SEGMENTATION OF FETAL CEREBRAL MRI USING DEEP NEURAL NETWORKS

Directed by M. Guillaume Auziàs

Supervised by M. Pierre Henri-Conze

and M. François Rousseau

Meritxell Riera i Marín

Double Degree in *Telecommunications Engineering (MET) at UPC* and

in *Master DNM EEA Signal, Image, Systems, Automatic (SISEA) at IMT Atlantique*

May 2020 - Octobre 2020

I would like to thank my family for giving me the option to study abroad and helping me throughout this journey. I would like to thank my partner as well for being so supportive during this whole year. Finally, I want to thank a friend that has been there during all the problems raised from this experience and helped me in every obstacle found.

Table of contents

Abstract

Fetal *magnetic resonance imaging* (MRI) is used for monitoring and characterizing fetal brain development from the 18th gestational week to term since MRI, with its superior soft tissue contrast resolution, provides much richer details compared with US images and any other imaging technique [3].

MRI has limitations for fetal brain study due to acquisition time constraints. Because of the relatively low signal strength, fetal brain MRI examination requires relatively long imaging times [3]. However, this increases susceptibility to unavoidable fetal movements during data acquisition which makes MRI very susceptible to motion. It is necessary to minimize the effects. Because of that, the MRIs are performed with fast imaging methods [6]. Each slice provides a good quality image of a section of the anatomy. However, inter-slice motion artifacts need to be considered.

Another milestone for the study of fetal brain MRIs is the segmentation of the images. On one hand, localizing the fetal brain and obtaining a segmented mask to exclude the surrounding tissues is crucial to achieve accurate motion correction. On the other hand, segmentation of the fetal brain into different tissue classes is the key point of volumetric and morphological analysis in fetal MRI. The need for brain masks has motivated a series of studies into fetal brain extraction into fetal brain MRI.

Performing a manual segmentation is time consuming and also requires a high level of expertise. Two important problems with segmentation of fetal brain MRIs are the low tissue contrast and the dynamic development of the fetal brain across the gestational weeks.

Cortical plate (CP) segmentation on fetal MRI is particularly challenging as the fetal CP is a very thin ribbon with a thickness that is comparable to the best achievable resolution on fetal MRI scans. Another factor that makes automatic segmentation of the fetal CP challenging is the substantial variations in fetal brain morphology due to the rapid development of the brain throughout gestation [5].

Deep learning methods have often outperformed traditional machine learning and model-based methods in medical image analysis [8][9]. One of the reasons for their successful implementations is their ability to extract the features relevant for the tasks directly from the data. The networks learn themselves to extract and interpret features relevant to the segmentation task.

The *Convolutional Neural Networks* (CNN), a class of deep neural networks, is mostly applied to analyzing visual imagery since they are space invariant neural networks, which makes them ideal to work with images. To work with CNN generates the need of large sets of diverse training data to obtain successful results. In order to enlarge the size of the training set and to ensure variability of the data it is possible to use basic data augmentation techniques, such as random rotation, random translation and random noise injection.

1. Introduction

This internship has taken place in the *Methods and Computational Anatomy* (MeCA) research group at the *Institut de Neurosciences de la Timone* (INT).

1.1. Institut de Neurosciences de la Timone

The *Institut de Neurosciences de la Timone* (INT) is a research entity whose purpose is to develop high level research projects within the field of fundamental neurosciences, from cellular to cognitive, and try to erase the gap between both the clinical approaches and the fundamental approaches [1].

The different goals of the INT are:

- To explore, understand and model normal and pathological brain and spinal cord function, with an integrated approach from neuron to behaviour through neurophysiology and multi-scale imaging.
- Understand how the dynamic of both big and small neuronal networks explains the emotions, perceptions or, for example, controls the different movements of the body.
- Understand how dysfunction or cell death of neurons or glial cells cause neurological or physical problems.

Generally, the formation for the research is another important priority in the INT.

In *Figure 1*, the organization chart of the INT can be observed. The institute is directed by M. Guillaume Masson, who is a researcher in the field of neurosciences as well. It has 10 research groups, within which the MeCa group can be found. This is the group in which the internship has taken place.

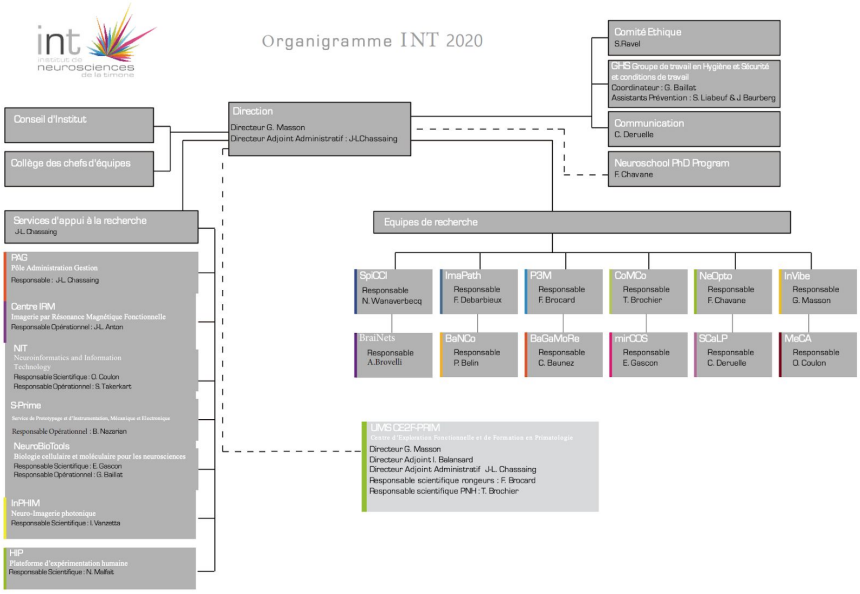

Figure 1. Organizational chart of the INT [1].

1.2. MeCa Team

The *Methods and Computational Anatomy* (MeCA) group is an interdisciplinary research team at the INT, in Marseille, France. Now it is a full INT team but in May 2014, when it was set up, it was the result of a collaboration between the INT and the L*aboratoire des Sciences de l'Information et des Systèmes* (LSIS).

The MeCA group is focused on the understanding and modelling of normal and pathological brain, more specifically, on the quantification and modelling of the cortical variability and development and their link with the white matter connectivity, using MRI on human and non-human primates [2]. Specially, the objectives of the MeCa team are:

- Quantifying and modelling cortical variability and organization.
- Quantifying and modelling cortical development.
- Studying the link between anatomy, function, and connectivity
- Developing cortical and white matter morphometry tools and applying them on large databases.

The group is organised in a way that prioritizes the communication among all the researchers in the group. The idea is to follow the other's work and, in order to do that, each week there is a meeting in which a member of the research group presents either their results or some interesting paper published in the neurosciences field.

Working in the MeCA group showed me how it is to work in a research environment and helped me to understand how research teams organise themselves. The job proposed by my tutor M. Guillaume Auzias for the internship showed me how a research project has to be properly handled. It has given me essential transversal competences such as work with other people as well as by myself, to have a critical spirit and to communicate myself to the rest of the group. It also has strengthened the knowledge acquired during my previous courses within the scope of deep learning.

With the assistance of M.Guillaume Auzias, M. Pierre-Henri Conze and M. François Rousseau, a weekly meeting took place in order to have continuous supervision and help during the internship.

1.3. Context and global aim of the project

Ultrasound (US) imaging is widely used in clinical practice for visualizing and monitoring fetal development. However, US imaging for fetal brain study is still limited by the presence of the skull and its ability to distinguish subtle differences in brain tissues [3]. Fetal *magnetic resonance imaging* (MRI) is used to evaluate the fetus in cases where the prenatal US shows suspicious or detected abnormalities that may not be apparent or cannot be accurately characterized. It is also used for monitoring and characterizing fetal brain development since MRI, with its superior soft tissue contrast resolution, provides much richer details compared with US images.

Fetal brain maturation can be studied by MRI from the 18th gestational week to term, and relies primarily on *T2-weighted* and *diffusion weighted* (DW) images [4]. It also provides advanced mechanisms to image the micro-structure and function of the fetal brain in-utero; thus it provides information that cannot be obtained by any other imaging technique [3].

MRI has limitations for fetal brain study due to acquisition time constraints. Because of the relatively low signal strength, fetal brain MRI examination requires relatively long imaging times [3]. However, this increases susceptibility to unavoidable fetal movements during data acquisition which makes MRI very susceptible to motion. It is necessary to minimize the effects.

Because of that, the MRIs are performed with fast imaging methods [6]; fast 2D slices are acquired one after each other to form a stack of slices. Thanks to the fast imaging methods, the MRI modality was introduced in clinical settings for in vivo fetal analysis as a complement to US [3]. Among the fast imaging methods, *Single-Shot Fast Spin Echo* (SSFSE) are used to acquire thick, low-resolution stacks of 2D slices that can largely freeze in plane motion. Other methods used to obtain MRI 2D slices are *half-Fourier acquisition turbo spin echo* (HASTE) , *gradient echo* (GRE) and *balanced turbo field echo* (BTFE) [6].

Each slice provides a good quality image of a section of the anatomy. However, inter-slice motion artifacts need to be considered, since either the fetus or the mother can move in between each slice acquisition [7].

In order to assess and quantify fetal brain development and pathology, it is desirable to reconstruct a single isotropic, high-resolution volume of the fetal brain in standard anatomical planes (axial, coronal and sagittal) from multiple low-resolution stacks acquired in different views.

Some reconstruction approaches have been implemented with multi-slice acquisitions in different spatial directions (axial, coronal and sagittal). These provide the radiologist a pseudo-3D visualisation of the fetal brain [3]. This results in motion-corrupted stacks of slices in multiple orientations, which does not allow 3D studies on the fetal brain. Consequently, it is necessary to estimate this inter-slice motion to correct the positioning of each slide, to correctly estimate the 3D image reconstruction of the fetal brain.

Several reconstruction methods have been proposed to cope with this issue of geometric integrity between slices due to the motion of the fetus. As a consequence, super-resolution methods have been applied to fetal MRI to improve the quality of the reconstructed images. Such reconstruction methods are essential for fetal MRI analysis [4].

Previous open-source image processing softwares were not adapted to deal with fetal MRI. There was the need of an open-source image processing toolkit primarily dedicated to fetal brain MRI. *Baby Brain Toolkit* (BTK), which relies on common libraries such as ITK, VTK, openMP, includes an image denoising algorithm, several image reconstruction methods and a probabilistic tractography technique [4].

Another milestone for the study of fetal brain MRIs is the segmentation of the images. On one hand, localizing the fetal brain and obtaining a segmented mask to exclude the surrounding tissues is crucial to achieve accurate motion correction. On the other hand, segmentation of the fetal brain into different tissue classes is the key point of volumetric and morphological analysis in fetal MRI. The need for brain masks has motivated a series of studies into fetal brain extraction into fetal brain MRI.

Performing a manual segmentation is time consuming and also requires a high level of expertise. However, fetal MRI is challenging due to the fact that the fetal brain is surrounded by the mother's organs and anatomy since the receiver coils can only be positioned on the maternal body as well as the motion of the fetus itself, which causes artifacts such as intensity inhomogeneity. Other problems with segmentation with fetal brain MRIs are the low tissue contrast and the dynamic development of the fetal brain across the gestational weeks.

As indicated above, the fetal MRIs are obtained by methods such as SSFSE in order to avoid motion artifacts. Because of that, intensity inhomogeneity artifacts may only appear in some of the slices, since they are a consequence of the motion, but they do not have to appear in the neighbour MRIs as well.

Cortical plate (CP) segmentation on fetal MRI is particularly challenging as the fetal CP is a very thin ribbon with a thickness that is comparable to the best achievable resolution on fetal MRI scans. Another factor that makes automatic segmentation of the fetal CP challenging is the substantial variations in fetal brain morphology due to the rapid development of the brain throughout gestation [5]. The reasons stated above make it difficult to develop effective and robust solutions for automatic CP segmentation in fetal MRI.

Deep learning methods have often outperformed traditional machine learning and model-based methods in medical image analysis [8][9]. One of the reasons for their successful implementations is their ability to extract the features relevant for the tasks directly from the data. The networks learn themselves to extract and interpret features relevant to the segmentation task.

The *Convolutional Neural Networks* (CNN), a class of deep neural networks, is mostly applied to analyzing visual imagery. Their main advantage is that they are space invariant neural networks, which makes them ideal to work with images.

One of the challenges of using CNN is that often they need large sets of diverse training data to obtain successful results. In order to enlarge the size of the training set and to ensure variability of the data it is possible to use basic data augmentation techniques, such as random rotation, random translation and random noise injection.

As indicated above, one of the most typical artifacts that appear in MRIs is intensity inhomogeneity. In order to let CNNs adapt and become invariant to such artifacts, it is possible to use data augmentation to deal with this problem. It is possible to randomly add synthetic intensity inhomogeneity to slices for which a corresponding reference segmentation is available.

Another challenge, as indicated before, is the fact that parts of the maternal body are also visualized and not only the head of the fetus. This is why, some of the approaches when working with fetal MRIs first automatically segment the *intracranial volume* (ICV) of the fetus in order to detect the *region of interest* (ROI). This segmentation is also implemented with CNN.

The main goal of this internship is to implement and validate a fetal MRI segmentation pipeline.

1.4. Datasets

During this project two datasets were used. Since the main goal was to segment the CP of fetal MRI scans, a fetal dataset was used in order to evaluate qualitatively the performances of the networks. For the training of these networks, since no fetal dataset with ground truth was available, a dataset of newborns was used. This dataset was also used for a quantitative evaluation of the performance of the deep neural networks used.

1.4.1 dHCP dataset

The main goal is to segment the CP of fetal brains. The dataset used for training was obtained from the *Developing Human Connectome Project* (dHCP) [10] as described hereafter.

The dHCP dataset consisted of 505 subjects, which each of them had at least one data acquisition session. In each session, four types of images were found:

- sub-*_ses-*_desc-drawem87_space-T2w_dseg.nii.gz (1)

- sub-*_ses-*_desc-drawem9_space-T2w_dseg.nii.gz (2)
- sub-*_ses-*_desc-ribbon_space-T2w_dseg.nii.gz (3)
- sub-*_ses-*_desc-restore_T2w.nii.gz (4)

The image (1) contains segmentations of the brain with 50 labels. The image (2) contains the segmentations of 9 different parts of the brain with the following labels: cerebrospinal fluid, cortical grey matter, white matter, background, ventricle, cerebellum, deep grey matter, brainstem and hippocampus. The image (3) contains four labels, one for the right hemisphere CP, another for the left hemisphere CP and the other two for the rest of the brain, one per each hemisphere. The image (4) contains the MRI scan in 3D.

The image (1) is not used throughout the whole study. The image (2) is used to obtain a total mask of the brain in order to separate it from the background and apply the normalisation. Further in the study, it is also used for obtaining more masks combining the different labels of the brain and work with multiple class segmentation. However, the first mask of the CP is obtained from the image (3), from the two labels of the CP of each hemisphere. Finally, the images for prediction are obtained from image (4).

In *Figure 2*, the first images used for the study of the performance of the Pytorch UNet. On the image of the left, the MRI image is shown. The second image shows the whole mask used for normalisation of the MRI image in order to discard the background; it is obtained, as indicated before, with all the labels of the image (2). The third image shows the first CP mask used for training, obtained from the image (3).

Figure 2 - Ground truths for the single label study.

In *Figure 3*, in the left image the MRI scan is shown as well. The three remaining figures are obtained from the image (4) using the different labels. These are used for testing the neural network for multiple classes. The second image, *drawem9 Mask (1+5)*, corresponds to the mask that combines the label 1 (cerebrospinal fluid) with label 5 (ventricle). The third image, *drawem9 Mask (2)*, corresponds to the mask obtained from the label 2 (cortical plate). The fourth image, *drawem9 Mask (3)*, corresponds to the mask obtained from the label 3 (white matter). These three labels are merged in the same image, creating a multilabel ground truth used for three different segmentations.

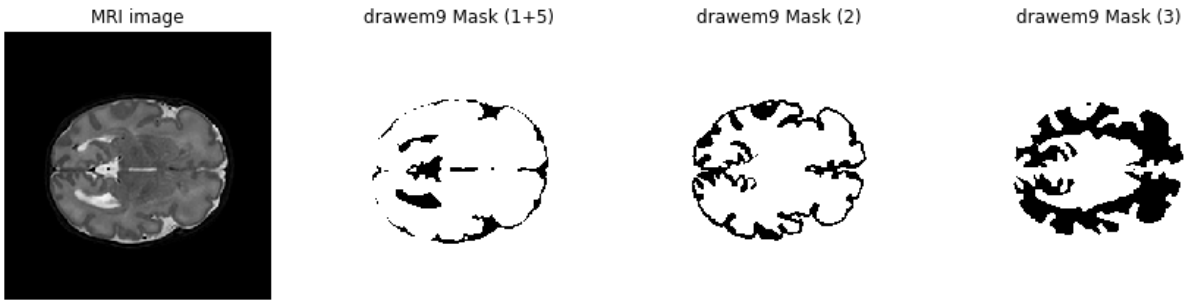

Figure 3. Ground truths for the multilabel study.

It is important to say that the CP mask used for single label classification is obtained from the image (3), which is different from the ground truth of the CP used for the multi label classification, obtained from the image (4). This is done because the mask from the image (3) is more anatomically correct than the one obtained from the image (4). However, for the multilabel classification the masks need to be coherent with each other and, this is why, all the masks used for this task are extracted from the same image (4).

Each image (4) was a 3D stack obtained from an MRI session. In order to be able to use the dataset for the learning, it was necessary to create 2D images for both the image and the respective mask. To do that, the slices in which the CP mask was not null, were saved as 2D images, both the MRI scan and the ground truth for the CP. The slices were obtained by cutting the 3D volume following the axial plane. This procedure of generating 2D slices from the 3D images was applied for all images and masks for the dHCP data, and for the images for the fetal data.

Importantly, the training dataset dHCP was obtained from newborns MRIs, which are older than the fetuses from the data acquired in Marseille. Anatomical differences due to different levels of brain maturation are expected between the two datasets. As a consequence, we restricted the selection of the images used for training to the 40 youngest subjects of the dHCP dataset, corresponding to 29 to 36 weeks of gestational age at MRI acquisition date. Selecting the youngest targets, helps to obtain more accurate results rather than selecting them randomly or selecting the oldest ones, since, as indicated before, the brain evolves rapidly during the gestation weeks and even after the birth. The youngest the newborn, the more similar anatomically speaking to the fetuses brains.

Another important aspect to take into consideration is to check that both datasets are compatible. This means that the distribution of the pixels of each 3D image is similar. In order to do that, a normalisation in 3D was performed. This is explained in *section 2.4*.

1.4.2 Fetal dataset

The data obtained from Aix-Marseille University consisted of 27 fetal brain MRI scans, with MRI scans within 27 weeks and 36 weeks, for which no ground truth segmentation (in particular for CP) was available.

The anatomical image was accompanied with a total mask covering the entire fetal brain which served in order to separate it from the background. This allowed us to normalise the images efficiently as detailed in *section 2.4.2*. The images were 3D stacks of slices; the generation of 2D slices was needed in order to obtain the predictions.

1.5. Objectives of the internship

The organization of this project is divided in four objectives as follows:

The first objective consists of searching in the literature for potential effective solutions that would fill our needs. More specifically, CNN have demonstrated their effectiveness in the segmentation of medical images. We will review the recent papers in this research field for potential application of CNN technique to tissue segmentation in early development MRI.

The second objective consists in testing two neural networks (*Pytorch UNet* [11] and *FetalCPSeg* [5]) for the CP segmentation with newborns MRI images. Indeed, the data used for the neural network learning on newborn data are obtained from the dHCP [10] database. In this database, ground truth segmentation masks are provided which allows for proper quantitative evaluation of the learning procedure by the two techniques. For the FetalCPSeg the study of the attentive module and the deep supervision modules had an important role as well as plugging the Pytorch UNet within the code.

Once evaluated on newborns MRI images, the third objective consists in using these neural networks for segmenting the CP in fetal images. The application in fetal MRI remains challenging due to the wide variation in the quality of the images to be processed.

Since the application was successful for the CP tissue, we pursued the project with the fourth objective was to investigate the ability of the Pytorch UNet to achieve multilabel segmentation (several tissues). The masks used for multilabel segmentation have been explained previously, in *section 1.4.1*.

2. Review of the literature and experimental setup

2.1 State of the Art

Several studies have been conducted in this field.

In the work of [7], two segmentation approaches were implemented and tested. First, a deep fully convolutional neural network based on the U-Net [11] was developed to segment brain sections independently on original 2D fetal MRI slices. It improves segmentation accuracy, since multi-scale information is used. Second, a voxelwise approach was developed. The architecture consists of three fully convolutional paths. Each contains four convolutional layers followed by a *rectified linear unit* (ReLU) nonlinear function and a *batch normalization* (BN). In this method, no preprocessing method was used.

