Suemy Inagaki Pinheiro Fagundes

DC-UNet for White Matter Lesions Segmentation

FINAL PROJECT

SCIENTIFIC TECHNICAL CENTER - CTC

DEPARTMENT OF INFORMATICS

Undergraduate Program in Computer Engineering

Rio de Janeiro
November 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Suemy Inagaki Pinheiro Fagundes

DC-UNet for White Matter Lesions Segmentation

Final Project

Final project presented to the Computer Engineering Course as a partial requirement to obtain the title of Computer Engineer.

Advisor : Prof. Marcelo Gattass
Co-advisor: Luiz Fernando Trindade Santos

Rio de Janeiro
November 2022

Acknowledgments

To PUC-Rio, especially to the Community Vice-Rectory, for providing me with a full scholarship throughout my graduation, without which I wouldn't have the opportunity to graduate.

To Luiz Fernando for all the support and guidance. He was certainly one of the essential people for the completion of this work.

To Professor Marcelo Gattass for accepting to be my new advisor and for encouraging me to carry out this work.

To Felipe, my parents and my grandmother, for their support and encouragement. They were with me in all moments of my graduation and this achievement belongs to all of us.

Abstract

Inagaki Pinheiro Fagundes, Suemy; Gattass, Marcelo (Advisor); Trindade Santos, Luiz Fernando (Co-Advisor). **DC-UNet for White Matter Lesions Segmentation**. Rio de Janeiro, 2022. 28p. Final Project – Department of Informatics, Pontifical Catholic University of Rio de Janeiro.

Analysis and segmentation techniques of magnetic resonance images of the brain have been widely explored. Manual interpretation of the brain image is quite time-consuming and directly depends on the operator's assessment. Thus, some automations were previously proposed, but recently, the study of automation using Deep Learning has gained prominence. In this context, we propose a model of neural networks with DC-UNet architecture for the segmentation of lesions in white matter in brain images.

Keywords

DC-UNet; Medical image segmentation; Brain lesions; Brain white matter.

Resumo

Inagaki Pinheiro Fagundes, Suemy; Gattass, Marcelo (Orientador); Trindade Santos, Luiz Fernando (Coorientador). **DC-UNet para segmentação de lesões de substância branca.** Rio de Janeiro, 2022. 28p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Técnicas de análise e de segmentação de imagens de ressonância magnética de cérebros vêm sendo amplamente exploradas. A interpretação manual da imagem do cérebro é bastante trabalhosa e depende diretamente da avaliação do operador. Assim, algumas automatizações foram propostas previamente, mas, recentemente, o estudo da automatização usando Deep Learning tem ganhado destaque. Nesse contexto, propomos um modelo de redes neurais com arquitetura DC-UNet para a segmentação lesões na substância branca em imagens do cérebro.

Palavras-chave

DC-UNet; Segmentação de imagens médicas; Lesões no cérebro; Substância branca do cérebro.

Table of contents

List of figures

1
Introduction

Quantitative analysis of magnetic resonance (MR) of brain images is necessary for the study of the aging brain and to support diagnostics in clinics (BOER et al., 2009). The main challenge in the manual segmentation of the brain is that it is a very time-consuming task and its performance is directly related to operator experience (DONG et al., 2017). Therefore, there is a large interest in automating the segmentation and analysis of MR images of the brain, especially the segmentation of white matter (BOER et al., 2009).

Previous research in medical images segmentation has mainly used traditional methods such as boundary extraction, threshold-based segmentation and region-based segmentation (DU et al., 2020). However, deep learning methods for image segmentation are being widely explored recently (DU et al., 2020). In this work, we use deep learning to segment lesions in the white matter of the human brain using DC-UNet architecture, which is a variation of an architecture proposed by Ronneberger, Fischer e Brox (2015), U-Net. The latter is commonly used for medical image segmentation task, but the former is an improved version that approaches our problem better.

One of the main challenges in training this network is data imbalance. This is particularly problematic in medical imaging applications such as lesion segmentation, as this causes a higher rate of false negatives than false positives. As we have very unbalanced data, where the number of pixels without lesions is many times greater than the number of pixels with lesions, we use the Tversky loss function, proposed by Salehi, Erdogmus e Gholipour (2017), which addresses the problem of image imbalance.

Finally, this work is divided into 7 chapters. In the chapter 2, we give an overview of the project, in the chapter 3 we explain the techniques used for the data pre-processing, in the chapter 4 we explain the evaluation metrics, in the chapter 5 we show the work methodology, such as the neural network architecture and the cross-validation method used, in the chapter 6 we show our results and also compare the results obtained in each approach and, lastly, in the chapter 7 we talk about the conclusion and future work.

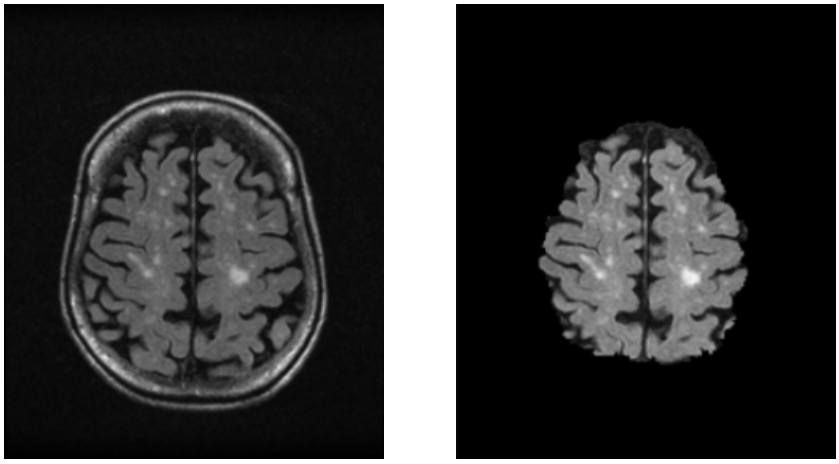
2
Overview

Our objective is to develop a model capable of segmenting a lesion in the brain with the least amount of false negatives possible. The solution we implemented for this involves the use of deep learning. There are many deep learning architectures available, but the ones that best suit the medical image segmentation problem are U-Net and its variations (RONNEBERGER; FISCHER; BROX, 2015). In our case, we chose to use DC-UNet and the reason is explained in chapter 5.

Along with the choice of architecture mentioned above, we have selected the Tversky and Binary Cross Entropy loss functions. We use each one to train a model and in the chapter 6 we show that the former trained a better model, as it deals better with the trade-off between false positive and false negative.

Another relevant point to the problem is our dataset. It consists of magnetic resonance images of brains provided by the DASA company. Because they are MR images, they could have different gray scales and this could negatively impact the performance of our model. In addition, the brains came with the skull and that could also hamper performance. So we had to deal with these challenges in the pre-processing step. The images 2.1-a and 2.1-b and show the before and after of the pre-processing that will be explained in the chapter 3.

Also related to dataset, we had to use techniques to deal with the limited amount of data as we have only 46 samples with lesion markings made by specialists. So we applied the k-fold cross validation technique during the training stage and this is explained in the chapter 5.



