium/ITKElastix.

[48] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[49] Nicolas Biondini and Riccardo Biondi. *Neuroradiomics.* 2023. URL: https://github.com/bionano94/Neuroradiomics.

[50] James F. Casella et al. "DESIGN OF THE SILENT CEREBRAL INFARCT TRANSFUSION (SIT) TRIAL". In: *Pediatric Hematology and Oncology* 27.2 (2010), pp. 69–89. DOI: 10.3109/08880010903360367.

[51] Andria L. Ford et al. "Silent infarcts in sickle cell disease occur in the border zone region and are associated with low cerebral blood flow". In: *Blood* 132.16 (2018), pp. 1714–1723. ISSN: 0006-4971. DOI: https://doi.org/10.1182/blood-2018-04-841247.

[52] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[53] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.

[54] Calpephore Nkikabahizi, Wilson Cheruiyot, and Ann Kibe. "Chaining Z-score and feature scaling methods to improve neural networks for classification". In: *Applied Soft Computing* 123 (2022), p. 108908. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.108908. URL: https://www.sciencedirect.com/science/article/pii/S1568494622002733.

[55] *Robust Scaler*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[56] Mark Jenkinson et al. "FSL". In: *NeuroImage* 62.2 (2012). 20 YEARS OF fMRI, pp. 782–790. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2011.09.015.