In the study of Ebner, M. et al. [6], a fully automatic framework for fetal brain MRI reconstruction to obtain *high-resolution* (HR) visualizations in standard anatomical planes from multiple *low-resolution* (LR) input stacks is proposed. There are automatic localization, segmentation and reconstruction parts.

For automatically localizing the fetal brain region in each input LR stack and obtaining a 3D bounding box of the fetal brain a CNN is used. Within the bounding box, another CNN is used to automatically generate a fine segmentation of the fetal brain. For the automatic high-resolution volume reconstruction part different stages are included: first, the two-step iterative *Slice-to-Volume Registration* (SVR); second, an outlier-robust *Super-Resolution Reconstruction* (SRR) step followed by a fast and robust standard anatomical template space alignment step.

In the article of N. Khalili, et al. [8], the proposed approach is applied to 2D slices of images reconstructed in a standard way, i.e. without reconstruction to HR volumes. The method proposed has two parts. First, the ICV from the fetal MRI slices is identified using a CNN. Then, the identified region is segmented by another 2D CNN.

In this method, also a data augmentation technique that synthesizes intensity inhomogeneity artifacts is proposed in order to improve robustness against these artifacts. Also, the fetal brain segmentation is performed into seven classes (*cerebellum* (CB), *basal ganglia and thalami* (BGT), *ventricular cerebrospinal fluid* (vCSF), *white matter* (WM), *brain stem* (BS), *cortical gray matter* (cGM) and *extracerebral cerebrospinal fluid* (eCSF)), instead on focusing only on WM, cGM and vCSF.

In the article of J. Li, Y. Luo and L. Shi et al. [9], a two-step framework using the deep learning method is proposed. It consists of two *fully convolutional networks* (FCN). The first shallow FCN locates the fetal brain and extracts the ROI containing the brain. Afterwards, within the ROI extracted, an extra deep *multi-scale FCN* (M-FCN) refines the segmentation and produces the final brain mask by leveraging the multi-scale information and residual learning blocks. Dilated convolutional layers were employed in both FCNs to control the size of feature maps and increase the field of view.

Finally, the two networks that are going to be studied are a neural network that has a basic UNet architecture, which consists of a contracting path and a symmetric expanding path [11] and a neural network which has a backbone fully convolutional encoder-decoder network with stagewise forward skip connections and a stagewise attention refinement module [5]. These two networks were chosen since, firstly, the basic UNet is one of the more classical architectures in neural networks and, secondly, the second architecture was specifically designed for dealing with our problem. These two networks are explained in more depth in the following *section 2.2*.

2.2 Convolutional Neural Networks

As stated in the *section 1.3.* the *Convolutional Neural Networks* (CNN) are a class of deep neural networks that is mostly applied to analyzing visual imagery since they are space invariant. Before talking about CNNs it is important to talk about the concept of *deep learning* (DL).

DL is a subset of *machine learning* (ML) in *artificial intelligence* (AI). AI refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions as well as exhibit traits associated with a human mind, such as learning and problem-solving [12]. ML, a field of AI, is the concept that a computer program can learn and adapt to new data without human intervention [13]. DL imitates the workings of the human brain in processing data and creating patterns to use in decision making. It has networks capable of learning unsupervised from data that is unstructured or unlabeled [14].

Images are considered a type of unstructured data. Unstructured data is information that is not organized in a pre-defined manner. Since we are working with images DL architectures are suitable for our study. Furthermore, both networks that will be studied are CNNs, which makes it even better for analyzing images due to their space invariance.

CNNs are a class of DL generally used for visual imagery. CNNs were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex [15].

CNNs are a specialized kind of neural network for processing data that has a known, grid-like topology. Image data can be considered as a grid of pixels, 2D or 3D. The CNNs employ the operation of *convolution*, which is a kind of linear operation, instead of general matrix multiplication. The mathematical expression of the convolution is denoted as:

$$s(t) = \int x(a)w(t-a)da = (x * w)(t)$$

In the DL terminology, the first argument x is referred as the input and the second argument w is referred to as the *kernel*. The output would be the *feature map*. Since we are working with images, the operation of convolution should be discretized and, moreover, since the images are 2D arrays we will be using a 2D kernel.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \qquad \rightarrow \qquad S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n)$$

The advantages brought by the convolution operation are three-fold:

- Sparse interactions.

When using matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each input unit and each output unit, every output unit interacts with every input unit. When using convolution instead of matrix multiplications, the kernel is smaller than the input, which results in sparse interactions. Thanks to that, the parameters needed to be stored are fewer, which results in both less memory requirements of the model and improves its statistical efficiency. Having m inputs and n outputs, the matrix multiplication will have $m \times n$ parameters and have $O(m \times n)$ runtime. If the connections each output may have are reduced to k, then the convolution requires $k \times n$ parameters and $O(k \times n)$ runtime.

- Parameter sharing.

With a traditional neural network, each element of the weight matrix is used once when computing the output of a layer. In a CNN, each element of the kernel is generally used at every position of the input, meaning that instead of learning a separate set of parameters for every location, only one set is learnt. This reduces the storage requirements of the model to k, which is several orders smaller than m.

- Equivalent representations.

In the case of convolution, the particular form of parameter sharing causes the layer to have a property called equivariance to translation. To say a function is equivariant means that if the input changes, the output changes in the same way [16]. This can be represented mathematically as: $f(g(x)) = g(f(x))$. One of the transformations that convolution is invariant is translation. This is really important to explain why CNN are translation invariant.

Finally, one last advantage of the convolution is that it lets you work with data of variable size.

Another important thing to consider about CNNs is the *pooling layer*. This layer uses a pooling function which replaces the output of the network at a certain location with a summary statistic of the nearby outputs [16]. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pooling helps to make the representation become invariant to small translations of the input. The typical operation is the *max pooling*, which can be observed in *Figure 4*.

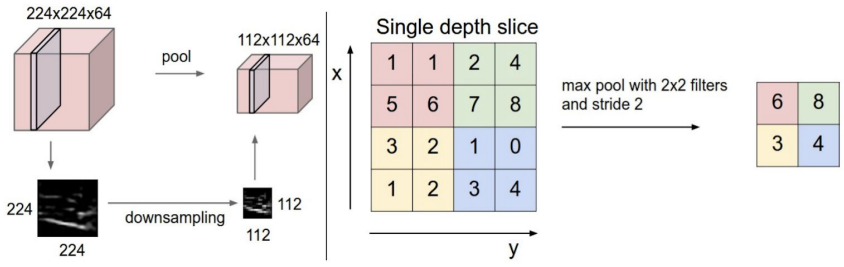

Figure 4 - Example of max pooling [17].

The *Rectified Linear Unit* (ReLU) is a function used as an activation function. The activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input [18]. The ReLU is a linear function that will output the input directly if it's positive or zero if not. It is the most widely used activation function for several types of neural networks because it makes the model easier to train and generally achieves better performance than with other activation functions. In *Figure 5*, the ReLU activation function is plotted.

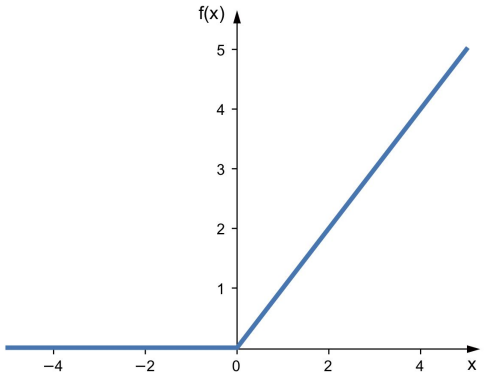

Figure 5 - Rectified Linear Unit (ReLU).

After clarifying some basic concepts of the CNNs, the two networks that will be studied are explained in more depth in the following subsections.

2.2.1 U-Net: Convolutional Networks for Biomedical Image Segmentation

This neural network consists of a basic UNet architecture [11], which is shown in *Figure 6*. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization.

The contracting path consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a ReLU and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step, the number of feature channels is doubled.

The expansive path consists of an upsampling of the feature map followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.

The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

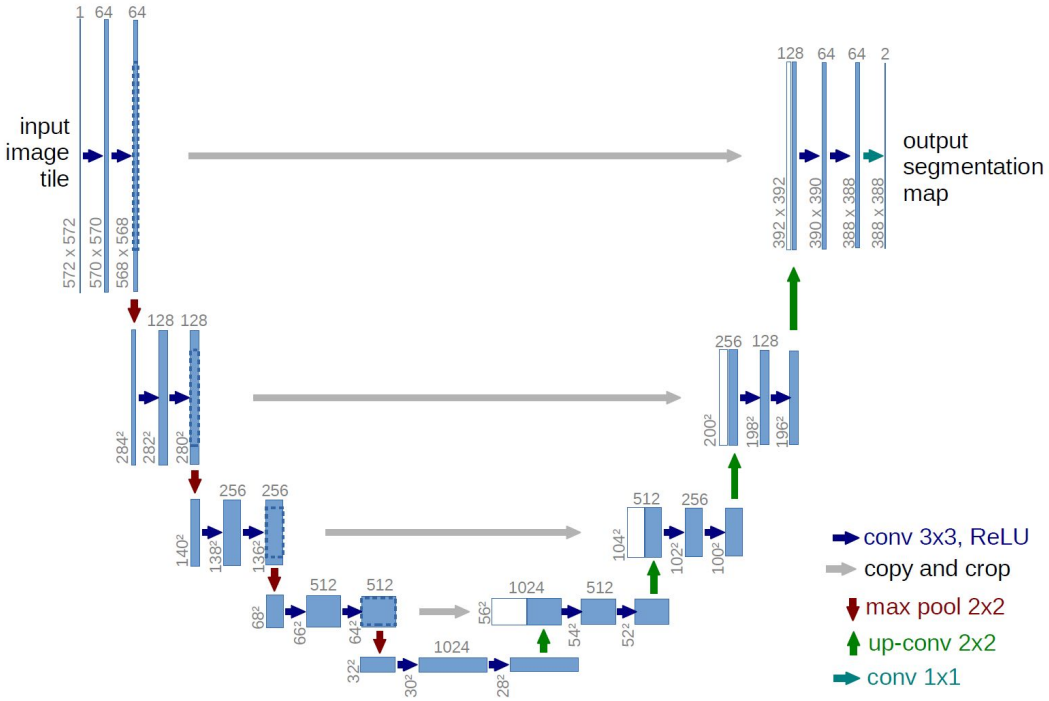

Figure 6 - U-net architecture.
Each blue box corresponds to a multi-channel feature map. White boxes represent copied feature maps. The arrows denote the different operations. [11]

This network is based on the architecture of a *fully convolutional network* (FCN) [19]. The main idea is to supplement a usual contracting network by successive layers, but instead of pooling operators, upsampling operators are used. Consequently, these layers increase the resolution of the output. High resolution features from the contracting path are combined with the upsampled output in order to be able to localize. Afterwards, a successive convolutional layer can learn to assemble more precise output.

Based on this FCN architecture, in the paper some modifications were done. One important modification is that in the upsampling part there also is a large number of feature channels, which allow the network to propagate context information to higher resolution layers [11]. Consequently, the expansive path is similar to the contracting path and yields to a u-shaped architecture. The network does not have any fully connected layers and the segmentation map only contains the pixels for which the full context is available in the input image. To predict the border pixels, the missing context is extrapolated by mirroring the input image. This strategy allows the segmentation of arbitrarily large images due to an

overlap-tile strategy, which means that the segmentation is done part by part for the whole image when the image is too large.

The code used and modified for this study can be found in the github [20]. In this pytorch code, the weight loss is the cross entropy function. This network is adapted to make multilabel prediction; however, it is not prepared to deal with imbalanced data. Because of that, few changes were made in order to use a weighted cross entropy loss function.

2.2.2 Fetal CPSeg: A Deep Attentive Convolutional Neural Network for Automatic Cortical Plate Segmentation in Fetal MRI

First of all, it is important to remark that this network was designed to work with 3D images. However, for this study, it has been adapted in order to work with 2D images.

To deal with the substantial variability in the size, shape, and complexity of the thin fetal brain CP, a new network architecture with novel attention modules using mixed kernel convolutions was developed.

The architecture of the proposed deep attentive fully CNN for CP segmentation in fetal MRI is shown in *Figure 7*. It consists of a backbone network with an encoder-decoder architecture with forward skip connections from the encoder stages to the corresponding decoder stages. Then, it is followed by a stage-wise attention refinement module that leverages mixed kernel convolutions to capture multi-scale contextual information.

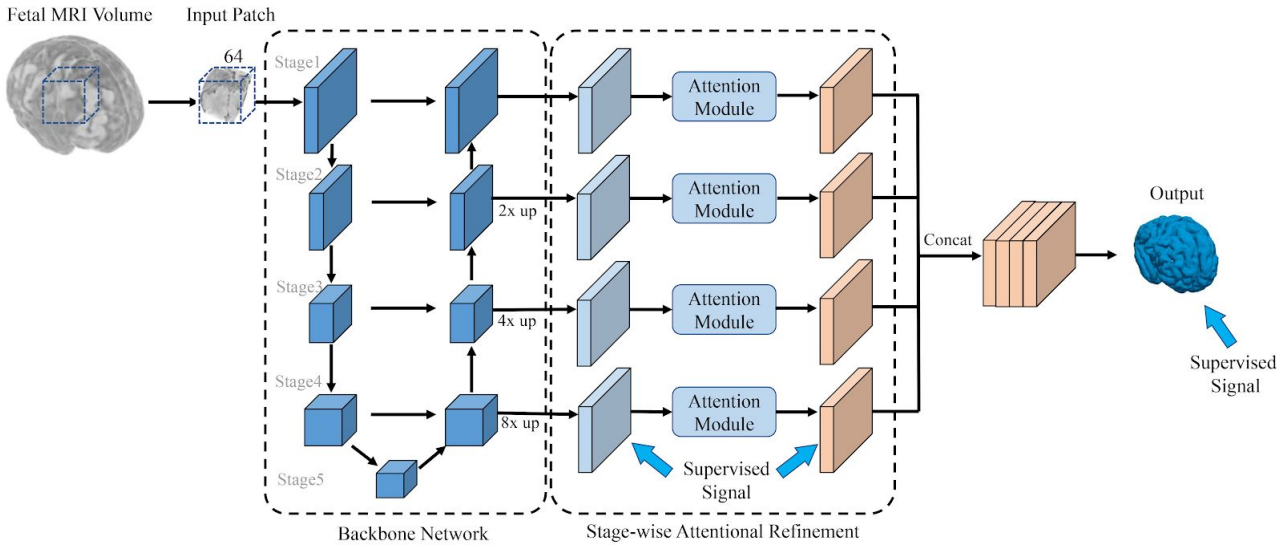

Figure 7 - Schematic illustration of our proposed CNN architecture.
Consists of a backbone fully convolutional encoder-decoder network with stagewise forward skip connections, and a stagewise attention refinement module. The network takes sliding, overlapping 2D patches of size 64x64 from the input image, and using training data, learns to generate a 2D segmentation of the input image. [5]

Figure 7 is obtained from the corresponding paper [5], this is why the patch taken is a 3D image. However, as already indicated, this network has been adapted to work with 2D patches. Any other 3D references in this section obtained from the paper have been changed to 2D in the code.

The *backbone network* (BBN) extracts a series of feature maps with different resolutions. The shallower feature maps contain HR details used to get an accurate delineation of the CP boundary and the deeper feature maps contain coarser information used to predict the overall outline of the CP. It has a basic UNet architecture with forward skip connections and convolutions with shortcuts or residual connections, hence, it results in a residual UNet architecture. The residual connections is a connection that bypasses one or more layers in the network. In *Figure 8* an example of a residual block can be observed.

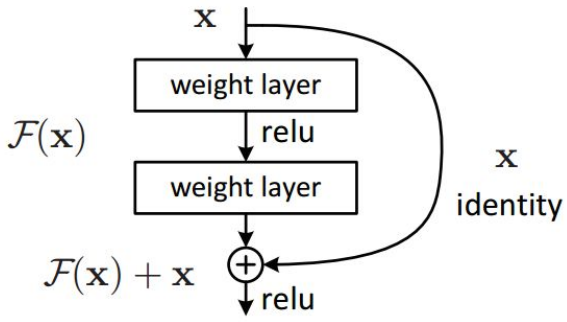

Figure 8 - Example of a single residual block. [21]

Generally, it is known that when working with neural networks the accuracy increases with the number of layers. However, there is a limit of the number of layers that can help the increase of the accuracy. If the number of layers is too big, it can even turn out to be a saturation of the accuracy that at some point will eventually degrade. This is known as the degradation problem. In this case, we know that the shallower networks work better than their deeper counterparts, which is counter-intuitive. One solution to that, would be to skip some of these extra layers. Because of that, some skip-connections are added or also known as residual connections, since the layers in a residual network are trying to learn the residual ($R(x)$), whereas the layers in a traditional network are learning the true output ($F(x)$) [21].

$$R(x) = Output - Input = F(x) - x \qquad \rightarrow \qquad H(x) = F(x) + x$$

In this architecture, the shortcuts or residual connections are added in the input of each decoder stage of the backbone network, between the output of the previous decoder stage and the output of the corresponding encoder stage. The input of each stage is processed by two 3x3 convolutional layers followed by a BN [22] and a parametric PReLU [23] is used for activation.

The sizes of feature maps in the encoder part are 16, 32, 64, 128 and 128; the number of features increases as their size shrinks.

Every feature map computed by the backbone network is then upsampled using bilinear interpolation to the size of the input patch. After that, a convolution with a kernel of 1x1 is applied to every feature map

to create 16 features in each of the feature maps. Then, these feature maps go through *deep supervision modules* (DSM) [24]. These modules improve the gradient flow and encourage learning more useful representations.

Similar DSM are also used on the outputs of the *attention modules* (AM) and merge the resulting feature maps via concatenation. These feature maps go through two convolutional layers followed by BN and PReLU to produce the CP probability map.

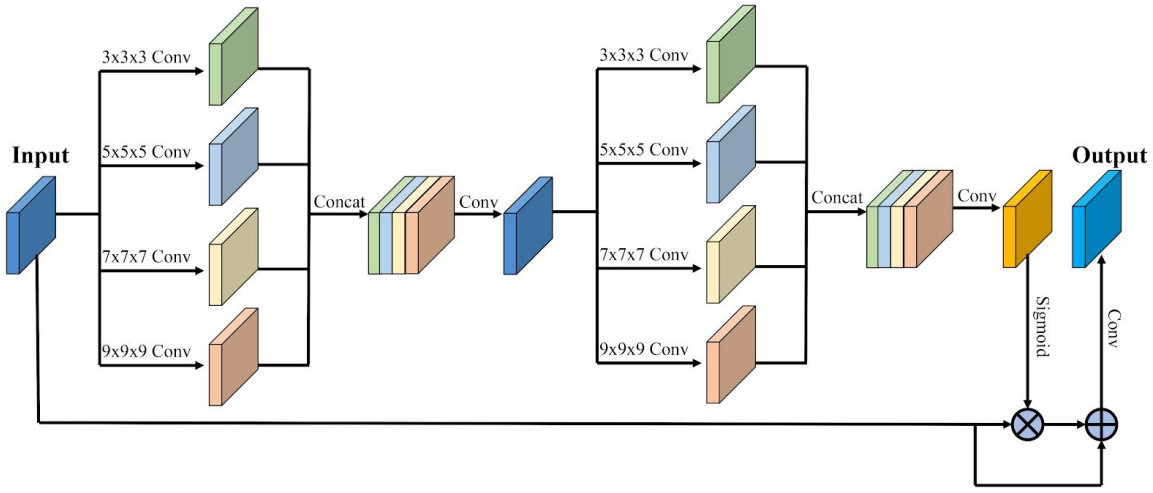

Figure 9 - Attention module architecture. [5]

The AMs are the most special part of this network. In *Figure 9* their architecture can be observed. Their goal is to increase the network's ability to capture multi-scale details of the brain CP, by increasing the richness of the information of the multi-scale feature maps learned by the backbone network.

The AM takes the feature maps from each stage of the BBN and outputs a refined attentive feature map of the same size. The operation that takes place in the AM is:

$$F_i^{'} = f_i(F_i;\theta) \otimes F_i + F_i$$

θ represents the parameters of the module, for example the weights of the convolutional layers. F_i is the input of the AM, i. e. the features maps obtained from the i_{th} stage of the BBN. $F_i^{'}$ is the output of the AM at the i_{th} stage. Finally, $\otimes$ denotes the element-wise multiplication.