(a) A slice of the original data (b) A slice of pre-processed data

Figure 2.1: Original and pre-processed brain comparison

3
Data Pre-Processing

It is common for medical images to contain unknown noise, poor image contrast, poor homogeneity, and parts irrelevant to the problem (RAMANI; VANITHA; VALARMATHY, 2013). The main purpose of pre-processing the images is to improve the quality of the image and make it ready for training (RAMANI; VANITHA; VALARMATHY, 2013).

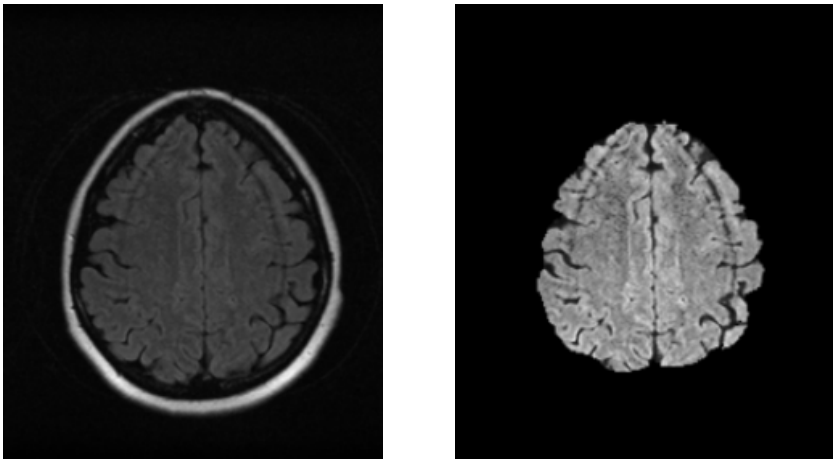
In our case, when analyzing the images, we found that the skull is an irrelevant part for the segmentation as the lesions are not in the skull. So its existence would hinder the training, as it would make it difficult to extract relevant features for the problem. In addition, MRI machines usually generate images of different intensities and this can be another factor hindering the extraction of features by our model.

Therefore, in this pre-processing step, we explain how we dealt with each of the issues to make the images ready for training.

3.1
Skull-stripping

The brain images present parts that are irrelevant to the problem we are addressing, such as the skull and the entire region external to it. Therefore, it is crucial that we remove these elements from the images before using them for training and testing.

Bauer, Fejes e Reyes (2013) developed a filter called *StripTsImageFilter* for the *Python* Insight Toolkit (*itk*) library that quickly removes the skull from the brain image. We used this filter for skull-stripping and one of the resulting images is shown below 3.1.



(a) Brain with skull (b) Brain without skull

Figure 3.1: Brain before (left) and after (right) skull-stripping

3.2
Histogram Matching

MR scanners show variations in scanner sensitivity and these variations cause the images to have a different grayscale intensity from each other. This could affect the performance of our model, so we applied the histogram matching technique to our images. According to the work of Wang et al. (1998), this method reduces image intensity variation and can equalize the intensity of the white matter of the brain.

The Histogram matching technique consists of modifying the contrast of an image based on the contrast levels of a target image. In this way, it is possible to balance the contrast levels of a group of images. In our dataset, now that the images are without the skull, we chose the image of index 0 to be the target and histogram matched the remaining images with the chosen image.

Below we show one example of histogram matching. In the Figure 3.2, the change in the histogram is clear. In the histogram after the histogram matching, the format of the distribution of intensities looks more like the histogram of the target image. This facilitates the extraction of features over the network. The Figure 3.2 also shows the resulting image after pre-processing.

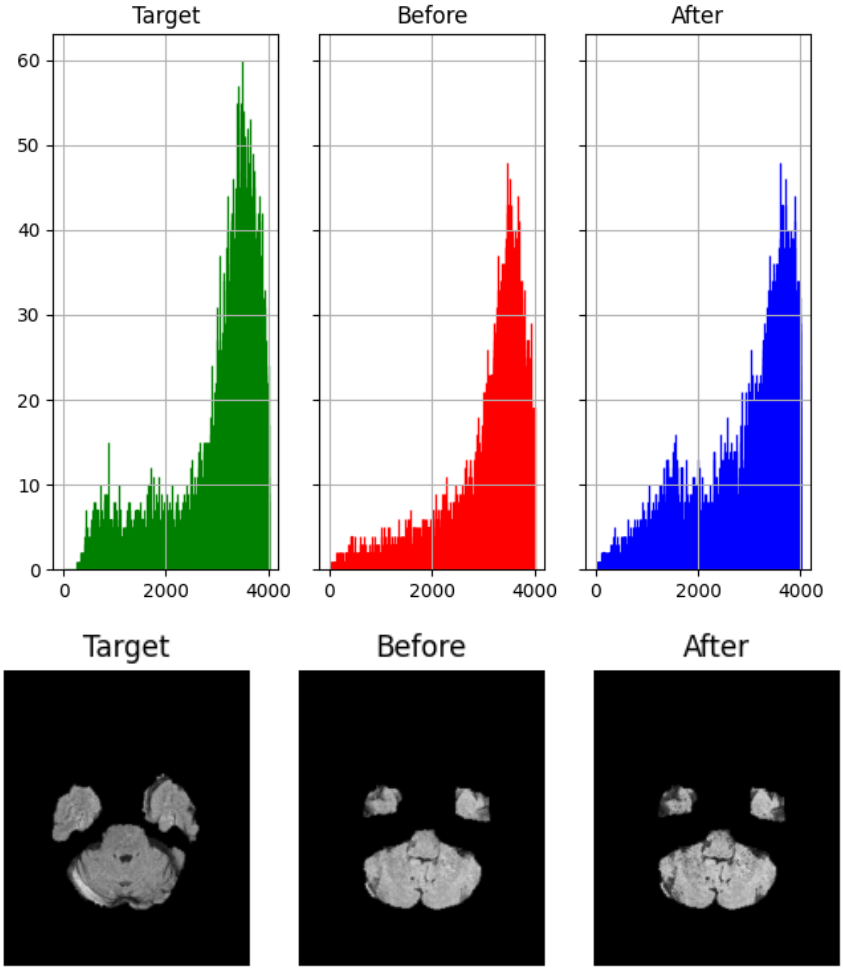


Figure 3.2: Comparison of brains and histograms before and after histogram matching

4
Evaluation Metrics

In this chapter we will discuss the metrics used to evaluate our model: the accuracy and F1-Score. In the chapter 6, we present the results obtained in each metric.

Some acronyms necessary to understand the methodology are: TP: the number of correct positive results; FP: the number of wrong positive results; TN: the number of correct negative results and FN: the number of wrong negative results.

4.1
Accuracy

Accuracy is the ratio between the number of correct answers and the total number of input samples, but it works well only if there are equal number of samples belonging to each class.