The AM processes the input feature map F_i with a group of convolution blocks. Their sizes go from 3x3 to 9x9. Each convolutional operation is followed by BN and PReLU. The generated feature maps are concatenated and passed through an exact same block and, afterwards, a convolutional layer with a kernel size of 1x1 is used to merge these multi-scale feature maps into a single feature map. As last steps, a sigmoid activation function is applied to obtain the attentive map A_i and it is multiplied element-wise with the input feature map. This is done to encourage attention on the relevant locations in the feature map. Finally, the refined attentive feature map is added to the input feature map F_i in

order to reduce the difficulty of learning the attentive map. Also, BN is used on the two branches before adding both feature maps together to ensure that the values of the two feature maps are not differently distributed.

The loss function used for the training is a weighted binary cross entropy.

$$L_{wbce} = \frac{1}{N}\sum_{i=1}^{N}[\alpha\, g_i\, log\, p_i + (1 - g_i)\, log(1 - p_i)]$$

N is the number of pixels in the patch; g_i is the binary grountruth CP probability map at pixel i; p_i is the predicted CP probability map at pixel i; α is the weight hyperparameter which is set independently in every training mini-batch.

The overall loss is the weighted sum of the weighted losses at different points in the network.

$$L_{total} = \sum_{i=1}^{n} w^i L^i_{signal} + \sum_{j=1}^{n} w^j L^j_{signal} + w^p L^p_{signal}$$

The first term is the weight and the loss of the points of supervision at stage i of the backbone network; the second term is the weight and the loss of the points of supervision at stage j of the attentive refinement network; finally, the third term is the weight and loss computed at the final network output. n is the number of stages for both first and second term.

The code used and modified for this study can be found in the github [25]. As already indicated, every 3D step in the code has been changed to work with 2D patches.

2.3 Hyperparameters in Neural Networks

Hyperparameters, in NN, are the variables which determine the network structure and the variables which determine how the network is trained [26]. The hyperparameters are set before training. The focus will be set to the hyperparameters related to the training algorithm since the network structure is already fixed, for both the *Pytorch-UNet* [20] and the *FetalCPSeg* [25].

Firstly, it is important to point out the typical hyperparameters related to the training algorithm. The *learning rate*, which defines how quickly the network updates its parameters; if it is small it slows down the convergence of the algorithm and makes it converge smoothly and, if it is big it speeds up the learning process but it may cause a non-convergence in the algorithm [26]. The other two most used hyperparameters are the *number of epochs* and the *size of the batch*, which are explained later.

The first hyperparameter to remark is the *number of epochs*. Passing the entire dataset through a neural network is not enough; the dataset must be passed multiple times to the same neural network to update the weights multiple times, not just once. The *number of epochs* is a hyperparameter that defines the number of times that the learning algorithm will work throughout the entire training dataset [27]. The *number of epochs* is traditionally large allowing the learning algorithm to run until the error from the model has been sufficiently minimized [27].

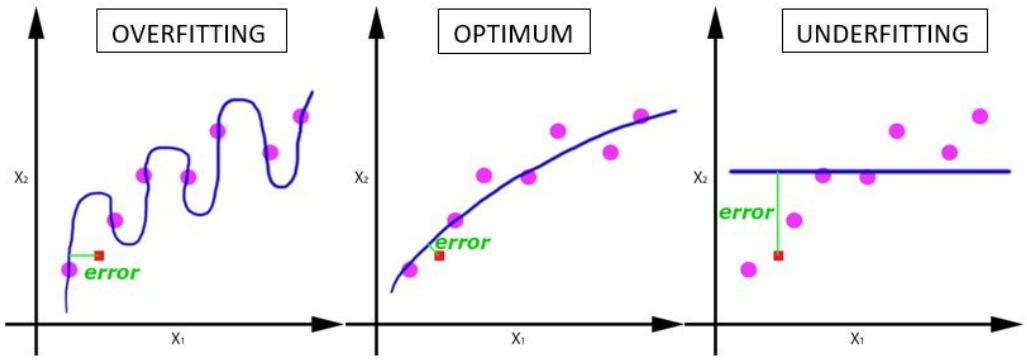

Figure 10 - Graphical examples of the different resulting cases when training a neural network [28].

In *Figure 10* the process of updating the weights several times is shown. As the number of epochs increases, the times the weights are updated as well. When the *number of epochs* is not enough, the network finds itself in underfitting, since the weights are not modified enough; when the *number of epochs* increases it can reach the optimal or if it is more than necessary the network is overfitted, which means that it only works well for a specific group of images, the training set.

In DL, usually the data is too big to be passed to the network all at once. To overcome this problem, the data is split into smaller sizes and passed to the NN [28]. These data splits are called *batches*, and their size, which is fixed, is the *batch size*.

The *batch size* is a hyperparameter that defines the number of samples to work through before updating the internal model parameters [27]. A training dataset can be divided into one or more batches.

The value of the *batch size* usually goes from 32 to 512 [29]. It has been observed that when using a larger *batch size* there is a significant degradation in the quality of the model [29]. However, for technical restrictions during the internship, the first studies were restricted to a *batch size* of 13. As soon as it became possible, a few studies with respect to the optimal *batch size* were done which can be found in *section 3.1.1.1*, and all the results were updated with the new optimal *batch size*.

For the *Pytorch-UNet* there are other hyperparameters that can be modified, such as the *downscaling factor* of the images and the *percentage of data* that is used for validation as well as the three already mentioned (l*earning rate*, *number of epochs* and *batch size*). The *downscaling factor* has been set to 1 in all the tests, even though the default value is 0.5, because it is indicated that for obtaining better results, even though more memory is used, it must be set to 1. The *percentage of the data* used for validation to 10%. The *learning rate* has also been fixed to 0.1 for all learning processes.

For the *FetalCPSeg*, the hyperparameters that can be modified are the *learning rate*, the *number of iterations* (which is the same as the *number of epochs*) and the *batch size* as in the *Pytorch-UNet*. Since the network is optimized with the *Adam optimizer* [30] another hyperparameter needed is the *weight decay*. In addition, since the images are split into patches the *patch size* is another hyperparameter that can be tuned, which by default was set to 64. Finally, there are two more hyperparameters who indicate whenever it is necessary to save the new model or reset the loss during the iterations.

2.4. Normalization of datasets

In this section, the normalization of both datasets is explained. This normalization is necessary in order to make sure that the different pixel distributions of the 3D MRI images are close and similar. It is needed in the dHCP dataset in order to make the training efficient, as well as ensuring that the prediction of the images from this dataset is compatible with the training previously done. It is needed in the fetal dataset as well for the same reason; the compatibility of the pixel distributions of the images in both training and testing datasets is necessary in order to have efficient predictions.

2.4.1. Normalization of the dHCP dataset

For normalizing the dHCP dataset, centering in the mean and dividing for the standard deviation was applied. This normalisation was done in the 3D images for both datasets. For an efficient normalisation, it is important to separate the brain from the background and then, apply normalisation just in the pixel values of the brain. For that, a total mask of the brain is necessary. This can be seen in *Figure 11*.

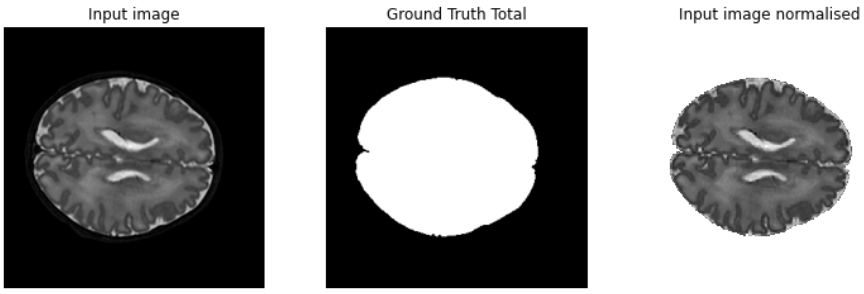

Figure 11 - Normalisation of the MRI images of the dHCP dataset with the total mask.

The image in the left shows an axial cut of the whole 3D MRI scan. Then, in the second image, the total mask of the brain can be observed. Finally, the last image shows an axial cut of the 3D MRI scan but separating the brain from the background and normalising it.

In *Figure 12*, the histograms of the pixel value distribution of the images can be observed. The left histogram shows the distribution of the images without normalization and the right histogram shows the distribution of the images with normalisation. As expected, the images normalised are closer and have more similar distributions.

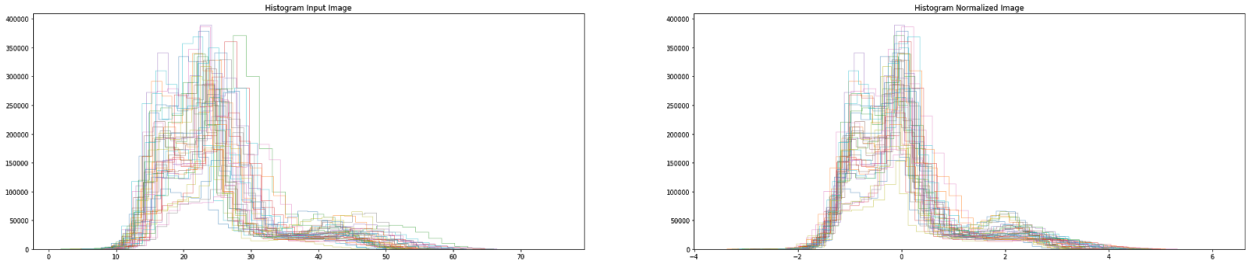

Figure 12 - Histograms of the distributions of the 3D MRI images of the dHCP dataset without normalisation (left image) and with normalisation (right image).

In *Figure 13*, the histograms plotted are used to check that the data augmentation (rotation and translation) do not change the image distribution. This histogram is obtained after generating 2D images, and applying the data augmentation to these images. In order to check the results, 40 random images were chosen among 5337 2D images generated from the 40 youngest subjects of the dHCP dataset. As it can be observed, the distribution of the images do not change after applying data augmentation.

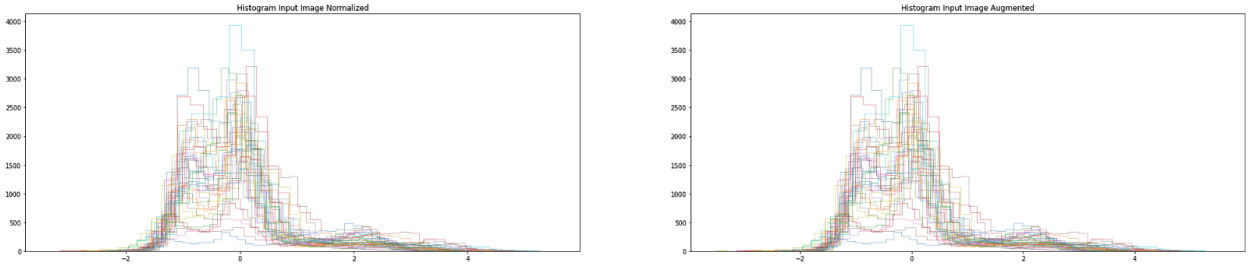

Figure 13 - Histograms of 40 2D slices randomly selected without data augmentations (left image) and with data augmentation (right image).

2.4.2. Normalization of the fetus dataset

As done for the dHCP dataset, the normalisation consisted on centering in the mean and dividing for the standard deviation. This normalisation was done in the 3D images as well. As it was done for the dHCP dataset, it was important to separate the brain from the background and then, apply normalisation just in the pixel values of the brain. For that, a 3D total mask of the fetus brain is necessary. This can be seen in *Figure 14*.

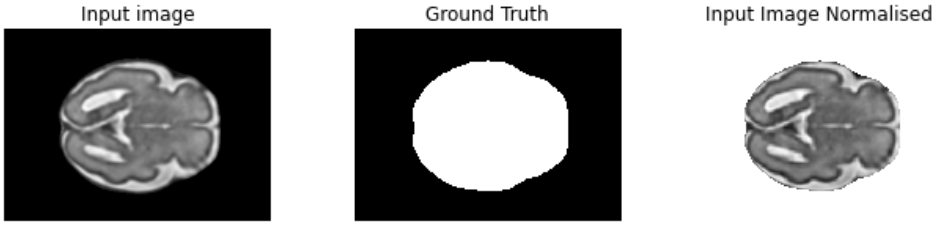

Figure 14 - Normalisation of the MRI images of the fetus dataset with the total mask.

The image in the left shows an axial cut of the whole 3D MRI scan. Then, in the second image the total mask of the fetal MRI scan is seen. Finally, the third image shows an axial cut of the 3D MRI scan separating the brain from the background and normalising it.

In *Figure 15*, the histograms of the different 27 fetus 3D images can be observed. On the left histogram, the distributions of the images without normalisation can be observed, and in the right histogram, the distributions of the images after normalization can be observed. As expected, the histograms after normalization are closer and have more similar distributions.

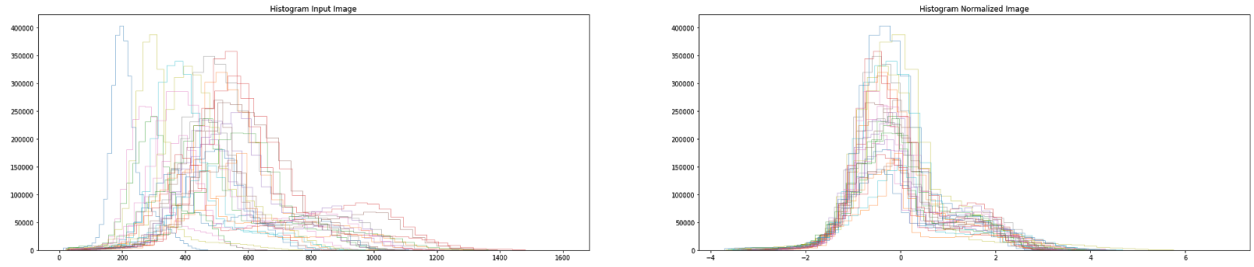

Figure 15 - Histograms of the distributions of the 3D MRI images of the fetus dataset without normalisation (left image) and with normalisation (right image).

Finally, the last step to check if both datasets have similar distributions and, hence, are compatible to be used in the same convolutional network, all histograms of the 3D distributions of normalised images histograms from both datasets are plotted together in *Figure 16*.

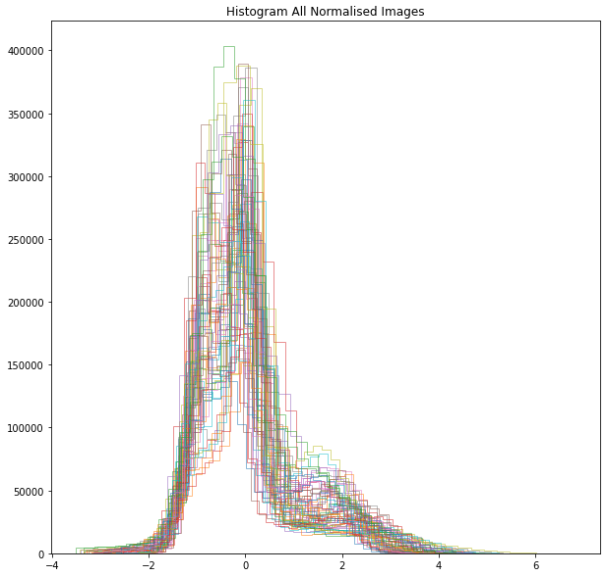

Figure 16 - Histogram of the distributions of the 3D normalised images of both datasets (dHCP and fetus).

After the normalisation, it can be observed that all 3D MRI scans from both datasets have a similar distribution and, hence, can be used in the same neural network.

2.5 Data augmentation

As indicated before, one way to improve the performance of a CNN is using more data for training. However, the amount of images is limited and if this amount is not big enough it can contribute to overfitting. Because of this, one of the most efficient ways to generate more data is with data augmentation. The main idea is to generate fake images through the original dataset, by modifying them, for example, rotating the images.

Even if the amount of images of the training dataset is big, they might not be well distributed. Consequently, another reason to use data augmentation would be to cover all the normal characteristics that are expected to be found. The performance and robustness of a trained model is directly linked to the amount of training data as well as the different qualities of the image and the characteristics they cover. Using mathematical models that describe natural variations, such as rotation or translation as well as any noise appearance in the images, it is possible to expand the training data and enrich it with artificial variations. Doing that, it is expected to increase the performance and robustness of the neural network with respect to the natural variations added in the training dataset [31].

The main task facing a classifier is to be invariant to a wide variety of transformations. Operations like translating, rotating and scaling the training images can often greatly improve generalization, even if the model has already been designed to be partially invariant [16].

2.5.1 Affine Transformations

The most common and immediate transformations for a training dataset are the *affine transformations*. An *affine transformations* is a geometric transformation that preserves lines and parallelisms but not necessarily distances and angles [32]. The two affine transformations chosen for the training dataset are *rotation* and *translation*. The other affine transformations would not do anything to improve the performance of the neural network but even they might confuse it. However, rotation and translation cover the motion errors that can be encountered in the MRI scans, due to the mother or the child. If a random rotation of a few degrees is applied as well as a random translation of small values, this motion in between the MRI 2D slices could be covered.

This allows the network to learn invariance to such transformations. This is particularly important in biomedical segmentation, since both translation and rotation are the most common variations in MRI scans and can be simulated efficiently.

There are two things that are important to be mentioned. Firstly, it is important to not apply any transformations that would affect the correct performance of the CNN, such as vertical or horizontal flips, as indicated above. The brains are not symmetric and applying flips would only confuse the

classification. Secondly, even if some transformations seem to be really helpful it is important to consider the difficulty of implementing such transformations.

2.5.2 Noise injection

Neural networks prove not to be very robust to noise. One way to improve the robustness of neural networks is simply to train them with random noise applied to their inputs. Because of this reason, different types of noise were injected to the training images.

Image noise is random variations that are introduced in the images which produces undesirable effects, such as artifacts, unrealistic edges and blurred objects, among others. The three more common types of noise that affect images are *Gaussian noise*, *Salt and Pepper noise* and *Speckle noise*.

Generally MRI images contain a significant amount of noise caused by operator performance, equipment and the environment, which leads to serious inaccuracies [33].

Gaussian noise, or *normal noise*, is caused by natural sources such as thermal vibration of atoms and discrete nature of radiation of warm objects. It generally disturbs the gray values in digital images. Because of that, the Gaussian noise model normalizes the histogram with respect to gray value [31].

Salt and Pepper noise is a combination of two impulse noises: *salt noise* (random bright pixels) and *pepper noise* (random dark pixels). Consequently, Salt and Pepper noise is an impulse noise resulting from a combination of random bright and dark pixels. It refers to a wide variety of processes that result in the same basic image degradation: only a few pixels are noisy, but they are very noisy. The effect is similar to sprinkling white and black dots -salt and pepper- on the image [34]. MRI images are corrupted by Salt and Pepper noise because of the sensor faults of image acquisition devices, which causes the sharp and sudden disturbances in the image signal which in turn degrades the image quality [35].

Speckle noise, or *texture* in medical imaging, is a granular noise that appears when a sound wave pulse arbitrarily interferes with small particles or objects on a scale comparable to the wavelength. It is the result of the destructive and constructive coherent summation of echoes [36]. This noise is mostly encountered in *Synthetic Aperture Radar* (SAR) images or *Ultrasound* (US) images [37], in which affects edges and fine details which limit the contrast resolution and make diagnostic more difficult, rather than in MRI images. However, since sometimes it is encountered in MRI images, it was decided to inject it in order to augment data.

2.6. Post processing

2.6.1 Threshold selection

In the field of digital image processing, *thresholding* is the method of segmenting images. When having a grayscale image and the thresholding is applied, a binary image is obtained.

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T ($I_{i,j} < T$) or a white pixel if the image intensity is greater than that constant [38].

There are different types of methods in order to binarize an image. These methods compute automatically the value of the threshold, instead of fixing it from the beginning. The idea of computing the threshold remains on improving the performance per each image specifically. These methods can be classified depending on which parts of the images are analyzed: histogram shape-based methods, clustering-based methods, entropy-based methods, object attribute-based methods, spatial methods and local methods.

Another way to classify the different types of thresholding methods is if they perform a global binarization (Fixed thresholding method, Otsu method and Kittler method among others) or if they perform a local binarization (Niblack method, Adaptive method, Sauvola method and Bernsen method among others) [39].

One of the most classical approaches for selecting the threshold is the Otsu method, which performs a global binarization of the image. This method is based on the shape of the histogram [40]. It consists of iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The main idea is to find the threshold for which the sum of foreground and background spreads is at its minimum.

However, after studying a little bit this field of research, it was concluded that the method that was going to be used for the different experiments is the *Fixed Thresholding Method*, which consists of assigning 1's or 0's for all pixels in the image for a fixed threshold value.

This thresholding method will be applied to the prediction masks obtained by the two neural networks. The decided value is 0.5 for both of them for two reasons: for several images it was observed to be the optimal one, the one that gave a higher dice coefficient; it was the default value used for both networks.

When working with multiple classes, the problem of thresholding can be addressed in a simple way. For the multilabel classification for the Pytorch UNet, the prediction is given in the following way: the output image has as many channels as labels need to predict plus one for the background. Therefore, if we want to predict k labels, the output will have $k+1$ channels. In order to generate the final image with multilabel prediction, the idea is to generate a new image in which each pixel will be assigned the label that had the biggest probability for that pixel. This way, just one 2D image with one channel will be obtained and per each pixel there will be just one label assigned.

As it can be observed in the procedure explained above, in the case of multilabel classification there is no need for any thresholding method.

2.6.2. Mathematical Morphology

Another way to improve the results obtained is using post processing techniques, which are methods who try to increase the quality of the predicted masks by applying different techniques.

Mathematical morphology (MM) is a theory and technique for the analysis and processing of geometrical structures and it is mostly applied to digital images, even though it can be applied to many other spatial structures. MM is the foundation of morphological image processing; it consists of a set of operators that transform the images according to concepts such as size, shape, convexity, connectivity and geodesic distance. The basic morphological operators are *erosion* and *dilation*. Also, two other basic morphological operators widely used are *closing* and *opening*, which are a combination of the *erosion* and *dilation*. MM was originally developed for binary images, and was later extended to grayscale functions and images. [41]

The different post processing methods were applied in 3D to compensate for the fact that the MRI scans were studied in 2D and, hence, regularize the mask in 3D.

2.6.2.1. Binary morphological operators

Binary images may contain numerous imperfections such as noises and textures can be distorted when the binary regions are produced by a simple thresholding method, such as the one it was decided to use during this project (fixed thresholding method). Binary morphological operations try to remove these imperfections by accounting for the form and structure of the image [42].

These binary morphological operations consist of a collection of non-linear operations related to the shape of the features in an image. They rely on the relative ordering of pixel values not on their numerical values and, consequently, they are especially useful for working with binary images [42].

When applying morphological operations in an image there are two main structures who take part in these operations: first, the binary image that is being processed; second, the structuring element. The structuring element is a matrix that identifies the pixel in the image who is being processed and defines the neighborhood used in the processing of each pixel. It is positioned in all the pixels in the image and in each pixel studies the corresponding neighbourhood pixels in order to modify or not the value of the corresponding pixel.

The basic concepts to define the morphological operators are *dilation* and *erosion*.

- *Dilation* adds pixels to the boundaries of the objects in the image [43]. The idea is that for each pixel of A, center the transposed structuring element B_x^T. If the *dark* squares are considered 1's and the white *squares* are considered 0's, set the current pixel to 1 if at least one 1 pixel of B_x^T is contained in A, if not set it to 0.

  The mathematical expression of *dilation*: $A \oplus B = \{x \mid B_x^T \cap A \neq 0\}$

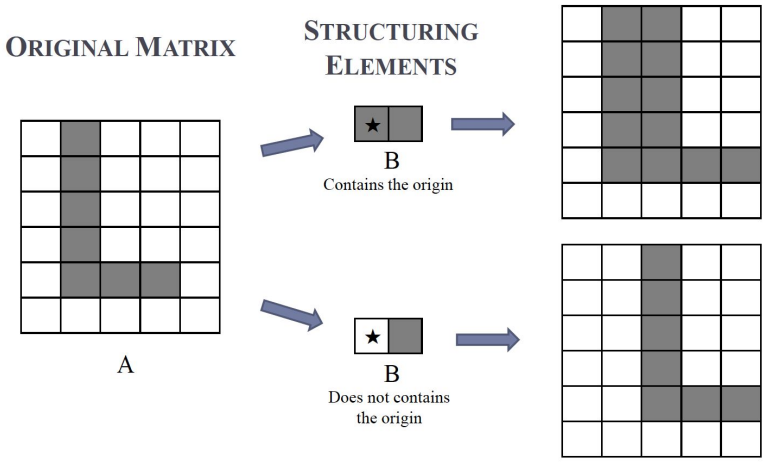

Figure 18 - Example of dilation [46].

- *Erosion* removes pixels to the boundaries of the objects in the image [43]. If the *dark* squares are considered 1's and the white *squares* are considered 0's, set the current pixel to 1 if all 1 pixel of B_x are contained in A, if not set it to 0.

  The mathematical expression of *erosion*: $A \ominus B = \{x \mid B_x \subseteq A\}$

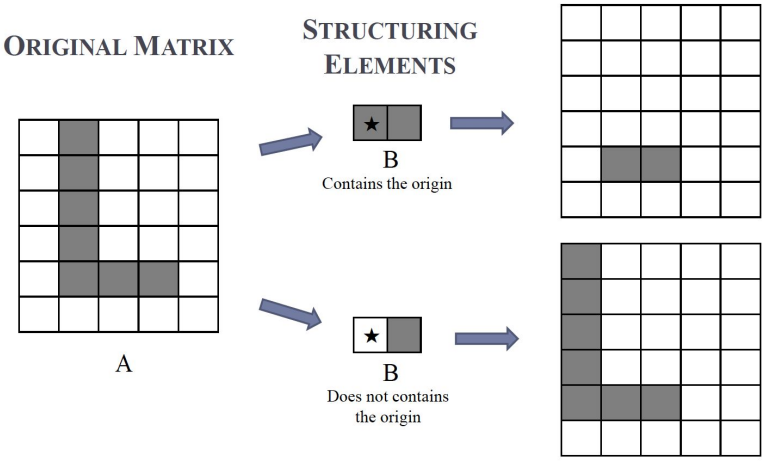

Figure 19 - Example of erosion [46].

Both *erosion* and *dilation* are invariant regarding translation, and they are not inverse to each other but dual.

$$not\ A \ominus B = not\ (A \oplus B^T)$$

$$not\ A \oplus B = not\ (A \ominus B^T)$$

Where B^T is the transposing of the structuring element symmetrical with respect to the origin.

Thanks to these two basic concepts of mathematical morphology, there are different operations that can be achieved combining both of them such as *region filling* or *contour detection*.

The *opening* operation corresponds to the succession of *erosion* and *dilation*; it removes the small objects from the foreground and places them in the background and detaches the weakly joined objects.

The mathematical expression of *opening*: $A \circ B = (A \ominus B) \oplus B$

The *closing* operation corresponds to the succession of *dilation* and *erosion*; it unifies the close objects removing the holes of the foreground and unifying them to the foreground and strengthens the weakly joined objects.

The mathematical expression of *closing*: $A \bullet B = (A \oplus B) \ominus B$

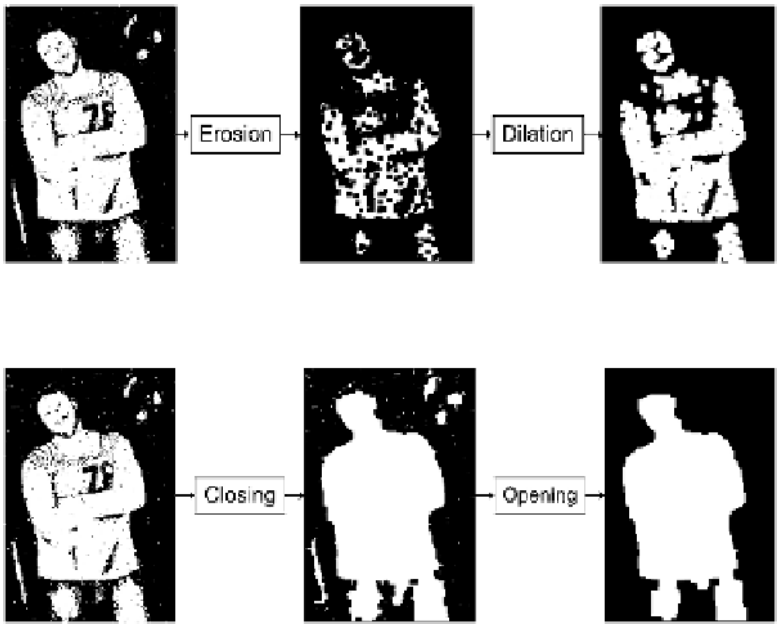

Figure 17 - Example of erosion and dilation (first row) and closing and opening (second row) [44].

In *Figure 17* the four morphological concepts remarked in this section can be observed. As indicated in the beginning of this section *erosion* removes pixels from the boundaries of the objects; it can be observed that the holes in the foreground get bigger and the small objects in the background disappear. On the other hand, *dilation* adds pixels from the boundaries of the objects making the foreground holes smaller.

In *Figure 17* both *closing* and *opening* can be observed as well. As indicated before, the *closing* operation unifies small objects removing holes in the foreground and the *opening* removes small objects from the foreground merging them into the background.

It is important to remark that all these operations depend on the shape of the *structuring element* who acts as a kernel.

2.6.2.2. Connected Component

Connected-component labeling (CCL) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. CCL is not to be confused with segmentation.

CCL is used in computer vision to detect connected regions in binary digital images, although color images and data with higher dimensionality can also be processed. Blob extraction is generally performed on the resulting binary image from a thresholding step, but it can be applicable to gray-scale and color images as well. Blobs may be counted, filtered, and tracked. [45]

$$X_k = (X_{k-1} \oplus B) \cap A \qquad for\ k\ =\ 1, 2, 3, \ldots$$

This algorithm is used to understand if the object points A are connected to each other. B is the *structuring element* for this mathematical morphology operation. The algorithm stops at step k if $X_k = X_{k-1}$, showing the different not connected components in the object A. In *Figure 18*, both the image previously applying CCL and afterwards applying CCL can be seen. Each color on the right image represents a different label.

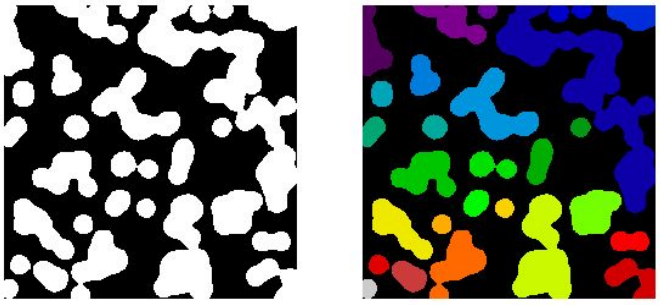

Figure 18 - Example of connected component labeling. Original image (left) and image after CCL (right).

As a post-processing in the resulting 3D stacks of images, the *Largest Connected Region* (LCR) was used. After computing the CCL, it consists of just preserving the biggest region in the 3D images and, hence, eliminating the small isolated segmentations. The code was obtained from M. Pierre-Henri Conze.

2.7 Evaluation metrics

For the evaluation of the results the dice coefficient is computed. The dice coefficient is really often used to evaluate quantitatively the performance of image segmentation methods. The dice score is a measure of how similar objects are; it is the size of the overlap of the two segmentations divided by the total size of the two objects.

In order to have a thorough evaluation, the dice in 3D is desired among the dice in 2D, since the final goal is to have a 3D segmentation of the CP and it is desired to study it this way.

Both networks are trained with 2D newborn MRI images and they predict 2D newborn MRI images as well. Even if the images used are 2D, this does not cause a problem since these images are obtained from 3D stacks.

In order to reconstruct the 2D predicted images to 3D predicted stacks it is necessary to apply some post processing methods. Firstly, it is necessary to select as a testset 30 different random subjects from the dHCP dataset and, afterwards, take all the slices belonging to these subjects. Secondly, indicate in each 2D image which subject and slice the image is. Finally, after the prediction of all these images, generate a 3D matrix of the same size as the original 3D stacks and attach each 2D predicted slice to its corresponding place.

3. Results

3.1 Segmentation of the CP (monolabel)

The main goal of the project, as indicated in *Section 1.5*, is to segment the cortical plate. In order to do that, two different architectures were tested: Pytorch UNet [11] and FetalCPSeg [5]. This section is subdivided in four main parts: the study of both networks separately, the evaluation of the fetus results and finally a conclusion of these different results.

3.1.1 Pytorch UNet - dHCP

3.1.1.1 Batch size study

In this section, the idea is to first study how the UNet performs with different batch sizes and conclude on which is the best. At the beginning of this project, due to software restrictions the batch size used was 13, but as soon as it became possible the study of the batch size was performed in order to improve the first results. In *Figure 19*, the different training performances can be observed.

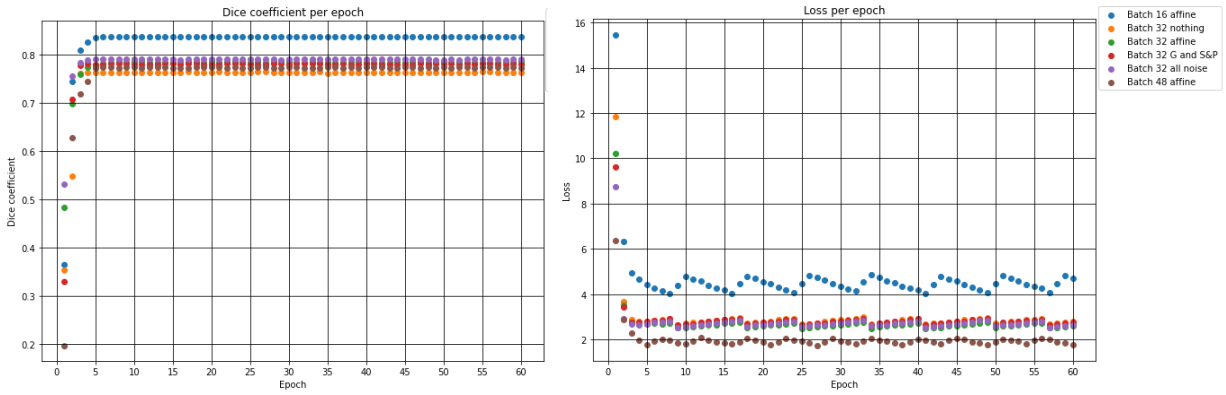

Figure 19 - Dice coefficient (left) and loss (right) for Pytorch UNet during the training.

For the dice coefficient values, it can be observed that the best performance is obtained with the batch size of 16 just using the affine transformations described in *subsection 2.6.1*. However, if the loss is observed, the batch size of 16 is the one that performs the worst. Finding this equilibrium between loss and dice coefficient the best performance is obtained when using batch size of 32 and using all data augmentation: the affine transformations (translation and rotation), and injection noise (gaussian, salt and pepper and speckle noise). Apart from considering the values of these two metrics, it is really important to consider an invariance of the network with respect to these different deformations. This makes the network robust to both translation and rotation as well as the three nose injections, which are the most common noises encountered in the MRI 3D stacks.

3.1.1.2 Dice coefficient evaluation

After studying the performance of the network during the training it is important to do a quantitative study of the predictions. In order to do that, three different studies take place. Firstly, the study of the dice coefficient per each slice of the 3D stack is performed for 30 3D random dHCP subjects. Afterwards, the 3D dice coefficient is computed for all these 30 3D dHCP subjects. Finally, the study of the different morphomath post processing techniques takes place.

In *Figure 20*, the dice coefficient per slice for the 30 dHCP subjects can be observed. In the edges of the images there are two important things to remark. On one hand, it can be observed that some of the edges obtain a dice coefficient of 1. This happens due to the fact that the ground truth of the CP in the extreme slices is null and, hence, the prediction is null as well, which causes the dice coefficient to be 1. On the other hand, if we do not take into consideration these values of the dice coefficient, in general, the value of the dice score is reduced in the edges. The fact that the prediction is less accurate in the extremes is a really frequent fact when predicting MRIs.

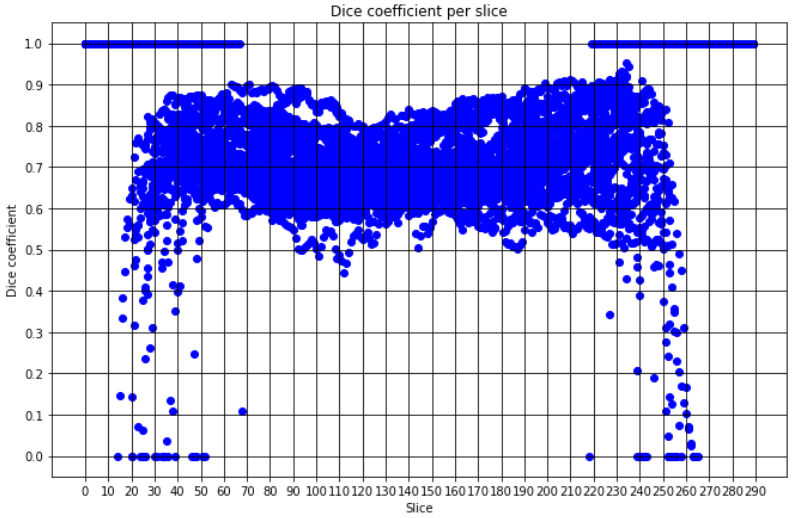

Figure 20 - Dice coefficient per slice of 3D test dHCP images for Pytorch UNet.

Afterwards, as indicated previously, the computation of the 3D dice coefficient takes place. In *Figure 21* this can be observed. Furthermore, in the image, some 3D binary morphological postprocessing is considered as well. For binary morphological postprocessing the basic operations have been considered, as explained in *section 2.6.2.1*: dilation, erosion, opening and closing. Additionally, the *Largest Connected Region* (LCR) explained in *section 2.6.2.2* has been considered as well.

In *Figure 21*, as indicated above, the 3D dice coefficient for the 30 dHCP subjects can be observed. After applying the mathematical morphology the 3D dice coefficient was calculated. As can be seen in *Figure 21*, the erosion really degradates the 3D prediction, as well as the opening. With respect to the dilation it sometimes improves the prediction but it also degradates it a lot in other cases. Finally, both closing and LCR always improves the 3D mask prediction.

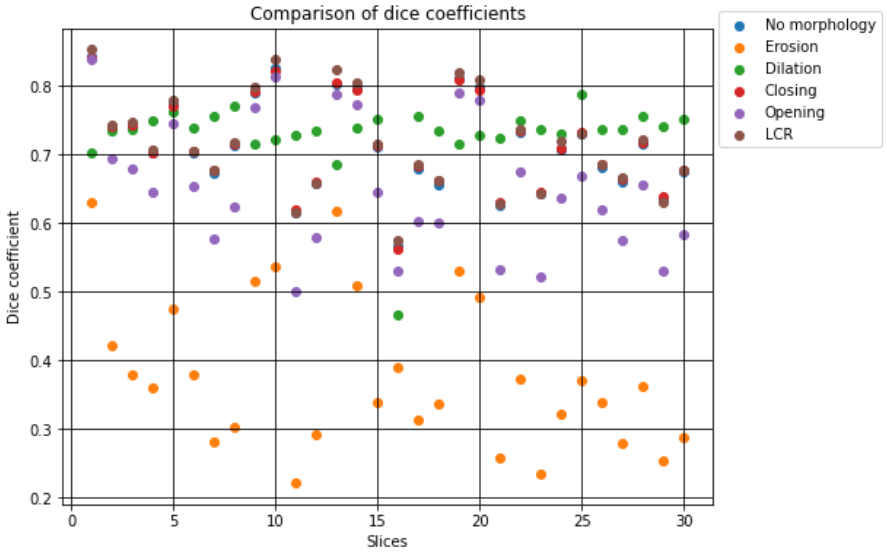

Figure 21 - 3D dice coefficient for 30 dHCP subjects with mathematical morphology post processing for Pytorch UNet.

Since closing and LCR are the two post processing operations that improve better the results, it was considered to combine them in order to obtain even a better improvement. However, it was observed that there is no significant improvement when combining.

3.1.2 FetalCPSeg - dHCP

For the study of the performance of the FetalCPSeg architecture different modifications were considered. Firstly, it is important to remark that in this network three important components take place. On one hand, there is the backbone network (BBN), which has a residual UNet architecture. On the other hand, in the second part of the architecture, we find the deep supervision modules (DSM) [24] and the attentive modules (AM).

The two most important studies are based on the DSM as well as on the AM. The goal of the DSM is to use multi-scale feature learning inspired by deeply supervised networks [46]. The goal of the AM is to increase the richness of the multi-scale maps learned by the backbone network.

In order to perform these studies, this part has been splitted in several parts. The first part is focused on studying the performance with just the backbone network. The following part is focused on studying how the network works without the AM, meaning that the focus of the study is set on the DSM. The next part consists of studying how the network performs with the AM. Finally, the last part consists of plugging inside the code the Pytorch UNet used for the first part of this project.

The following experiments have been trained on the 40th youngest subjects of the dHCP dataset as done in *section 3.1.1*.

3.1.2.1 Backbone Network

The backbone network (BBN) consists of a residual UNet structure as explained in *section 2.2.2*. In *Figure 22* the performance of the network during the training can be observed. It can be said that both the dice coefficient and the loss converge at really good values, the dice score is around 0.85 and the loss is really close to 0.