Our images are very unbalanced, as they have many more regions without lesions than regions with lesions. Therefore, the accuracy can give us a false sense of the model's high performance.

$$\text{Accuraccy} = \frac{\text{Correct answers}}{\text{Total input samples}} \tag{4-1}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4-2}$$

4.2
F1 Score

F1 Score is a value between 0 and 1 and uses harmonic mean to seek a balance between *precision* and *recall*. It evaluates the model's precision and robustness, so the higher the F1 Score, the better the model performs. It can be mathematically calculated by the equation below.

$$\text{F1-Score} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{4-3}$$

Where precision is the ratio between the number of correct positive results and the number of positive results predicted.

$$\text{Precision} = \frac{\text{Correct positive results predicted}}{\text{Total positive results predicted}} \tag{4-4}$$

On the other hand, recall is the ratio between the number of correct positive results and the number of all samples that should have been classified as positive.

$$\text{Recall} = \frac{\text{Correct positive results predicted}}{\text{All samples that should have been classified as positive}} \tag{4-5}$$

In a simplified way, we have the expressions 4-6 and 4-7 below for precision and recall, respectively.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4-6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4-7}$$

5
Methodology

In this chapter we cover the architecture of the neural network, the cross-validation method, the loss function and the optimizer we used in the project. The choice of each methodology significantly impacts the performance of the model due to the low number of images with masks that we have for training. Thus, we chose to use a U-Net based neural network architecture: DC-UNet. This architecture can train with very few images, giving us satisfactory results. Furthermore, the cross validation method allows us to obtain a more generic model and the loss functions we are using handles unbalanced data well. We cover each choice in the sections below.

5.1
Architecture

Despite U-Net being very well known and popular for segmenting medical images, according to Ibtehaz e Rahman (2020), the MultiRes U-Net architecture surpasses U-Net and gives us much better results than the first one, as it is capable of to extract features of different scales. However, according to Lou, Guan e Loew (2021) when medical imaging is very challenging, MultiRes U-Net does not perform well. Therefore, Lou, Guan e Loew (2021) suggest a new architecture: the DC-UNet. We'll use DC-UNet and explain all three in the sections below.

5.1.1
U-Net

The U-Net architecture consists of two parts: the left part is contraction, also called encoder and the right part is expansion, also called decoder. The name comes from the U-shaped architecture, as shown in the Figure 5.3. The training image enters from the left, in the encoder, receives some layers of convolutions and then the decoder uses the transpose convolution to generate the segmentation mask as output. (RONNEBERGER; FISCHER; BROX, 2015; ANJOS et al., 2022).

The contraction part consists of four blocks, each containing two convolution layers 3×3 with ReLU as activation, followed by a 2×2 max pooling layer. The Figure 5.1 demonstrates the structure well.

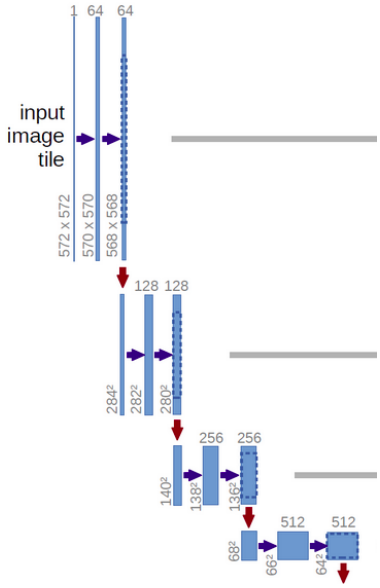


Figure 5.1: U-Net contraction (RONNEBERGER; FISCHER; BROX, 2015)

On expansion, the image is upsized to its original size via 4 blocks, each containing a 2×2 transpose convolution step, a concatenation with the corresponding block from the contraction step and then two more 3×3 convolution layers as shown in Figure 5.2 (RONNEBERGER; FISCHER; BROX, 2015).

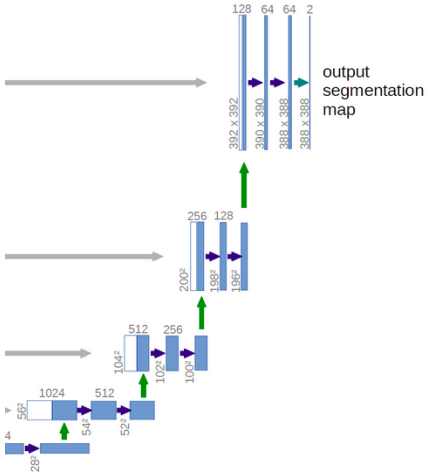


Figure 5.2: U-Net expansion (RONNEBERGER; FISCHER; BROX, 2015)

The transposed convolution is responsible for expanding the size of the images. After that, in the concatenation step, by combining information from the corresponding contraction part, a more accurate prediction is achieved (RONNEBERGER; FISCHER; BROX, 2015).

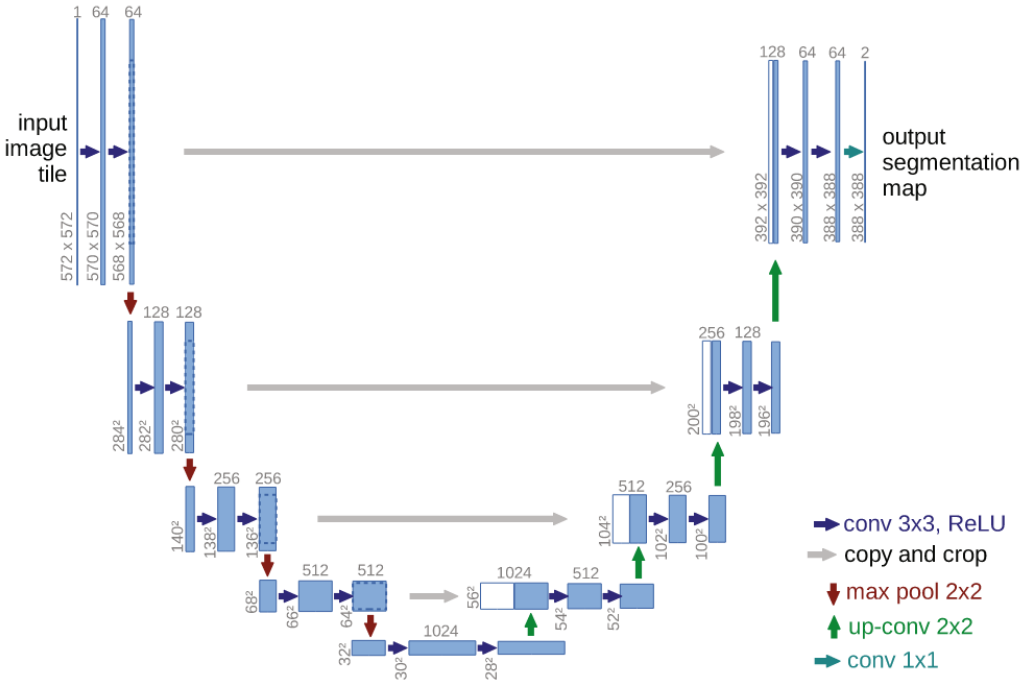


Figure 5.3: U-Net architecture (RONNEBERGER; FISCHER; BROX, 2015)

5.1.2 MultiRes U-Net

MultiRes U-Net is an architecture that modified the U-Net and surpassed it in 5 datasets in the image segmentation task (IBTEHAZ; RAHMAN, 2020). The main difference between the two architectures is that MultiRes U-Net replaces the two U-Net 3×3 convolution sequences at each level with a block that the authors call MultiRes Block and that it uses MultiRes Path to combine encoder and decoder features (see Figures 5.5 and 5.6). Figure 5.4 below shows the MultiRes U-Net architecture.