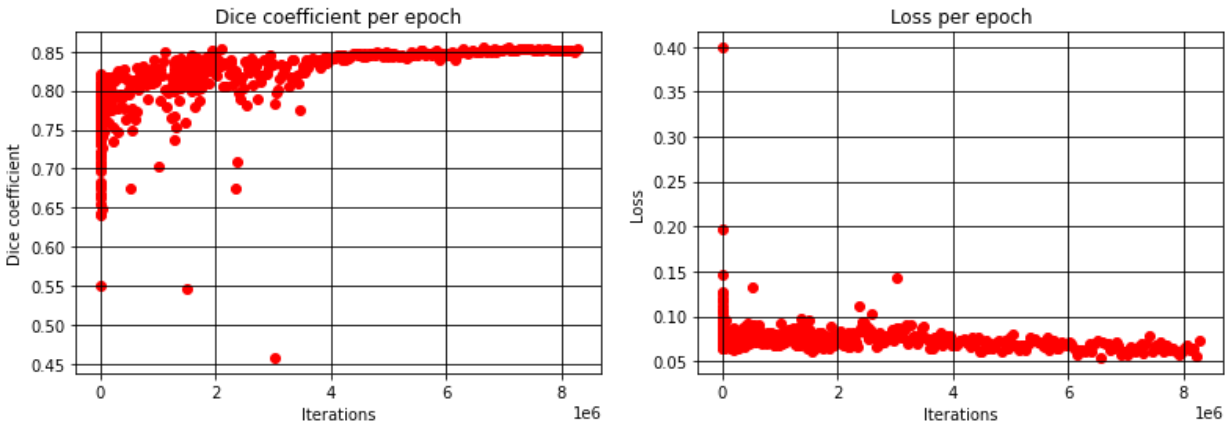

Figure 22 - Dice coefficient (left) and loss (right) during the training for the BBN study of the FetalCPSeg architecture.

For the study of the performance of this residual UNet, the dice coefficient of 30 dHCP subjects has been computed. The dice coefficient per slice for the 30 subjects can be observed in *Figure 23*. It can be observed that generally the dice coefficient in the whole images is really good but, furthermore, the extreme slices are overall good predicted as well.

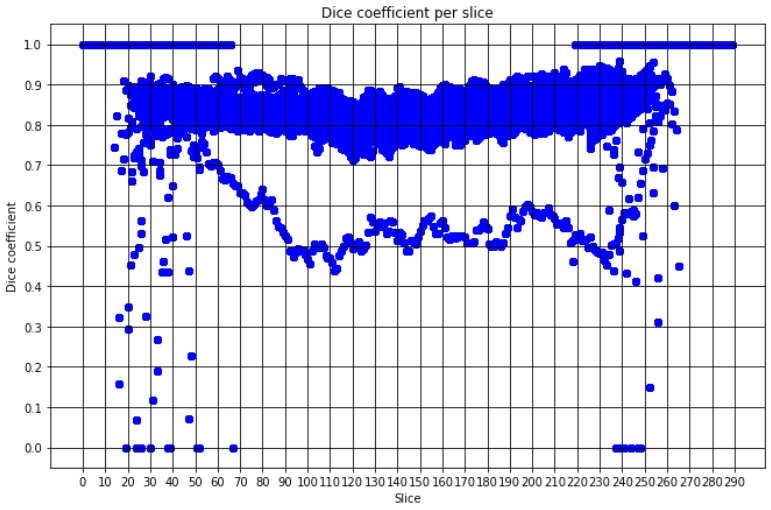

Figure 23 - Dice coefficient per slice of 3D test dHCP images for BBN of FetalCPSeg.

In *Figure 24*, the value of the dice coefficient for each 30 subjects can be observed for both before and after post processing mathematical morphology. As expected, the same as in the results of Pytorch UNet happened. The best results are achieved with the method of LCR. Moreover, the other mathematical post processings operations perform really similar to *Figure 21* for the Pytorch UNet results, meaning that, for example, erosion is the one who degrades the most results.

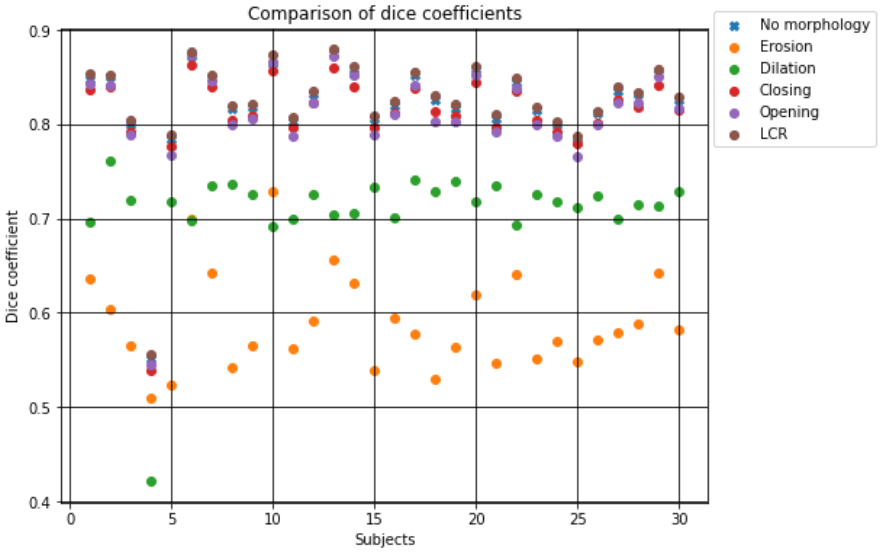

Figure 24 - 3D dice coefficient for 30 dHCP subjects with mathematical morphology post processing for BBN of FetalCPSeg.

3.1.2.2 Deep Supervision Modules

In *Figure 25* the dice coefficient and the loss per epochs during the training of the network can be observed. It can be seen that it converges after around 4.5×10^6 iterations to a dice coefficient of 0.8. Afterwards the model starts to degrade.

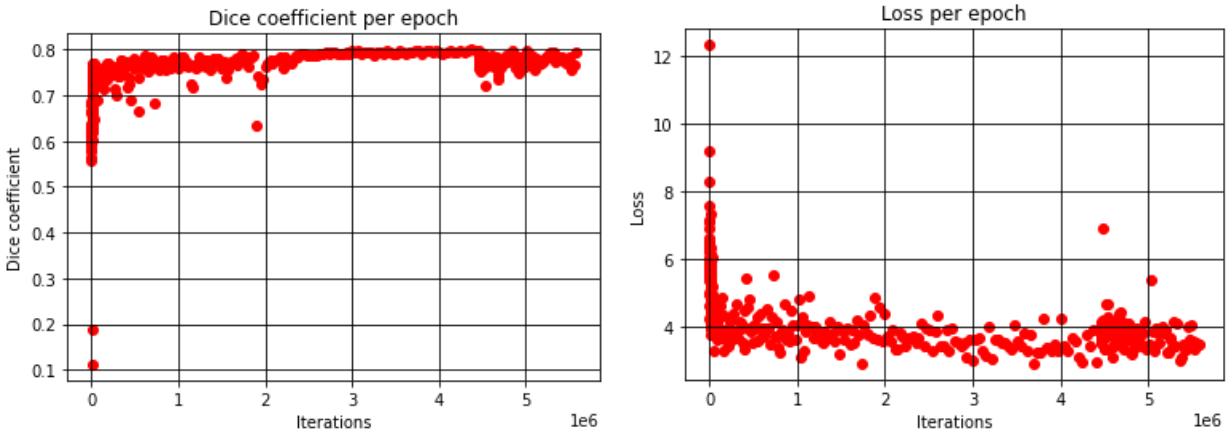

Figure 25 - Dice coefficient (left) and loss (right) during the training for the DSM study of the FetalCPSeg architecture.

As done previously, the dice coefficient per slice for the 30 subjects can be observed in *Figure 26*. It can be observed that generally the dice coefficient for all the slices is really good but some of the extreme slices are badly predicted since there appear more 0 values than for the BBN, which indicates that even though there is no CP to be predicted something is being considered as CP and some pixels are being mislabeled as CP.

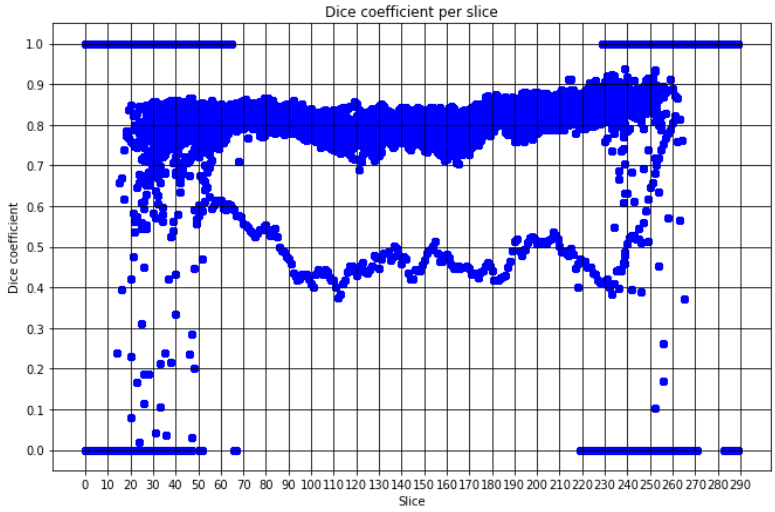

Figure 26 - Dice coefficient per slice of 3D test dHCP images for DSM study of the FetalCPSeg architecture.

In *Figure 27*, the value of the dice coefficient for each 30 subjects can be observed for both before and after post processing mathematical morphology. As expected, the best results are achieved with the method of LCR, as well as the worst performances are obtained with erosion and dilation.

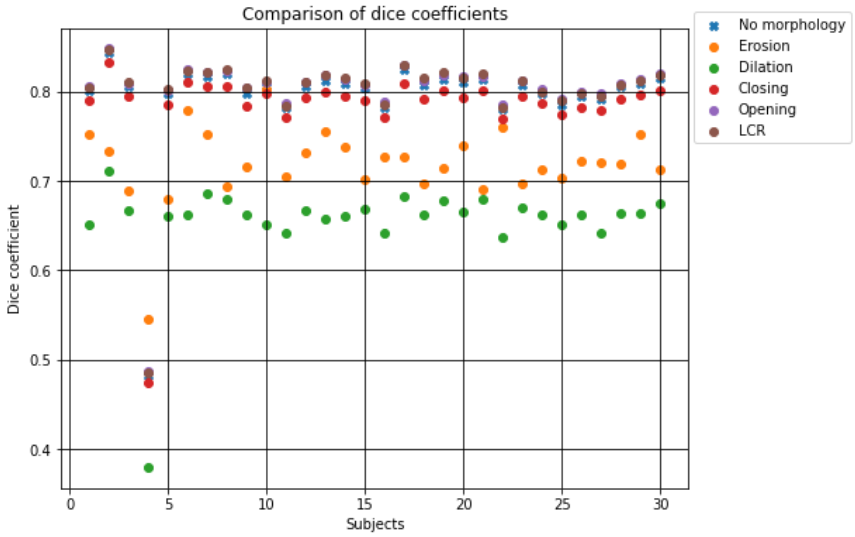

Figure 27 - 3D dice coefficient for 30 dHCP subjects with mathematical morphology post processing for DSM of FetalCPSeg.

3.1.2.3 Attentive Modules

In *Figure 28,* the dice coefficient and the loss per epochs during the training of the network can be observed. It can be seen that it converges after around 4.5×10^6 iterations to a dice coefficient of 0.8. Afterwards the model starts to degrade.

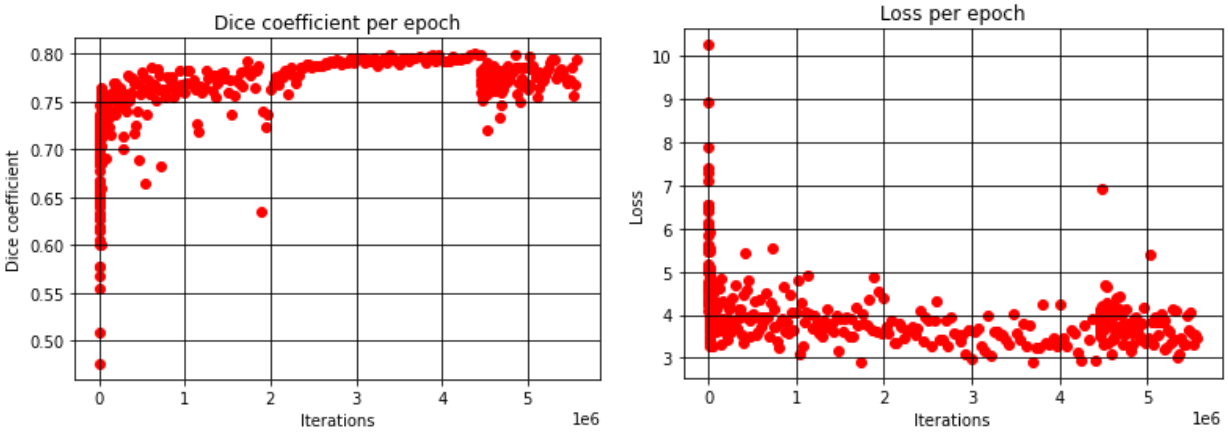

Figure 28 - Dice coefficient (left) and loss (right) during the training for the AM study of the FetalCPSeg architecture.

The results of this section will be seen and studied in the following *subsection 3.1.2.3.1 Study of Attentive Modules.*

3.1.2.3.1 Study of Attentive Modules

In this subsection, it is aimed to compare the same network architecture with and without attentive modules. In order to do that, the results of the network with and without attentive modules are put together and compared in order to study the two performances.

Firstly, it is important to refresh what are the attentive modules and what is their goal. As indicated in *section 2.2.2.*, the idea of the attentive modules is to increase the richness of the information of the multi-scale feature maps already learned from the network. It takes the feature maps generated from each stage of the backbone network and output a refined attentive feature map of the same size.

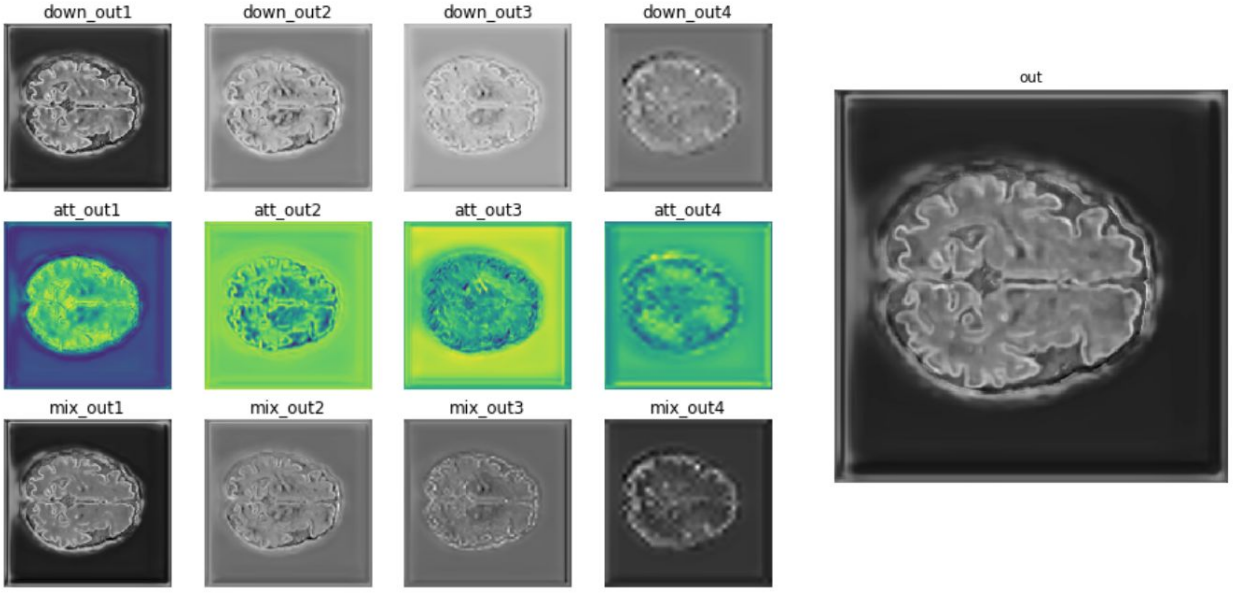

Figure 29 - Feature maps before (first row) and after (third row) the attention modules with the respective attentive maps (second row) per each stage.

In *Figure 29*, it can be observed the influence of the feature maps. The first row (*down_outX)* shows the feature maps in each four stages before entering in the attention module. The second row (*att_outX*) shows the attentive maps obtained for each stage inside the attention module. The third row (*mix_outX*) shows the feature maps obtained outside the attention modules. Finally, the image in the right shows the output of the whole network.

As it can be observed in the image, the differences between the feature maps before and after the attention modules do not have significant differences and the refinement is not that clear. Because of that, it could be said that the addition of attention modules is not that relevant. However, it is important to make empirical tests to support or not this hypothesis. The more relevant way to study the performance is to study a 3D dice coefficient.

First of all, an important study is to check the dice per each slice since it is important to check if the dice coefficient really depends on the location of the slice. The comparison between the dice scores per slice for the 30 dHCP subjects of the two architectures with and without attentive modules can be observed in *Figure 30*.

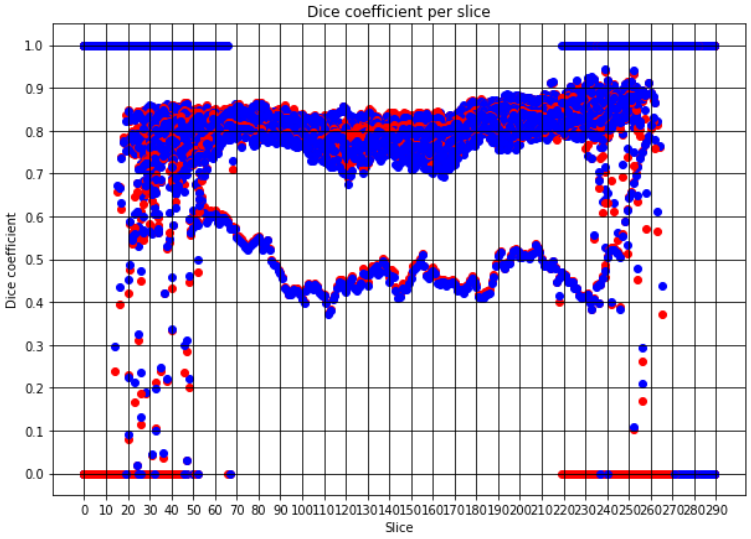

Figure 30 - Dice coefficient per slice of 3D test dHCP images for the AM study of FetalCPSeg architecture with AM (blue) versus without AM (red).

In *Figure 30* it can be observed that generally the attentive module does not improve the predictions. However, in the first slices, which correspond to the lowest part of the brain, the performance improves a lot. In these slices appear things like the cerebellum or the brainstem and they generate more difficulties to predict the CP correctly. In these slices, it can be observed that the attentive module really helps to improve the predictions and, hence, it may improve the dice coefficient in 3D. However, if we observe all the slices besides from the edge ones, the best dice coefficient generally is achieved by the architecture without AM.

It is important to remark that in *Figure 30* it is proved that the performance of the network does not depend at all on the date of the MRI acquisition but just on the slice or on the quality of the image.

In *Figure 31* the dice coefficient in 3D of the previous studied subjects is plotted. It can be observed that both with and without the attentive module have similar performances. In some cases, the attentive module improves the dice in 3D as a consequence of the better performance in the lower parts of the brain. In other cases, it degreadates the result in the center slices in a way that the performance without an attentive module is better. In addition to that, morphological mathematical post processing is applied to the 3D images. The computation of the LCR shows an improvement in both cases, with and without attentive module.

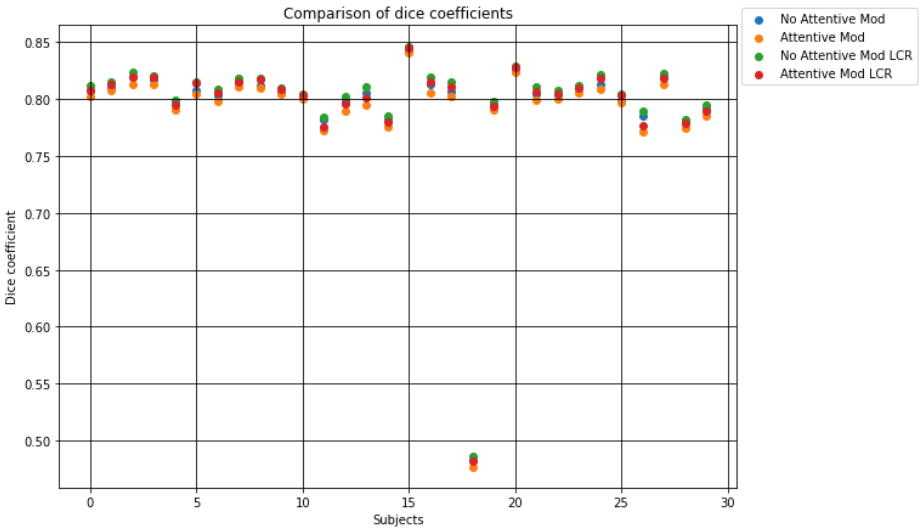

Figure 31 - 3D dice coefficient for 30 dHCP subjects with LCR post processing for the AM study of FetalCPSeg architecture.

It can be observed in the figure that the better performance is found in the 15th subject and the worse performance is found in the 18th subject. In order to find the reason why this happens, both images are plotted in the next *figures* (*32* and *33*). The prediction is plotted in blue and the ground truth is plotted in red. Consequently, when both prediction and ground truth are superposed the color obtained is purple, which indicates the places in which the prediction corresponds to the ground truth.

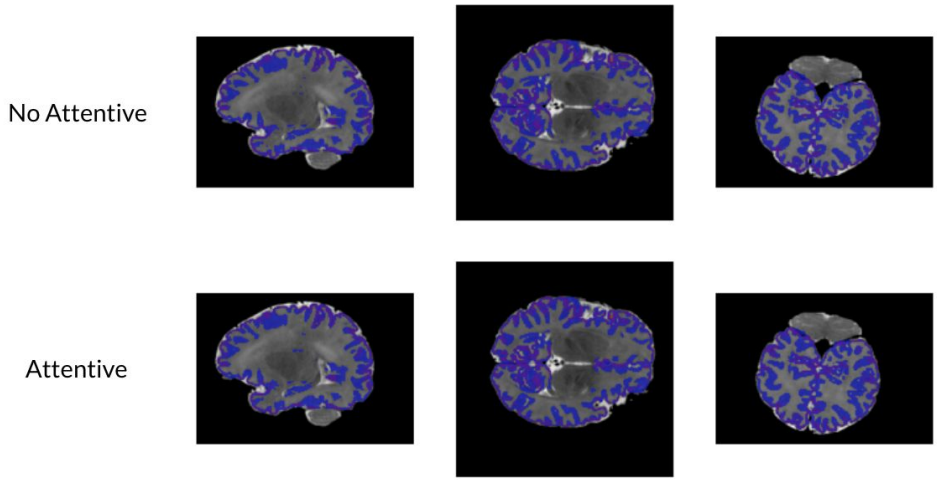

Figure 32 - Worst case. MRI date acquisition: 42 weeks.
Dice coefficient 3D (without attentive module) = 0.477
Dice coefficient 3D (with attentive module) = 0.483

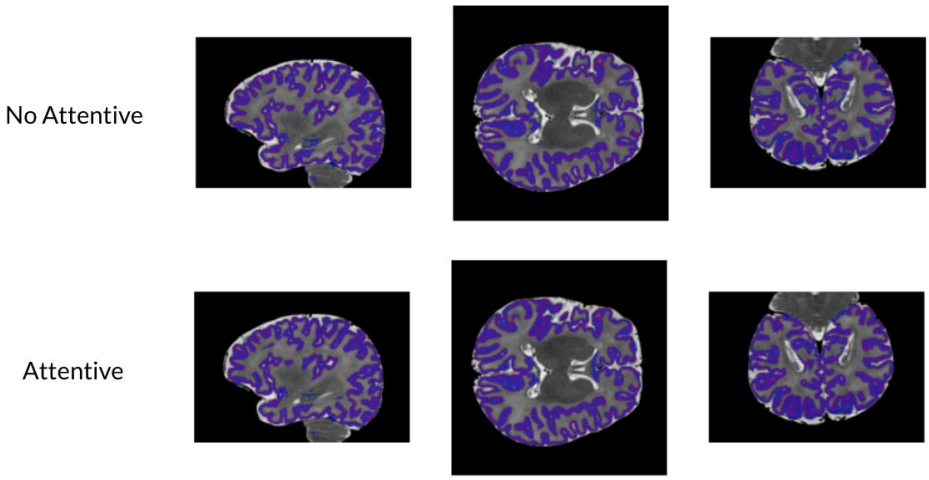

Figure 33 - Best case. MRI date acquisition: 44 weeks.
Dice coefficient 3D (without attentive module) = 0.841
Dice coefficient 3D (with attentive module) = 0.844

In these two figures above, the worst case and best case found in *Figure 31* can be observed. For the worst case with and without attentive module the dice is around 0.5 and for the best case for both with and without attentive module the dice is around 0.85. As already explained, the prediction is plotted in *blue* and the ground truth is plotted in *red*. If both prediction and ground truth correspond to each other the mask plotted should be *purple*. For the worst case, the mask plotted is mostly *blue*, which tells us that the prediction does not correspond with the ground truth and the MRI image is over segmented, meaning that the prediction is much bigger than the ground truth. For the best case, the mask plotted is overall *purple*, which indicates that both prediction (*blue*) and ground truth (*red*) are superposed and correspond to each other. Consequently, it can be concluded that the predictions do not depend on the MRI date acquisition since for the worst case it is 42 weeks and for the best case is 44 weeks. This is coherent with the conclusion obtained in *Figure 30* as it was observed as well that it did not depend on the age but on the slice or quality of the image as well.

3.1.2.4 Pytorch UNet plugged into the code

In this section the idea is to study the performance of the architecture when plugging the Pytorch UNet inside the FetalCPSeg. In *Figure 34* the performance of this architecture can be observed. The dice coefficient converges to 0.8, which is pretty good; however, the loss converges to ~4 which is not good.

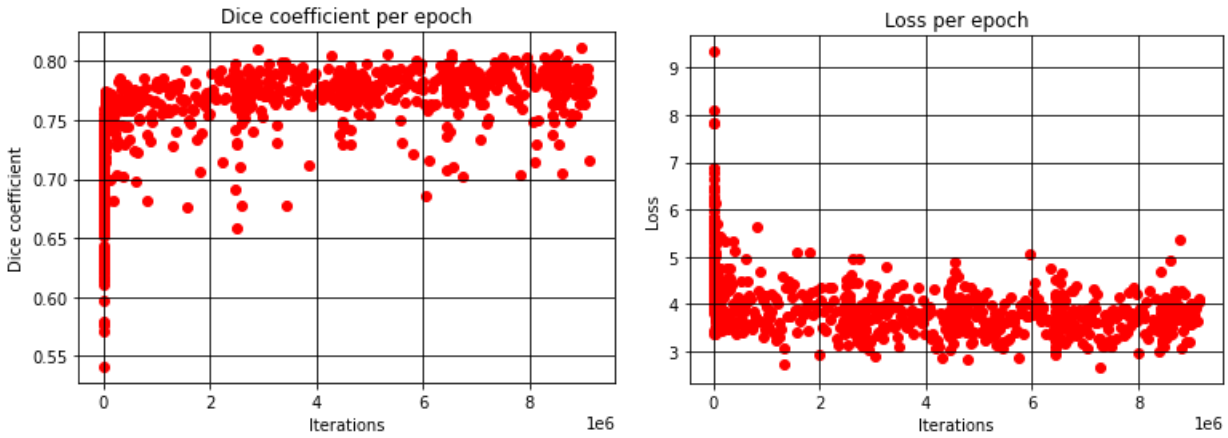

Figure 34 - Dice coefficient (left) and loss (right) during the training of Pytorch UNet plugged into FetalCPSeg.

In *Figure 35*, the dice coefficient per slice of the 30 dHCP subjects can be observed. Generally, the performance for all the subjects is not bad. In the edge slices it can be seen that, mostly in the lowest part of the brain, the performance is worse. However, it can be also found some values of dice coefficient equal to 1, which means that in the slices that were not any CP pixels, the prediction is null hence, the dice score is 1.

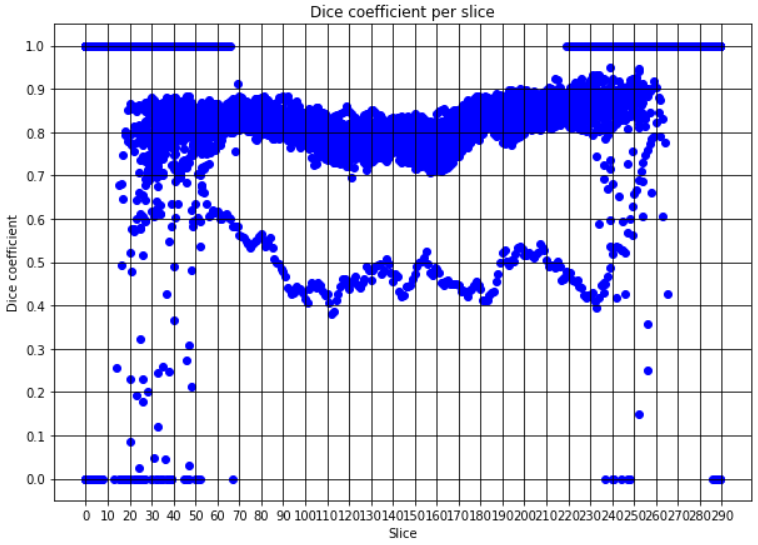

Figure 35 - Dice coefficient per slice of 3D test dHCP images for the Basic UNet plugged into the FetalCPSeg architecture.

In *Figure 35*, the value of the dice coefficient for each 30 subjects can be observed for both before and after post processing mathematical morphology. As expected, the best results are achieved with the method of LCR, as well as the worst performances are obtained with erosion and dilation.

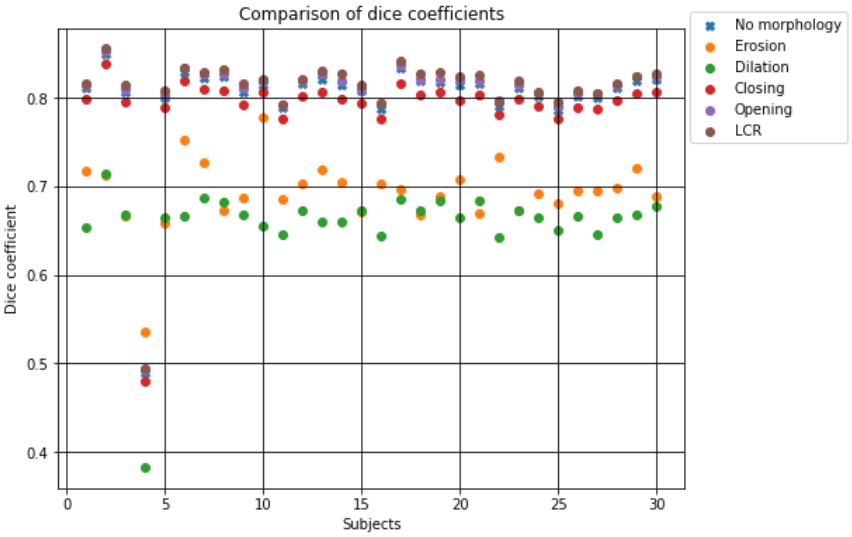

Figure 36 - 3D dice coefficient for 30 dHCP subjects with mathematical morphology post processing for the Basic UNet plugged into the FetalCPSeg architecture.

Now, the important study left is to compare all of the performances depending on which elements of the FetalCPSeg architecture are used.

3.1.2.5 Study of the relevance of the FetalCPSeg architecture parts

In this subsection the main idea is to compare the different performances depending on which part of the architecture of the FetalCPSeg network is being used.

For performing this study, firstly, the dice coefficient per slice is plotted in *Figure 37*. It can be easily observed that the better performance is obtained when just the backbone network is obtained, for all the subjects and independently of the location of the slices. However, it is important to remark that for all four implementations, generally, the edge slices do have a good performance, which is something that usually degrades the dice coefficient in 3D for the bad results.

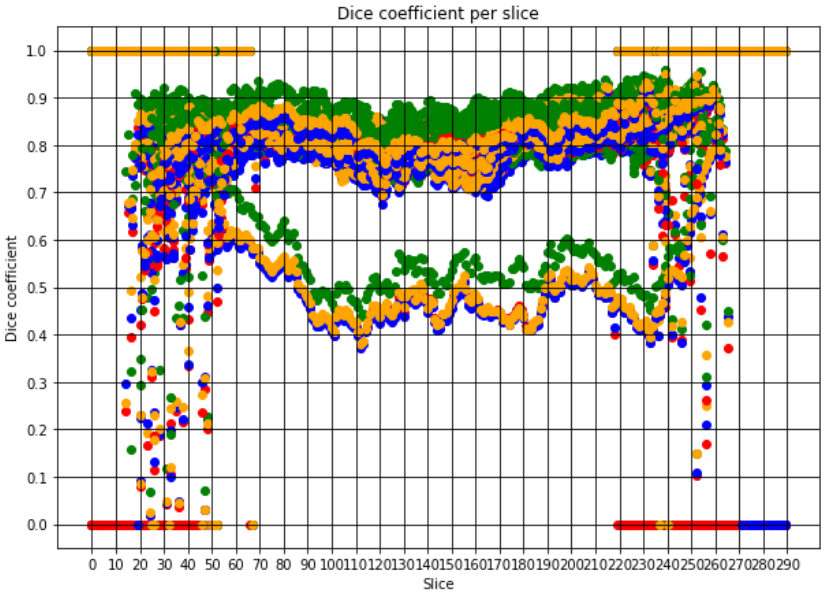

Figure 37 - Dice coefficient per slice of 3D test dHCP images for the different studies performed with the FetalCPSeg architecture. Backbone network (green) vs without attentive module (red) vs with attentive module (blue) vs basic UNet plugged into FetalCPSeg architecture (orange).

To continue with the study of the different architectures, the dice coefficient in 3D is plotted in *Figure 38*, without any post processing applied since the idea is to compare the performance of each architecture transparently. It can be observed, as expected since the commentary of the *Figure 37*, that the better performance is obtained just when using the backbone network (*blue*). When the deep supervision modules are used (*orange*) the performance of the network degrades a little bit, and, when the attentive modules are added (*green*) it degrades even more. However, something interesting to point out is that, when using all the modules within the FetalCPSeg architecture, the performance is better when working with the basic UNet instead of the backbone network which turns out to be a residual UNet.

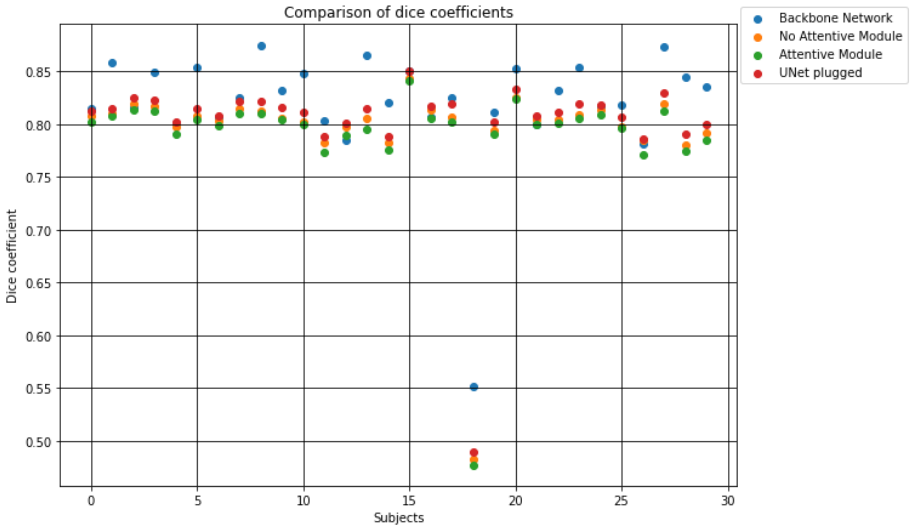

Figure 38 - 3D dice coefficient for 30 dHCP subjects for the different studies performed with the FetalCPSeg architecture.

3.1.3 Conclusion for the CP monolabel segmentation

After studying the architecture and the performance with and without the different parts of the FetalCPSeg network, it was concluded that the best performance was obtained with just using the backbone network which, as explained before, is a residual UNet. After concluding that, it is important to compare this network with the Pytorch UNet studied in the *section 3.1.1*. This network is based on a basic UNet architecture without any residual links.

The first step to study these two UNets is to check the dice coefficient per slice for the 30 3D dHCP subjects used as a training set. The plot of these dice values can be observed in *Figure 39*. It can be fastly observed that the results obtained with the residual UNet are better overall. In addition, if the edges of the images are observed closely there is a significant improvement from the basic UNet to the residual UNet. The 2D slices from 20 to 50 and from 240 to 260 the dice scores are much better with the residual UNet and, it can be already said that the 3D dice coefficient would improve significantly from the basic UNet to the residual UNet.

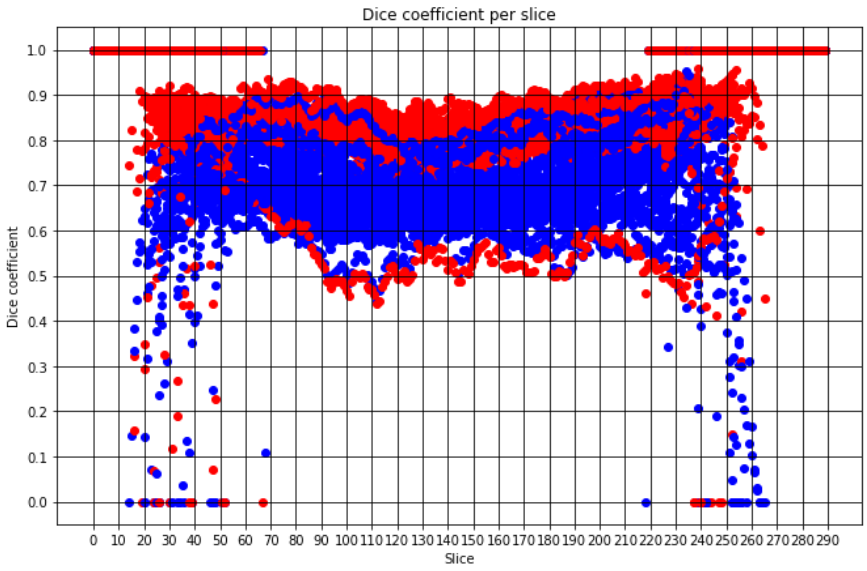

Figure 39 - Dice coefficient per slice of 3D test dHCP images of the Pytorch UNet (blue) versus the FetalCPSeg UNet (red).

In *Figure 40* the 3D dice coefficient for the 30 dHCP subjects is plotted. As expected, the value of the 3D dice scores improves a lot when predicting with the residual UNet with respect to the basic Pytorch UNet. For both cases, when applying the LCR post processing the result tends to improve, although this improvement is not really significant for some of the dHCP subjects.

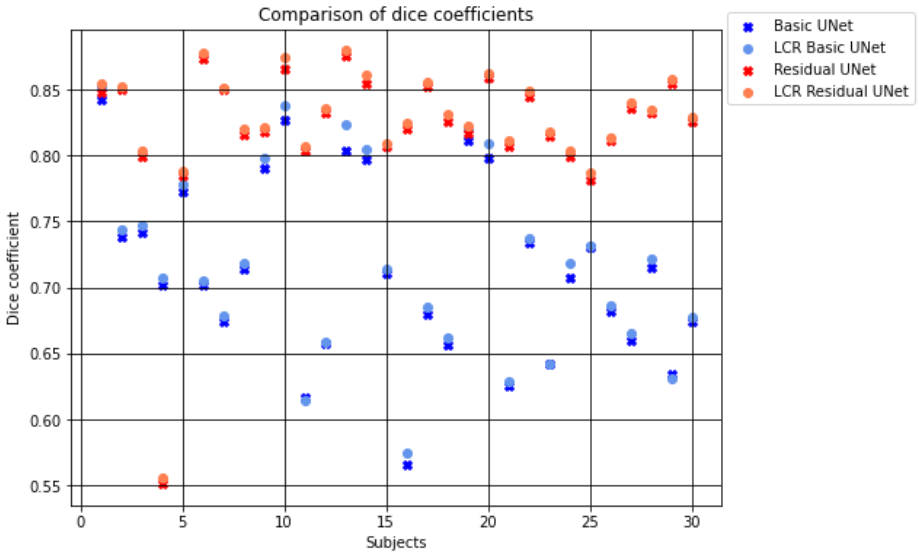

Figure 40 - 3D dice coefficient for 30 dHCP subjects with LCR post processing for the comparison of the basic UNet and the residual UNet (backbone network).

3.1.3.1 Fetus results

In this subsection, a fetal result is going to be compared between both the basic UNet and the residual UNet. This can be observed in *Figure 41*. Both predictions are good considering that the training has been performed with newborns MRIs and this is a fetal MRI as well as considering that most likely the quality of the image is worse.