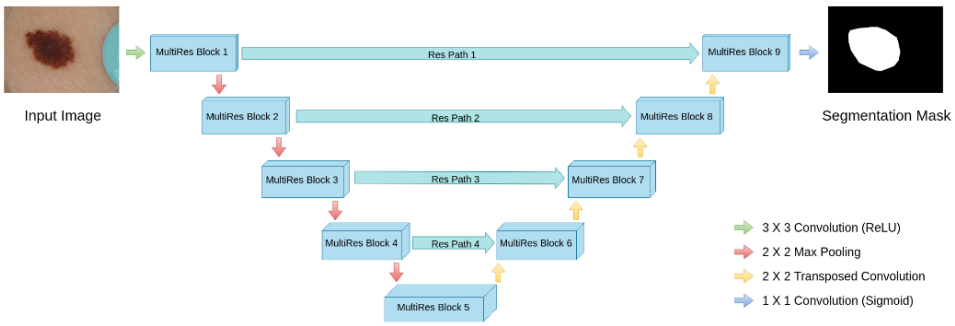


Figure 5.4: MultiRes U-Net architecture (IBTEHAZ; RAHMAN, 2020)

The structure of each MultiRes block is illustrated in Figure 5.5. It basically consists of successive layers of 3×3, 5×5 and 7×7 convolutional filters, where the last two, which are more expensive and larger, are factorized into successions of 3×3 filters. Furthermore, it has a concatenation of a residual connection for the conservation of dimension.

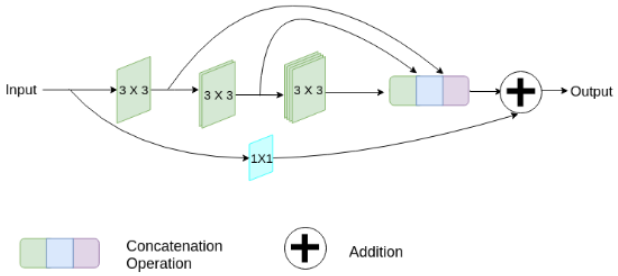


Figure 5.5: MultiRes Block (IBTEHAZ; RAHMAN, 2020)

To combine the encoder and decoder feature maps, instead of doing it in a straight-forward manner, Ibtehaz e Rahman (2020) pass the encoder features through a sequence of convolution layers, as illustrated in Figure 5.6. This reduces the semantic gap between encoder and decoder features.

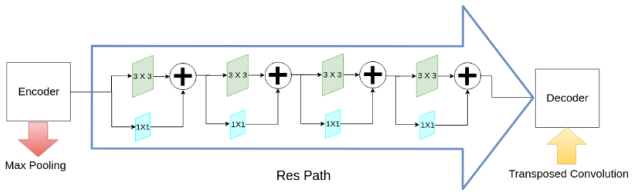


Figure 5.6: ResPath (IBTEHAZ; RAHMAN, 2020)

5.1.3 DC-UNet

According to Ibtehaz e Rahman (2020), the residual connection they added in the MultiRes Block provides a few additional spatial features and this may not be enough for more challenging segmentation tasks, such as segmenting very small lesions (LOU; GUAN; LOEW, 2021) . To solve the problem of lack of spatial features, Lou, Guan e Loew (2021) proposed the DC-UNet, which replaces the residual connection of the MultiRes Block with a sequence of three convolutions 3×3. The authors called this new block Dual Channel Block and its structure is shown in Figure 5.7.

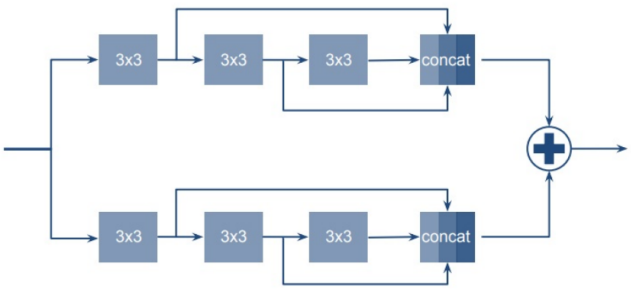


Figure 5.7: Dual Channel Block (ANJOS et al., 2022)

By replacing the MultiRes block with the DC block shown in Figure 5.7, we obtain the DC-UNet architecture shown in Figure 5.8.

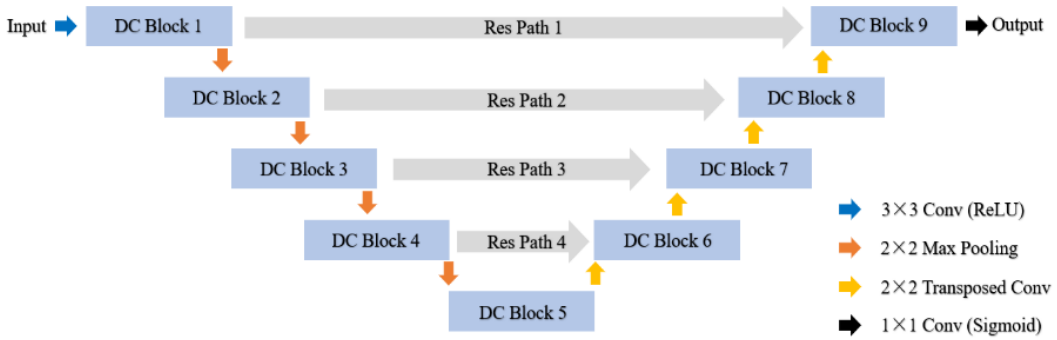


Figure 5.8: Dual Channel U-Net architecture (LOU; GUAN; LOEW, 2021)

5.2 K-fold Cross-Validation

A common problem faced when we want to evaluate a model is that it may have good prediction ability on training data, but poor prediction ability on unseen data. Cross-validation is a procedure to estimate performance in a generalized way and that allows us to compare the performance of different models and find out the best one for the modeled problem. (REFAEILZADEH; TANG; LIU, 2009)

K-fold cross-validation is one of the most used methods in machine learning and consists of dividing the data into k folds of equal or nearly equal size. In sequence, k training iterations are performed so that in each iteration, a different fold is chosen for testing and the remaining $(k-1)$ are used for training (REFAEILZADEH; TANG; LIU, 2009).

The Figure 5.9 shows the scheme of the 4-fold method. It shows the data separated into training and testing for each iteration, represented as a row. After the four iterations, we will have four different models, each with a performance. The overall performance of our model will be the arithmetic mean of the performances.

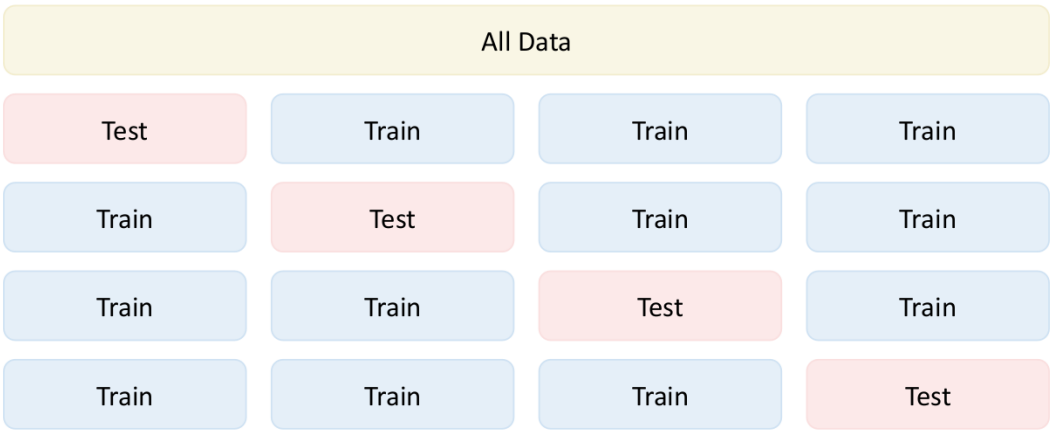


Figure 5.9: K-fold scheme when $k = 4$

We use K-fold cross-validation with $K = 11$ and show the performances of each fold in the chapter 6.

5.3
Loss Function

The loss function evaluates how good the model is. The higher the value returned by the loss function, the worse the prediction of our model. And the smaller the value given by the loss function, the better the prediction of the model. So, in a simplified way, the loss function measures the absolute difference between the value predicted by the model and the correct value (HENNIG; KUTLUKAYA, 2007).

Choosing the loss function should be as important as defining the architecture used to address the problem. Zhao et al. (2015) compared several loss functions and showed that choosing the appropriate function for the problem directly impacts the result. In our case, we chose to train the model with the Binary Cross Entropy function and with Tversky, both explained below. We compare the results in the chapter 6.

5.3.1
Tversky

As mentioned before, our data are unbalanced. Not balancing them causes the learning process to converge to a local minimum of a loss function that is not the best for this case and leads the predictions to a skewed result (SALEHI; ERDOGMUS; GHOLIPOUR, 2017). To achieve a better tradeoff between precision and recall, we use the Tversky loss function, which is based on the Tversky index, proposed by Salehi, Erdogmus e Gholipour (2017).

In image segmentations, it is very common to choose Dice similarity coefficient, shown in the equation 5-1, to calculate the loss function. If it were used for the loss function in the training, it would weigh the FN and FP equally. However, we need to weigh the FN more than the FP as we want to detect even small lesions and we have unbalanced data. So instead of the Dice similarity coefficient, the Tversky cost function is based on the Tversky index, as shown in the equation 5-2 (SALEHI; ERDOGMUS; GHOLIPOUR, 2017).

P and G are the set of predicted and ground truth binary labels, respectively and α and β control the magnitude of penalties for FPs and FNs, respectively.

Dice similarity coefficient

$$D(P,G) = \frac{2 \mid PG \mid}{\mid P \mid + \mid G \mid} \tag{5-1}$$

Tversky index

$$S(P,G;\alpha,\beta) = \frac{\mid PG \mid}{\mid PG \mid + \alpha \mid P/G \mid + \beta \mid G/P \mid} \tag{5-2}$$

The Tversky loss function is defined below, where p_{0i} is the probability of pixel be a lesion and p_{1i} is the probability of pixel be a non-lesion. Furthermore, $g_{0i} = 1$ for a lesion pixel and $g_{0i} = 0$ for a non-lesion pixel and vice

verse for the g_{1i} (SALEHI; ERDOGMUS; GHOLIPOUR, 2017).

$$T(\alpha, \beta) = \frac{\sum_{i=1}^{N} p_{0i} g_{0i}}{\sum_{i=1}^{N} p_{0i} g_{0i} + \alpha \sum_{i=1}^{N} p_{0i} g_{1i} + \beta \sum_{i=1}^{N} p_{1i} g_{0i}} \tag{5-3}$$

The gradient of the equation 5-3 with respect to p_{0i} and p_{1i} are shown in the equations 5-4 and 5-5 below.

$$\frac{\partial T}{\partial p_{0i}} = 2\frac{g_{0j}(\sum_{i=1}^{N} p_{0i} g_{0i} + \alpha \sum_{i=1}^{N} p_{0i} g_{1i} + \beta \sum_{i=1}^{N} p_{1i} g_{0i}) - (g_{0j} + \alpha g_{1j}) \sum_{i=1}^{N} p_{0i} g_{0i}}{(\sum_{i=1}^{N} p_{0i} g_{0i} + \alpha \sum_{i=1}^{N} p_{0i} g_{1i} + \beta \sum_{i=1}^{N} p_{1i} g_{0i})^2} \tag{5-4}$$

$$\frac{\partial T}{\partial p_{1i}} = -\frac{\beta g_{1j} \sum_{i=1}^{N} p_{0i} g_{0i}}{(\sum_{i=1}^{N} p_{0i} g_{0i} + \alpha \sum_{i=1}^{N} p_{0i} g_{1i} + \beta \sum_{i=1}^{N} p_{1i} g_{0i})^2} \tag{5-5}$$

By adjusting the α and β hyperparameters, it is possible to control the trade-off between FPs and FNs. Larger βs lead to greater recall than precision, as more emphasis is placed on false negatives (SALEHI; ERDOGMUS; GHOLIPOUR, 2017). Salehi, Erdogmus e Gholipour (2017) demonstrated that using larger βs in the generalized loss function during training allows for greater generalization and better performance in cases of unbalanced data, giving more emphasis to FNs to improve recall.

5.3.2
Binary Cross Entropy

Binary Cross Entropy is used for binary classifications. In our work, we want to classify whether or not each pixel is a lesion, so it's an option that makes sense for the problem. It returns predictions that are either 0 or 1 (RUBY; YENDAPALLI, 2020). Below we explain how it is calculated.