If both results are compared qualitatively, it can be observed that the CP prediction of the residual UNet is more robust than the one obtained with the basic UNet. However, it can be observed that some of the parts predicted for the residual UNet may not belong to the CP. This can not be concluded thoroughly since there is no ground truth to compare both predictions. As the dHCP results were better when the training and prediction was performed with the residual UNet, it could be said that the prediction of the fetuses as well. Nonetheless, this should be reviewed and evaluated by an expert.

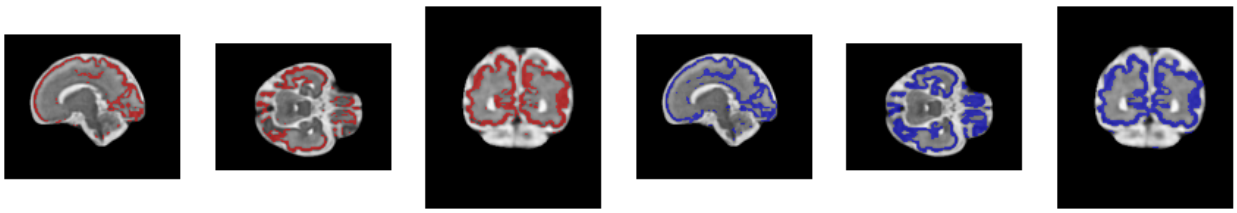

Figure 41 - Predictions of 3D fetus images with Pytorch UNet (red) and for residual UNet from FetalCPSeg architecture (blue). Subject marsFet021 (MRI date 32).

3.2 Multilabel - Pythorch UNet

The idea of this section is to analyze and study if the multilabel classification in the Pytorch UNet helps improve the results of the CP labelization. In order to test that, it was necessary to use a multilabel mask, which was generated from the different *drawem9* labels. Besides the CP *drawem9* label, two other labels were used for this multilabel classification: the cerebrospinal fluid together with the ventricle and the white matter. These other labels were selected both for their relevance within the anatomy and for the amount of images they can provide since single label training will be performed as well for these masks; the cerebrospinal fluid together with the ventricle provided 6607 images and the white matter 5147. The images for the CP obtained from the *drawem9* label were 5378.

The dice coefficient evaluation of the CP, even if the ground truth used for training is obtained from the *drawem9* label, the comparison is done with the CP grountruth obtained from the image (3) since, as indicated in *section 1.4.1*, the CP obtained from this image is more anatomically accurate than the one obtained from the mask *drawem9* from the image (4).

This section is split into three parts: the first part is focused on studying the performance of these three labels separately; the second part is focused on studying the performance of the multilabel classification and finally, the third part is focused on comparing the multilabel results with the single label results.

In this section, every training has been made with batch size of 32 but without noise as data augmentation, just affine transformations. The reason for that is that the noise injection, when working with segments such as white matter or cerebrospinal fluid, changes significantly the image histogram distribution. This causes a bad performance from the network in both multilabel segmentation and in monolabel segmentation for the label 1 (cerebrospinal fluid and ventricles) and for the label 3 (white matter) since it confuses the classifier breaking the homogeneity of the area. Consequently, it was concluded the noise injection had to be removed. However, it was necessary to make a fetus study for each label without noise to test it had a good performance even if the network was no longer invariant to this kind of deformations.

3.2.1 Labels separately

The first part of this section, as already indicated, consists of testing each label separately. Each label was trained under the same conditions as the section 3.1.1 but without noise injection.

In *Figure 42* the dice coefficient and the loss are plotted for the three different trainings of the Pytorch UNet. As indicated above, the Label 1-5 corresponds to the cerebrospinal fluid (CSF) and the ventricles (V); the Label 2 corresponds to the cortical plate (CP) and the Label 3 corresponds to the white matter (WM). All these masks are obtained from the image *drawem9*. It can be observed that the three separate training sessions achieve really good dice coefficients and loss values.

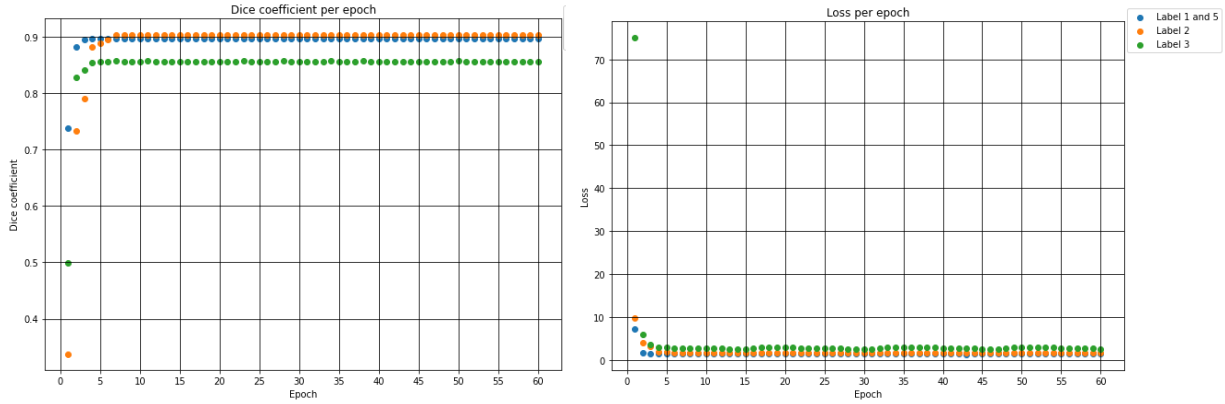

Figure 42 - Dice coefficient (left) and loss (right) for Pytorch UNet during the multiple training sessions.

In *Figure 43*, the 3D dice coefficient for the 30 dHCP subjects can be observed. It can be observed that all 30 dHCP subjects obtain really good values of 3D dice coefficient independently for the three labels. Several post processing operations have been applied and it was observed that for each label a different operation gave the best performance.

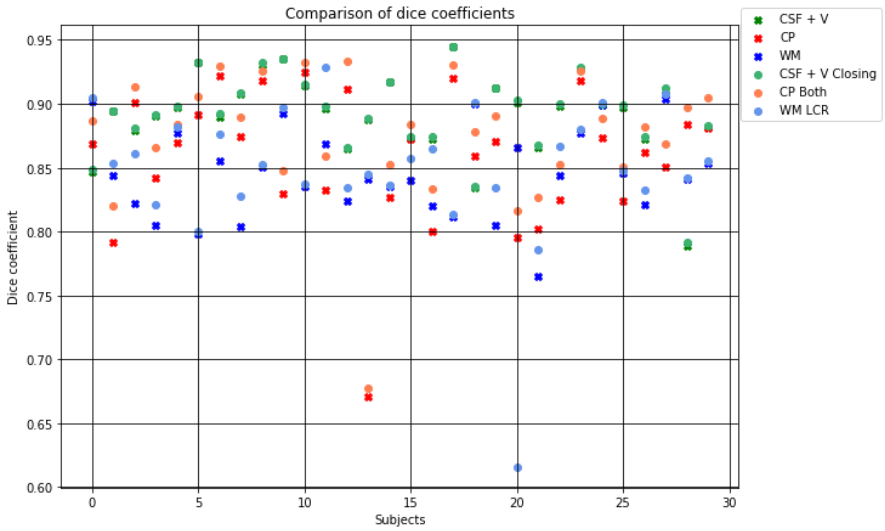

Figure 43 - Comparison of 3D dice coefficient for 30 dHCP subjects for the three different labels.

For the Label 1-5 (CSF + V) the better performance was reached with the binary post processing closing operation. This makes sense since the LCR would eliminate the ventricles and hence the dice coefficient will degrade considerably. For the Label 2 (CP) the best performance was obtained with applying first the LCR and afterwards the closing. This makes sense as well since taking the LCR will eliminate random small points that could appear as CP and applying the closing will make the prediction of the CP more wide and robust. Finally, for the WM the better performance was obtained applying just the LCR, which will eliminate sporadic labeled points and increase the dice score.

3.2.2 Labels together

In this section the different labels studied in the previous *section 3.2.1* are studied together, as a multilabel problem. As a consequence, during the training of this multilabel segmentation problem is done without noise injection because some of the labels do not benefit from using synthetic noise but the contrary; even if the label being studied is the CP.

Finally, the important thing to remark here is the fact that we were working with an imbalanced dataset, meaning that the number of data points available for multiple classes is different. This causes that, for example, the optimization criteria or performance measures may not work effectively.

There are two typical approaches to deal with an imbalanced dataset: over-sampling and class weighting. When using the over-sampling method we are actually working with data augmentation. Data augmentation is the process of changing samples and adding them into the set, enriching the diversity of these samples. It is related to over-sampling since we can both enrich and increase the size of a class; this increasing aspect is called data over-sampling and produces the same effect as working with a balanced dataset [52].

Since increasing the amount of samples of a class by two and assigning a two weight to the class is equivalent, it can be said that data over-sampling and class weighting are equivalent. However, the weighting is better from the storage and computational point of view since it avoids working with a larger data-set.

The idea of class weighting is generally done by over-penalizing the miss-classification of a C class compared to other classes. In this case, the modification was done in the cross entropy loss function. The main idea is to apply weights on the cross entropy loss function for the different labels. The first mathematical expression is the cross entropy function without weighting and the second mathematical expression is the loss function with class weighting.

$$H_y(y') = -\sum_i \sum_{k=1}^{K} y_{ik} log(y'_{ik}) \qquad \rightarrow \qquad H_y(y') = -\sum_i \sum_{k=1}^{K} w_k y_{ik} log(y'_{ik})$$

The weights assigned for our training was, as explained, directly related to the amount of samples of each label. The values of the weights can be observed on *Table 1*. The total number of pixels was 684069400.

Class	**Number of pixels**	**Relation**	**Weight**
Label 0	Num_0: 609996799	Num_total/Num_0	1.1214311306574578
Label 15	Num_15: 17672697	Num_total/Num_15	38.707696963287496
Label 2	Num_2: 21663251	Num_total/Num_2	31.57741190368888
Label 3	Num_3: 34736653	Num_total/Num_3	19.693014177272634

Table 1 - Weight values used for weighting the cross entropy loss function.

Basically, what this does is to tell the model that miss-classifying class one member from class 0 is as punishable as miss-classifying ~31 members from the class 2.

Firstly, the dice coefficient per slice is plotted for the 30 dHCP subjects in *Figure 44*. It can be observed that the performance of the CP is generally bad for all the slices, independently if they are on the edge of the image or not, but worse if the slice is in the middle of the image which is something singular. The performance of the cerebrospinal fluid is not good either; however, it performs better if it is in the middle of the image, which is usually what tends to happen in medical images. Regarding the white matter label, the performance is generally good but it performs better in the edge slices as well as the CP, which, as already said, is something peculiar.

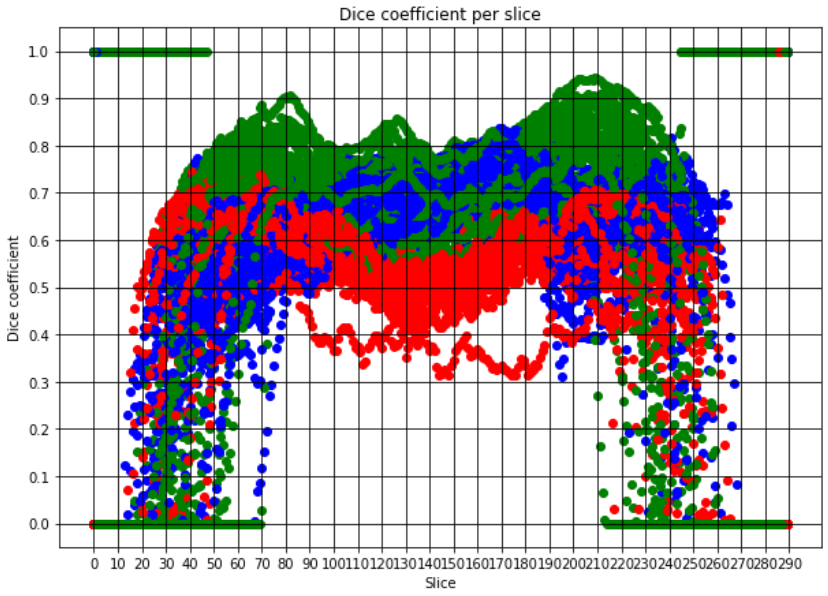

Figure 44 - Dice coefficient per slice of 3D test dHCP images of the multilabel classification: cerebrospinal fluid with ventricles (blue), cortical plate (red) and white matter (green).

In *Figure 45* the 3D dice coefficient for the different labels can be observed. It can be seen that for the white matter label the result is quite good; however, for both the cerebrospinal fluid with ventricles and for the cortical plate the results are bad, even if we apply some morphological post processing. If the postprocessing methods used are observed, it can be seen that they are not the same as used for separate labeling. These methods were chosen after comparing all post processing techniques and they were selected as the ones that performed better. A clear example that the multi labeling does not perform good is the fact that, for example, for the cerebrospinal fluid with ventricles label, the post processing method that performs the best is LCR, which means that the ventricles would eliminated if the prediction was accurate and, hence, degrade the dice coefficient. The fact that this is the best option for the post processing, makes it clear that the prediction is not accurate.

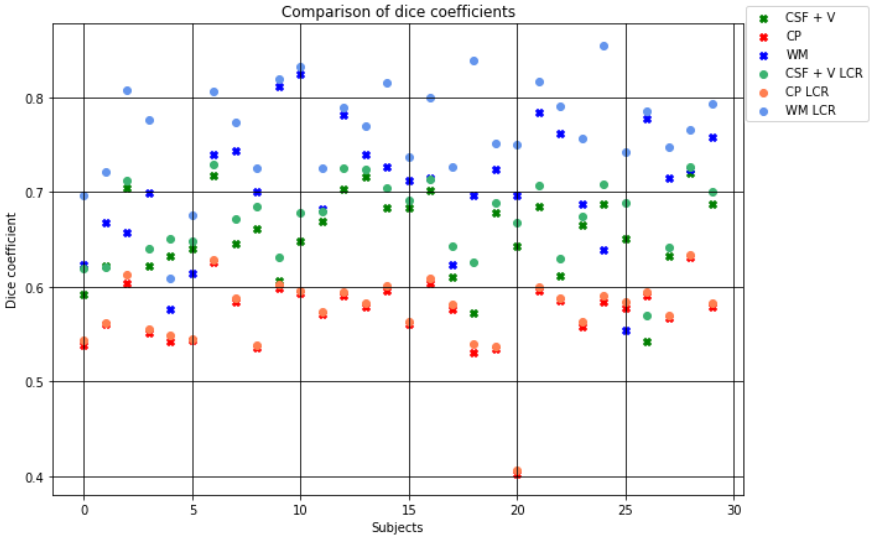

Figure 45 - Comparison of 3D dice coefficient for 30 dHCP subjects for the multilabel classification of the three different labels.

In *Figure 46* the confusion matrix of a specific subject is plotted. It is necessary to remark that it has been normalized to the total number of pixels. It can be easily observed that for the Label 0, which corresponds to the background, and for the Label 1, which corresponds to the cerebrospinal fluid together with the ventricles, there is no significant confusion. However, for both Label 2, which is the CP, and for Label 3, which is the white matter, the number of misclassified pixels increases. The biggest misclassification appears when it the pixel was supposed to be classified as CP and it ended up classified as white matter.

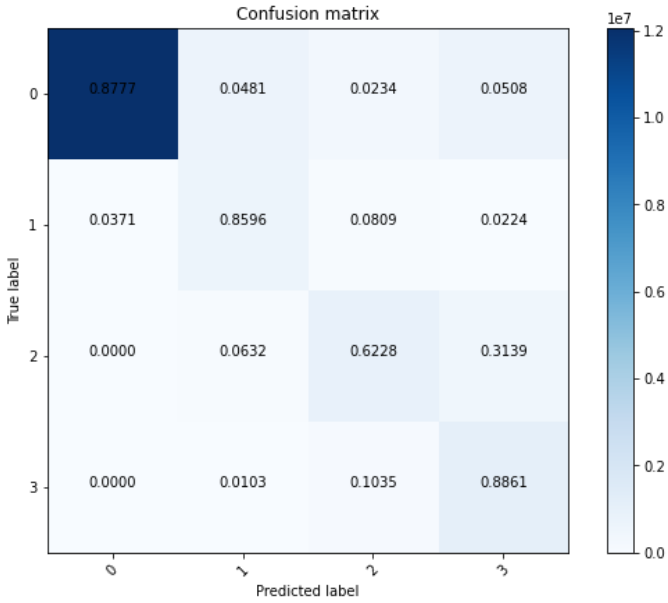

Figure 46 - Confusion matrix for the subject CC00500XX05_145900.

In *Figure 47* the sagittal, axial and coronal cuts of the 3D reconstruction of the same subject from the confusion matrix are printed.

The first row corresponds to the prediction of the cerebrospinal fluid with ventricles (Label 1). It can be easily seen that the prediction of the cerebrospinal fluid is mostly correct except for the *line* predicted around the brain, which corresponds to the Label 0 or background. This can be observed in the confusion matrix as well, since there appears to be a confusion between the background and the cerebrospinal fluid, in which the only background misclassified pixels are labeled as 1.

The second row corresponds to the prediction of the cortical plate (Label 2) and the third row corresponds to the prediction of the white matter (Label 3). With respect to the CP prediction some pixels that should be labeled as white matter are classified as CP as it can be seen in the confusion matrix. These pixels are mostly located in the center of the brain. It can be easily observed in the three different cuts: in the CP segmentation appear as blue since there is no corresponding ground truth and in the white matter segmentation appear as red since they are misclassified as CP. With respect to the white matter prediction an important amount of pixels are classified as CP instead of white matter. This can be seen in the confusion matrix (True label = 2, Predicted label = 3). It can be easily observed in the segmentation of the CP and of the white matter of *Figure 47*. In the segmentation of the CP it can be easily seen that a lot of pixels are not classified as CP, since they appear in red. In the segmentation of the white matter, a lot of pixels of the CP are misclassified as white matter hence, they appear in blue. These misclassified pixels are generally the deeper folds of the CP.

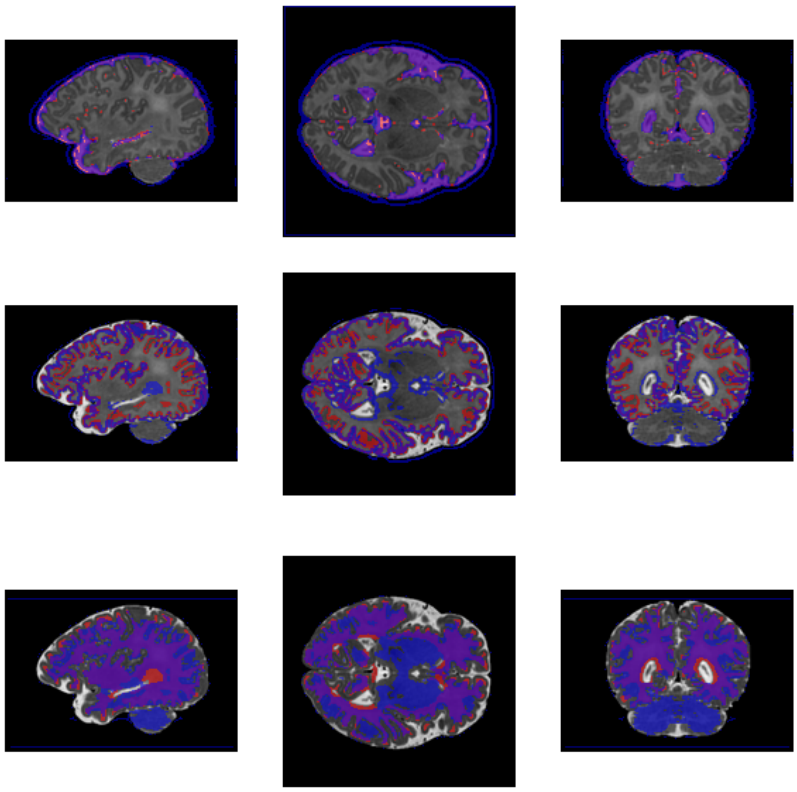

Figure 47 - Sagittal (left), axial (center) and coronal (right) cuts of the 3D representation for the subject CC00500XX05_145900. Cerebrospinal fluid with ventricles (first row), cortical plate (second row), white matte (third row). Predicted mask (blue) and ground truth (red) are plotted.

3.2.3 Comparison of both performances

In this section the idea is to compare the 3D dice coefficients for both monolabel and multi label classification. In *Figure 48*, the dice scores can be observed. It can be clearly concluded that the results obtained with mono label training are much better than the ones obtained for multi label training, as expected. The dice coefficients of the monolabel evaluation are around 0.8-0.9 and, however, the dice coefficients of the multi label evaluation are around 0.6-0.7, which is a significant decay. Also, as already said, the fact that the post processings chosen as the best ones for the multilabel classification are LCR for the three labels gives away the fact that the predictions are not accurate, as explained for *Figure 45*.