The equation 5-6 represents the probability mass function, where y_i is the ground truth and can take values 1 or 0 as it is a binary classification (WATT; BORHANI; KATSAGGELOS, 2020).

$$p(y_i) = y_{pred}^{y_i}(1 - y_{pred})^{1-y_i} \tag{5-6}$$

The probability of a pixel belonging to the category "with lesion" or "without lesion" is shown in the equation 5-7 (WATT; BORHANI; KATSAGGELOS, 2020).

$$P(y_i) = \prod_{i=1}^{n} p(y_i) = \prod_{i=1}^{n} y_{pred}^{y_i}(1 - y_{pred})^{1-y_i} \tag{5-7}$$

Now we take the log of the expression 5-7, as this makes the calculation simpler and the result is in equation 5-8 (WATT; BORHANI; KATSAGGELOS, 2020).

$$\log P(y_i) = \sum_{i}^{n} [y_i \log(y_{pred}) + (1 - y_i) \log(1 - y_{pred})] \tag{5-8}$$

We want to maximize the expression 5-8, which is exactly the probability that a pixel belongs to the same class as y_i. On the other hand, we can minimize the expression $-\log P(y_i)$ below:

$$-\log P(y_i) = -\sum_{i}^{n} [y_i \log(y_{pred}) + (1 - y_i) \log(1 - y_{pred})] \tag{5-9}$$

The above equation represents the loss function itself. To calculate the average value of the loss function, we need to divide the value by the number of samples. So the final expression for the binary cross entropy (BCE) is given by the equation 5-10.

$$\text{BCE} = -\frac{1}{N} \sum_{i}^{n} [y_i \log y_{pred} + (1 - y_i) \log(1 - y_{pred})] \tag{5-10}$$

5.4 AdamW Optimizer

One of the main challenges in machine learning is the minimization of the loss function. The optimizer is responsible for changing network parameters (such as weights and learning rate) to minimize the loss function. Each optimizer has a different methodology to approach this problem and in this work we chose to use the optimizer proposed by Loshchilov e Hutter (2018), AdamW, an improved version of Adam that generalizes better and can compete with stochastic gradient descent, another optimizer.

5.4.1 Adam

First, let's understand how Adam works. To minimize a loss function f, you randomly initialize the weights of the network and the goal is to reach a minimum for the loss function as quickly as possible. Before each step, you need to define which direction has the greatest slope, that is, which direction has the greatest gradient ∇f. Thus, the weight $x(t)$ is updated as in the expression 5-11 (LOSHCHILOV; HUTTER, 2018).

$$x(t) = x(t - 1) - \alpha \nabla f \tag{5-11}$$

What the Adam optimizer does is adapt the steps according to how much the gradient changes. So it allows us to take bigger steps when the gradient doesn't change much, keeping step in the same order as α and to take smaller steps when it varies rapidly, making step size much smaller (LOSHCHILOV; HUTTER, 2018).

5.4.2 Weight decay

According to Loshchilov e Hutter (2018), networks with lower weights are less likely to overfit and are able to generalize better. Adding weight decay is a strategy to try to keep the weights smaller for better generalization.

The weight decay rate per step ω determines the relative importance of minimizing the original cost function (when the ω is small, $1-\omega$ is close to 1) and finding smaller weights (when the ω is very large, $1-\omega$ is close to 0). The weights update expression is shown in 5-12 (LOSHCHILOV; HUTTER, 2018).

$$x(t) = (1-\omega) \cdot x(t-1) - \alpha \nabla f \tag{5-12}$$

The term $(1-\omega)$ exponentially decays the weights x and thus forces the network to learn smaller weights.

Loshchilov e Hutter (2018) applied the weight decay strategy in the Adam optimizer and got the AdamW. They experimentally showed that the latter can generalize better than models trained with Adam and besides, it achieves results as interesting as the ones of the stochastic gradient descent with momentum, which is one of the most used by researchers.

6
Results

In the Table 6.1 we describe the distribution of samples from our dataset between training, validation and testing. As our dataset is small, with only 46 samples, we had to choose a larger K to perform the K-fold cross validation, otherwise we would have very few images for training and the model would not present a good result. We chose $K = 11$ and in each fold approximately 10% of the samples were left for testing and the rest was left for training and validation.

	Number of samples		
Fold Number	Train	Validation	Test
1	37 (80%)	5 (11%)	4 (9%)
2	37 (80%)	5 (11%)	4 (9%)
3	37 (80%)	5 (11%)	4 (9%)
4	37 (80%)	5 (11%)	4 (9%)
5	37 (80%)	5 (11%)	4 (9%)
6	37 (80%)	5 (11%)	4 (9%)
7	37 (80%)	5 (11%)	4 (9%)
8	37 (80%)	5 (11%)	4 (9%)
9	37 (80%)	5 (11%)	4 (9%)
10	36 (78%)	5 (11%)	5 (11%)
11	36 (78%)	5 (11%)	5 (11%)

Table 6.1: K-Fold description

After distributing the samples between training, validation and testing, we performed the training for each fold both with the Binary Cross Entropy loss function, whose results are in the Table 6.2, and with the Tversky loss function, whose results are in the Table 6.3.

Fold Number	Accuracy	Precision	Recall	F1 Score
1	98,0%	67,4%	86,2%	75,7%
2	98,9%	67,2%	79,3%	72,6%
3	99,4%	50,3%	78,4%	59,7%
4	98,1%	75,2%	86,6%	80,4%
5	98,4%	55,1%	80,2%	64,9%
6	98,5%	55,6%	76,5%	63,9%
7	98,9%	43,1%	60,4%	48,4%
8	98,3%	67,2%	79,8%	73,0%
9	98,4%	66,4%	62,3%	64,2%
10	98,4%	67,1%	87,0%	75,7%
11	97,8%	78,6%	81,9%	80,2%
Average	98,4%	66,8%	79,6%	68,7%

Table 6.2: Metrics results using the Binary Cross Entropy loss function

Some folds obtained better results than others. In the Table 6.2, the best fold was Fold 4, whose F1 Score reached 80.4%, while the worst was Fold 7. The difference between the results of each fold is expected, since the data separated for training is different in each case. Therefore, the overall performance of the model is given by the average of each metric, as shown in the last rows of the Tables 6.2 and 6.3. To facilitate the comparison, the general result of each model is in Table 6.4.

Fold Number	Accuracy	Precision	Recall	F1 Score
1	97,9%	82,6%	74,7%	78,4%
2	98,4%	82,6%	60,6%	69,8%
3	99,4%	71,8%	66,8%	68,9%
4	98,0%	78,4%	83,4%	80,8%
5	98,3%	61,5%	70,1%	64,3%
6	98,4%	74,3%	66,4%	69,6%
7	98,7%	71,7%	65,3%	66,4%
8	98,3%	76,9%	75,7%	76,2%
9	98,6%	79,8%	75,8%	77,5%
10	98,3%	83,5%	75,4%	79,2%
11	97,4%	85,6%	70,3%	76,8%
Average	98,4%	77,6%	72,4%	73,0%

Table 6.3: Metrics results using the Tversky loss function

Loss	Accuracy	Precision	Recall	F1 Score
Tversky	98,34%	78,37%	70,28%	76,23%
Binary	98,35%	77,61%	70,28%	73,03%

Table 6.4: Comparison of overall performance between models trained with Tversky and with Binary Cross Entropy

In the two models presented above, the accuracy is always very high with values above 90%. Since accuracy measures the ratio between the correct pixels and total pixels, this is expected, because our model correctly predicts almost every region of the brain.

Below we present some results that we obtained with the model trained using Binary Cross Entropy loss function and with the model trained using Tversky loss function. The two models made very similar predictions, but in some cases, as in the Figure 6.1, the prediction of the former is a little worse than the latter. Other results are in Figures 6.2 and 6.3.

In the figures below, Binary is the result generated by the model trained using the Binary Cross Entropy loss function and Tversky is the result generated by the model trained using the Tversky loss function.

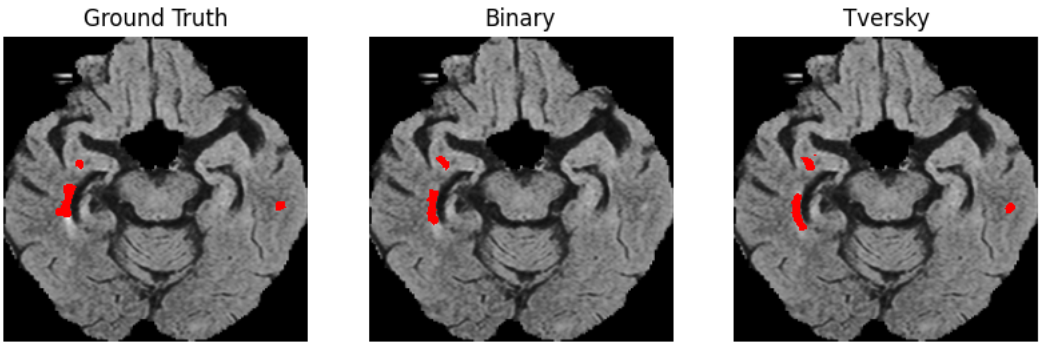


Figure 6.1: Result in which the model with the Binary Cross Entropy loss function is worse than the model with the Tversky loss function