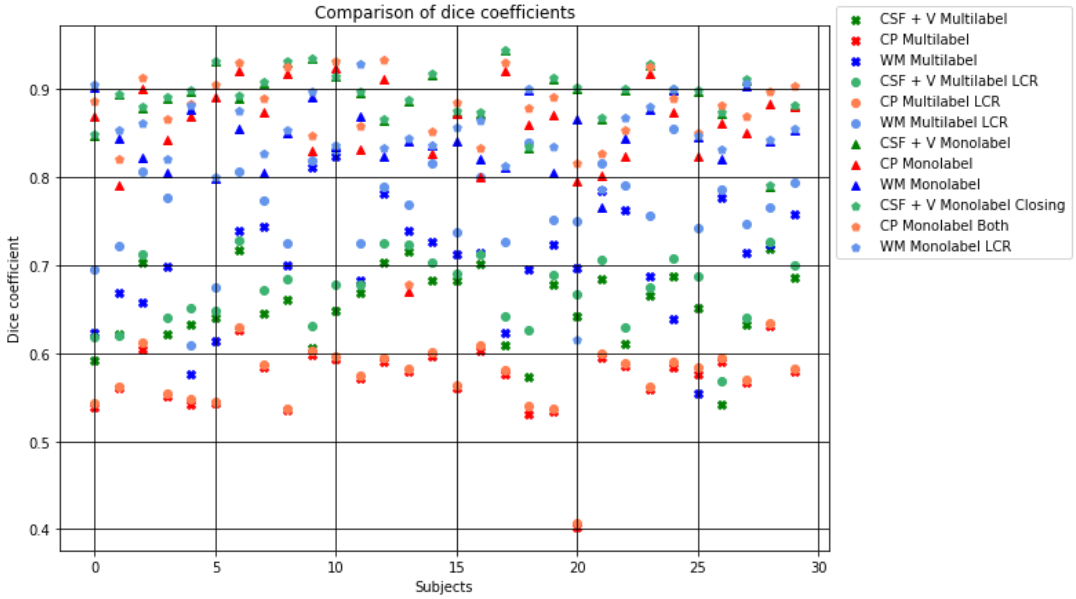

Figure 48 - Comparison of 3D dice coefficient for 30 dHCP subjects for both monolabel and multilabel classification of the three different labels.

3.2.4 Application to fetuses

As indicated at the beginning of this *section 3.2*, both the different labels independently and together were trained without noise injection because, since the area of the labels is bigger and more homogen, adding the noise changes too much the histogram pixels distribution. Consequently, it was really important to test if the network still was invariant to noise and performed well either way. Because of that, the images of the fetuses are plotted below.

In *Figure 49*, the prediction of the fetuses when the network is trained independently per each label is plotted in *red*. It can be easily seen that even if there is no noise injection during the training the performance is really good. After studying which morphological post processing per each label obtained the better performance in *section 3.2.1* it was applied in the fetuses and it can be observed in the *blue* mask. As it can be observed, the best post processing obtained for these predictions are the same as obtained in *section 3.2.1*: for cerebrospinal fluid with ventricles the closing; for the CP the combination of LCR and closing and, for the white matter the LCR. This shows that these predictions are accurate enough to be coherent with their best post processing as explained previously in *section 3.2.1*.

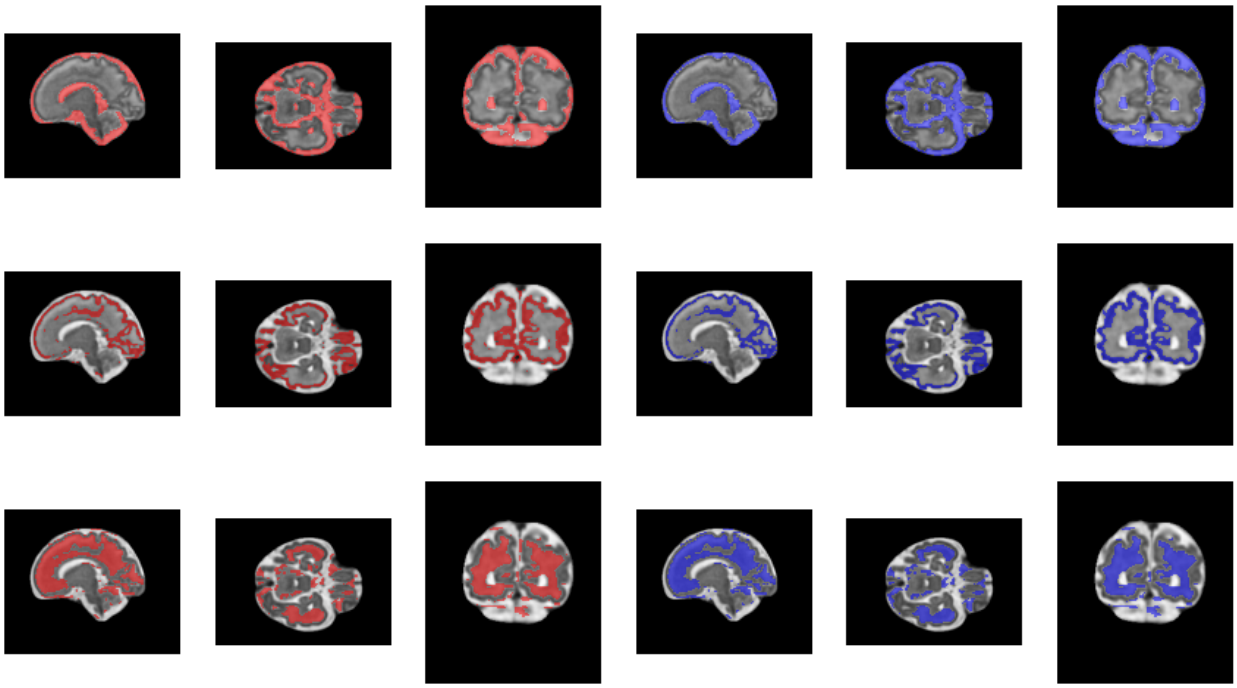

Figure 49 - Predictions of 3D fetus images (red) for each label separately: cerebrospinal fluid with ventricles (first row), cortical plate (second row) and white matter (third row). Morphological post processing has been applied in each of them (blue): for cerebrospinal fluid with ventricles closing; for cortical plate LCR + closing; and, for white matter LCR. Subject marsFet021 (MRI date 32).

In *Figure 50*, the predictions per each label can be observed in *red* and the LCR post processing applied in *blue*. In here for all of them was applied LCR since in the *section 3.2.2* it was observed that it achieved the better performance for all three labels.

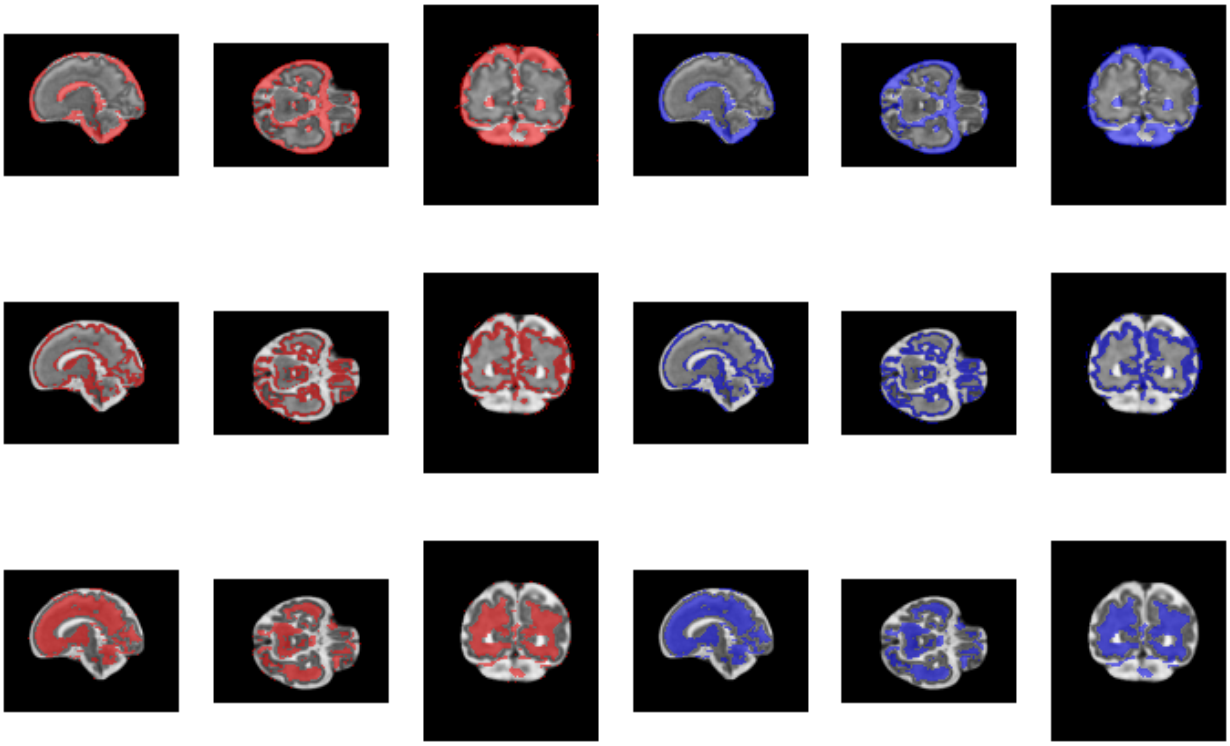

Figure 50 - Predictions of 3D fetus images (red) for multilabel: cerebrospinal fluid with ventricles (first row), cortical plate (second row) and white matter (third row). Morphological post processing LCR has been applied in each of them (blue). Subject marsFet021 (MRI date 32).

If both predictions, monolabel and multilabel, are compared even if none of them is awfully predicted, it can be easily seen that the better performance is obtained with training the network per each label separately. For example, for the cerebrospinal fluid there are voids in the mask when predicting it with multilabel training but a lot of them disappear when the network is done with just this label. For the CP prediction, when training it with multilabel, lots of other parts of the brain are considered CP as well as the edge of the brain; the prediction is much more accurate when training just with the label of *drawem9* CP independently. And, finally, for the white matter label happens something similar, the performance with training it independently is better. However, it could be said that is the one that obtains better results when training it with multilabel.

3.2.5 Conclusions for multilabel segmentation

For the multilabel segmentation, the overall conclusion is that it does not work properly since it degradates the results obtained with monolabel, which should happen contrarily. However, since the results are not completely awful, it can be concluded as well that maybe there is an error with the weights values or that some other approach could be considered, for example, in order to deal with imbalanced classes such as data upsampling or using another type of loss who may be performing better for multilabel classification.

3.3 Final conclusions

The final conclusions can be splitted into two main parts: monolabel segmentation and multilabel segmentation.

Firstly, it is important to talk as well about the amount of resources each network consumes for the monolabel segmentation. When working with the basic UNet the number of parameters that need to be trained are 74. When working with the residual UNet, the number of parameters that need to be trained are 125. This implies that the training time for the residual UNet (around three weeks) is enormously bigger than the time used for the basic UNet (around 20 hours). When working with NN it is important to consider the computation time and, even more important if the resources are limited.

Regarding the performances, for the FetalCPSeg architecture it was concluded clearly that the best performance was achieved by just using the backbone network. When comparing both UNets, the main conclusion is that the better performance is obtained with the residual UNet obtained from the FetalCPSeg architecture. With respect to the fetuses, it seems that the CP predicted by the residual UNet is more robust but it should be supervised by an expert to check that no misclassifications of other areas of the brain appear as CP.

For the multilabel segmentation, the conclusion was clear: the multilabel does not help to improve the prediction of the CP, which is already pretty good with mono label classification. However, this part should be studied in more depth in the upcoming projects.

One conclusion that can be generalized for both architectures as well as for multi and mono label segmentation is that the predictions do not seem to depend on the age of the MRI acquisition. This leads to an important question, which is if the training set was well selected. If the prediction performance does not depend on the MRI week, the training set could be enlarged to more subjects or, moreover, could be selected randomly among all the dHCP images without caring about the age of the subject. This would give more richness to the dataset and invariance to the anatomical changes of the CP.

4. Future Work

This project has opened several branches and most of them need a more thorough and deep study. This work can be splitted into two big parts: first, the study of the UNet and, second, the study of the code of the FetalCPSeg. In each of these parts a more profound study could take place.

Firstly, let's deepen on the first part, the study of the Pytorch UNet. In this part of the project, most of the different points of study got covered but the part of multilabel segmentation would need more testing done. On one hand, a good point of study would be to try other function losses that perform better than the weighted cross entropy, for example, the Focal Loss studied in the work [47] or to play with upsampling in order to balance the different classes and do not work with the imbalanced dataset or find another way to apply the weights [48]. In addition to that, it could be good to study exactly the influence of the different parts of the brain on the prediction of the cortical plate.

Secondly, for the study of the network FetalCPSeg more studies could be performed. On important study, would be, if the amount of 3D stack images obtained for the training make it possible, try to make the training and predictions with 3D images, as it was originally designed. Furthermore, the multilabel segmentation should be studied for this architecture as well.

Thirdly, in my opinion it would be good to spend an important amount of time and resources to study more thoroughly the influence of the training dataset, since it was observed that the performance of the predictions do not depend on the MRI week. For example, if the amount of images used increases the networks improve their performance or try to train it with other than the youngest newborn MRIs and check if it is true that the performance is better when the youngest are used, even if this seems counter-intuitive. It would be really interesting as well to try to train the networks with fetal MRIs and study the impact of this training dataset.

Finally, in the field of study of the pre and post processing several considerations could be taken. On the pre processing part, a more thorough study of the data augmentation applied could be performed in order to see if it could be possible to apply other modifications to enrich the training set. On the post processing part, the most important part to keep on studying is the application of the different mathematical morphology operations in order to improve the predictions of the networks. Different scopes of study could be, first, the impact of the different kernels when applying the classical operations or, secondly, apply 2D post processing and reconstruct later the images. In addition, other more complex morphological operations could be tried out. Another significant study that could be performed for the post processing is the study of the thresholding method. It is possible that maybe using another method such as [49] it could improve significantly the results.

Other post processing techniques could be used. For example, in the work [50] they perform a prediction in the sagittal, axial and coronal cuts and then they reconstruct it. This could be a really interesting approach to try out.

Figures List

Reference

[1] INT - Institut de Neurosciences de la Timone

[2] MeCA research group » Methods and Computational Anatomy

[3] F. Rousseau, et al., "In Vivo Human Fetal Brain Analysis Using MR Imaging." In: Reissland N., Kisilevsky B. (eds) Fetal Development. Springer, Cham. (2016), doi: 10.1007/978-3-319-22023-9_20

[4] F. Rousseau, et al., "BTK: An open-source toolkit for fetal brain MR image processing", Comput. Methods Programs Biomed. (2012), doi: 10.1016/j.cmpb.2012.08.007

[5] Haoran Dou, Davood Karimi, Caitlin K. Rollins, Cynthia M. Ortinau, Lana Vasung, Clemente Velasco-Annis, Abdelhakim Ouaalam, Xin Yang, Dong Ni, Ali Gholipour, "A Deep Attentive Convolutional Neural Network for Automatic Cortical Plate Segmentation in Fetal MRI", arXiv:2004.12847v1, 27 Apr 2020

[6] Ebner, M. et al., An automated framework for localization, segmentation and super-resolution reconstruction of fetal brain MRI, NeuroImage, doi: 10.1016/j.neuroimage.2019.116324

[7] Seyed Sadegh Mohseni Salehi, Seyed Raein Hashemi, Clemente Velasco-Annis, Abdelhakim Ouaalam, Judy A. Estroff, Deniz Erdogmusy, Simon K. Warfield, Ali Gholipour, "Real-time automatic fetal brain extraction in fetal MRF by Deep Learning", arXiv:1710.09338v1, 25 Oct 2017

[8] N. Khalili, et al., Magnetic Resonance Imaging, doi: 10.1016/j.mri.2019.05.020

[9] J. Li, Y. Luo and L. Shi et al., "Automatic fetal brain extraction from 2D in utero fetal MRI slices using deep neural network", Neurocomputing, doi: 10.1016/j.neucom.2019.10.032

[10] http://www.developingconnectome.org/

[11] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation" arXiv:1505.04597v1 [cs.CV], 18 May 2015

[12] Investopedia. Artificial Intelligence (AI) Definition.

[13] Investopedia. Machine Learning Definition.

[14] Investopedia. Deep Learning Definition.

[15] Wikipedia. Convolutional neural network.

[16] Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning Book. MIT Press - Massachusetts - England. 2016.

[17] Pierre-Henri Conze. TAF MCE - UE Computer Vision. Object detection and segmentation with deep learning. IMT Atlantique | Bretagne - Pays de la Loire - France. February 2020.

[18] Machine learning mastery. A Gentle Introduction to the Rectified Linear Unit (ReLU).

[19] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014), arXiv:1411.4038 [cs.CV]

[20] milesial/Pytorch-UNet

[21] Towards data science. Residual blocks — Building blocks of ResNet | by Sabyasachi Sahoo.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[24] S. Xie and Z. Tu, "Holistically-nested edge detection," in Proceedings of the IEEE Int. conference on computer vision, 2015, pp. 1395–1403., arXiv:1504.06375

[25] bchimagine/FetalCPSeg.

[26] Towards data science. What are hyperparameters and how to tune the hyperparameters in a deep neural network

[27] Machine learning mastery. Difference Between a Batch and an Epoch in a Neural Network.

[28] Towards datas science. Epoch vs Batch Size vs Iterations | by SAGAR SHARMA.

[29] Nitish Shirish Keskar et al., "On large-batch training for deep learning: generalization gap and sharp minima" arXiv:1609.04836v2 [cs.LG] 9 Feb 2017

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[31] Ameen Mohammed Abd, Alsalam Selami, Ahmed Freidoon Fadhil, "A study of the Effects of Gaussian Noise on Image Feature", Kirkuk University, College of Engineering, Electrical Engineering Dept, April 2016

[32] Wikipedia. Affine transformation.

[33] S. Vaishali, K. K. Rao and G. V. S. Rao, "A review on noise reduction methods for brain MRI images," *2015 International Conference on Signal Processing and Communication Engineering Systems*, Guntur, 2015, pp. 363-365, doi: 10.1109/SPACES.2015.7058284.

[34] Charles Boncelet, The Essential Guide to Image Processing. Al Bovik, 2009: The Essential Guide to Image Processing

[35] H. M. Ali, "A new method to remove salt & pepper noise in Magnetic Resonance Images," *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, Cairo, 2016, pp. 155-160, doi: 10.1109/ICCES.2016.7821992.

[36] I. Kumar, H. S. Bhadauria, J. Virmani and J. Rawat, "Reduction of speckle noise from medical images using principal component analysis image fusion," *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, Gwalior, 2014, pp. 1-6, doi: 10.1109/ICIINFS.2014.7036562.

[37] Faouzi Benzarti, Hamid Amiri, "Speckle Noise Reduction in Medical Ultrasound Images", Signal, Image and Pattern Recognition Laboratory (TSIRF) ENIT Engineering School of Tunis (ENIT).

[38] Wikipedia. Thresholding (image processing).

[39] Naresh Kumar Garg, "Binarization Techniques used for Grey Scale Images", International Journal of Computer Applications (0975 – 8887) Volume 71– No.1, June 2013, doi: 10.5120/12320-8533

[40] N. Otsu, "A thresholding selection method from gray-scale histogram", IEEE Transactions on System, Man, and Cybernetics 9 (1979) 62–66.

[41] Wikipedia. Mathematical morphology.

[42] Medium. Morphological Operations in Image Processing | by Nickson Joram

[43] Silvana Dellepiane. Digital Image Processing - Mathematical Morphology. Università degli Studi di Genova - Italy. 2018-2019.

[44] What-When-How. Morphology - Introduction to Video and Image Processing - page 97

[45] Wikipedia. Connected-component labeling.

[46] Chen-Yu Lee and Saining Xie and Patrick Gallagher and Zhengyou Zhang and Zhuowen Tu, Deeply-Supervised Nets, arXiv:1409.5185 [stat.ML] 2014

[47] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar, "Focal Loss for Dense Object Detection", arXiv:1708.02002v2 [cs.CV] 7 Feb 2018

[48] Data science. Stack Exchange. CNN - imbalanced classes, class weights vs data augmentation

[49] Koichi Sasakawa Shin-ichi Kuroda Shigeki Ikebata, "A method for threshold selection in binary images using mean adjacent-pixel number", 1991, doi: 10.1002/scj.4690220307

[50] Leonie Henschel, Sailesh Conjeti, Santiago Estrada, Kersten Diers, Bruce Fischl, Martin Reuter, "FastSurfer - A fast and accurate deep learning based neuroimaging pipeline"