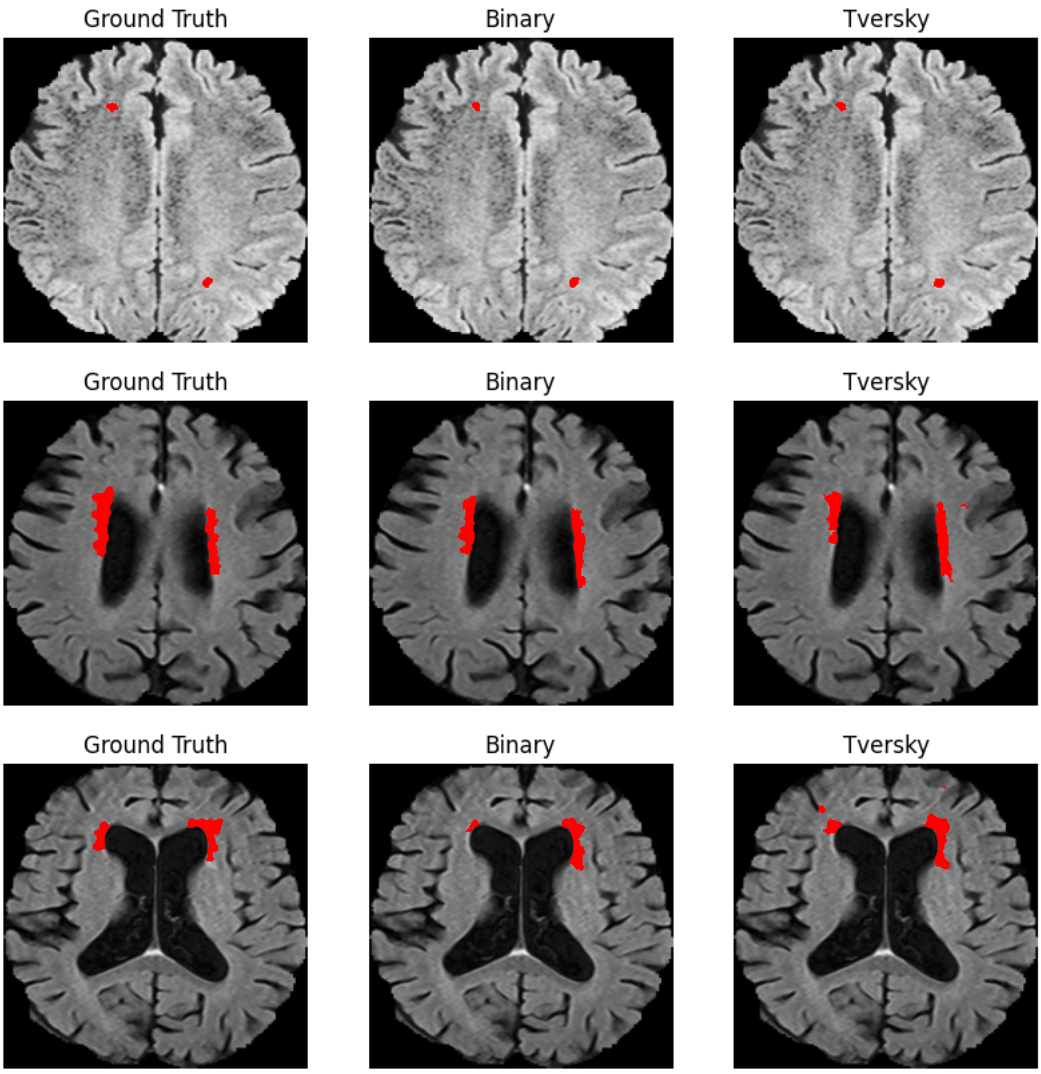


Figure 6.2: Some results comparing the two models - Part 1

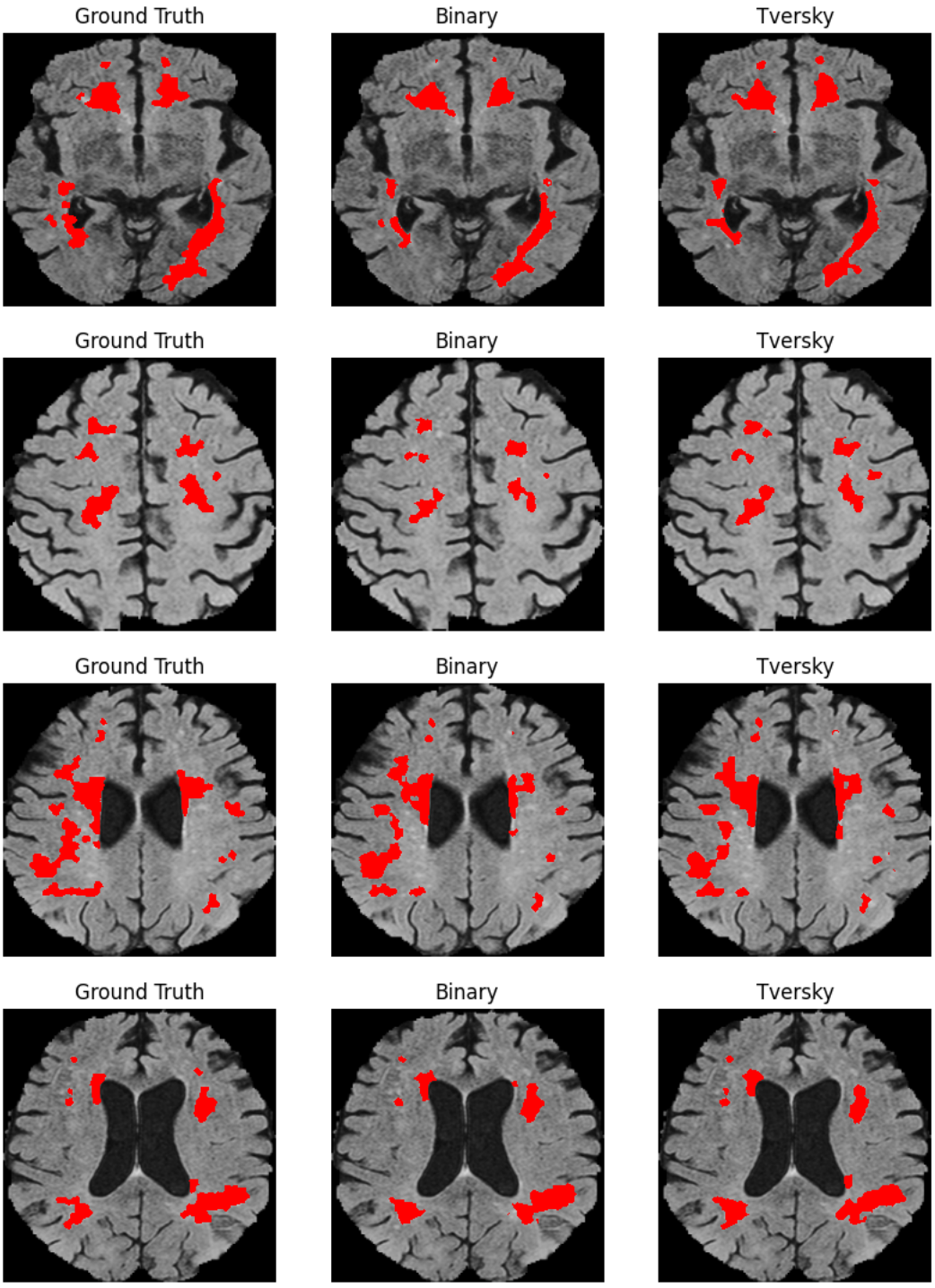


Figure 6.3: Some results comparing the two models - Part 2

7
Conclusion and Future Work

In this work we developed a deep learning model to perform lesion segmentation in brain images. We had a very small dataset and we had to take some actions to get around this problem, performing K-fold cross validation and choosing the DC-UNet architecture that handles this situation well. Also, we selected two loss functions to compare the results and choose the best one for our problem: Binary Cross Entropy and Tversky. The latter performed a little better than the former.

In addition, the pre-processing of the images was essential for the performance of the model, since the skull and the difference in intensity of the magnetic resonance images could prevent the neural network from extracting important features.

In general, our results were very encouraging and this indicates that, in the future, techniques such as the one applied in this work may be used by specialists to aid in the diagnosis of brain lesions. Furthermore, we still have about 500 data without masking and some techniques can take advantage of this data to improve the result. For example, as future work, we can use self-supervised learning techniques, such as Masked Autoencoder (ZHOU et al., 2022), to improve training performance and model generalization or perform finetune for models like U-NetR (HATAMIZADEH et al., 2022) or ViTDet (LI et al., 2022) to improve segmentation.

8
Bibliography

ANJOS, G. et al. Automatic segmentation of breakouts in acoustic borehole image logs using convolutional neural networks. **Proceedings of the XLIII Ibero-Latin-American Congress on Computational Methods in Engineering, ABMEC**, CILAMCE, 2022.

BAUER, S.; FEJES, T.; REYES, M. A skull-stripping filter for itk release 1 . 0. In: . [S.l.: s.n.], 2013.

BOER, R. D. et al. White matter lesion extension to automatic brain tissue segmentation on mri. **Neuroimage**, Elsevier, v. 45, n. 4, p. 1151–1161, 2009.

DONG, H. et al. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In: SPRINGER. **annual conference on medical image understanding and analysis**. [S.l.], 2017. p. 506–517.

DU, G. et al. Medical image segmentation based on u-net: A review. **Journal of Imaging Science and Technology**, Society for Imaging Science and Technology, v. 64, p. 1–12, 2020.

HATAMIZADEH, A. et al. Unetr: Transformers for 3d medical image segmentation. In: **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision**. [S.l.: s.n.], 2022. p. 574–584.

HENNIG, C.; KUTLUKAYA, M. Some thoughts about the design of loss functions. **REVSTAT-Statistical Journal**, v. 5, n. 1, p. 19–39, 2007.

IBTEHAZ, N.; RAHMAN, M. S. Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation. **Neural networks**, Elsevier, v. 121, p. 74–87, 2020.

LI, Y. et al. Exploring plain vision transformer backbones for object detection. **arXiv preprint arXiv:2203.16527**, 2022.

LOSHCHILOV, I.; HUTTER, F. Fixing weight decay regularization in adam. 2018.

LOU, A.; GUAN, S.; LOEW, M. Dc-unet: rethinking the u-net architecture with dual channel efficient cnn for medical image segmentation. In: SPIE. **Medical Imaging 2021: Image Processing**. [S.l.], 2021. v. 11596, p. 758–768.

RAMANI, R.; VANITHA, N. S.; VALARMATHY, S. The pre-processing techniques for breast cancer detection in mammography images. **International Journal of Image, Graphics and Signal Processing**, Modern Education and Computer Science Press, v. 5, n. 5, p. 47, 2013.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. **Encyclopedia of database systems**, Springer, v. 5, p. 532–538, 2009.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **International Conference on Medical image computing and computer-assisted intervention**. [S.l.], 2015. p. 234–241.

RUBY, U.; YENDAPALLI, V. Binary cross entropy with deep learning technique for image classification. **Int. J. Adv. Trends Comput. Sci. Eng**, v. 9, n. 10, 2020.

SALEHI, S. S. M.; ERDOGMUS, D.; GHOLIPOUR, A. Tversky loss function for image segmentation using 3d fully convolutional deep networks. In: SPRINGER. **International workshop on machine learning in medical imaging**. [S.l.], 2017. p. 379–387.

WANG, L. et al. Correction for variations in mri scanner sensitivity in brain studies with histogram matching. **Magnetic resonance in medicine**, Wiley Online Library, v. 39, n. 2, p. 322–327, 1998.

WATT, J.; BORHANI, R.; KATSAGGELOS, A. K. **Machine learning refined: Foundations, algorithms, and applications**. [S.l.]: Cambridge University Press, 2020.

ZHAO, H. et al. Loss functions for neural networks for image processing. **arXiv preprint arXiv:1511.08861**, 2015.

ZHOU, L. et al. Self pre-training with masked autoencoders for medical image analysis. **arXiv preprint arXiv:2203.05573**, 2022.