3D semantic segmentation using topographic information

Anne-Claire Fouchier, Hunor Laczkó

***Abstract*—Improvements of classification and localisation techniques in image processing has led to the refinement of the technique to a pixel level detection, referred to as segmentation. Semantic segmentation has become a substantial task in the field of computer vision. It is especially valuable in fields like biomedics and self-driving cars, where extracting objects of interest, like tumors and pedestrians, is critical. Several neural networks deal with this task. Their architectures diverge depending on their strategy, but their loss functions rely on standard measures such as cross-entropy, L1 loss, or dice coefficient loss. However, these measures do not take into account topographical properties; they handle every pixel in a similar manner, thus ignoring valuable information. Al-Afandi et al. [2] introduced the wave loss metric to include topographic properties in 2D images and have shown promising results. We further develop their method to apply it to point clouds and 3D images. To do so, we present a qualitative comparison of different metrics highlighting the improvements with the Waveloss. Our results show greater accuracy in terms of determining similarity between objects. With this, we illustrate the importance of topographical information and how this metric could be used to enhance the accuracy of existing networks.**

I. Introduction

Semantic segmentation is a prominent research field in computer vision. It is relevant in many applications ranging from autonomous driving to medical imaging. In images, it defines the process of classifying an image at pixel level, that is, assigning one of the predefined classes to each pixel in the image. In point clouds, it refers to assigning each voxel to a predefined class. To perform this operation, a large set methods utilise neural networks. A considerable amount of research tends to focus on presenting different network architectures to segment images. However, refining the process of evaluating the output produced by these networks is equally important to ensure a better model. Neural networks use loss functions during training and performance evaluation to determine the similarity of output generated by the network to the ground truth. Popular metrics found in literature to evaluate the segmentations are cross-entropy and dice loss. These methods are discussed in detail in Section V. However, these loss functions are rarely equipped to handle topographical information of the generated segmentation. In this report a novel loss function, waveloss, is presented for 3D images and point clouds. The metric is able to evaluate different aspects of the segmentation generated by the network including its shape and location of misclassified pixels. The evaluation of the proposed metric is carried out on 3D Brain MRI.

The objectives of this work are as follows:

anneclaire.fouchier@gmail.com, hunor.laczko@estudiante.uam.es

- Study of state of the art
- Proposing a metric considering topographical information
- Evaluating the metric to establish a proof of concept

The rest of the paper is organized as follows: Section II where the used tools are listed, Section III where other similar methods are discussed, Section IV where the new method is presented, Section V presents and discusses experimental results leading to a set of conclusions in section VI.

II. Preliminaries and Tools

A. Segmentation Method

Convolutional neural networks (CNNs) yield hierarchies of features extracted from input images. Originally, they were used for classification tasks. However, work by Long et al. [1] marked huge increase of popularity of CNNs in the area of semantic segmentation. Since then, CNNs have exceeded the state-of-the-art in semantic segmentation.

Another ground-breaking network architecture, U-Net, was proposed by Ronneberger et al. [6]. It is one of the most popular segmentation networks for medical imaging. The architecture is named so because its structure looks like a 'U'. There are three main components of the architecture: the contraction, the bottleneck and the expansion section. The contraction section is made of numerous blocks that take an input and apply convolution layers followed by max pooling. The last layer of the contraction mediates between the contraction layer and the expansion layer. It uses two CNN layers followed by up convolution layer. The heart of this architecture lies in the expansion section. It also consists of several expansion blocks which pass the input to two CNN layers followed by a upsampling layer. In each of the expansion block the input gets appended by feature maps of the corresponding contraction layer. This action ensures that the features that are learned while contracting the image will be used to reconstruct it. The number of expansion blocks is the same as the number of contraction blocks. After that, the resulting mapping passes through another CNN layer with the number of feature maps equal to the number of segments desired. The loss function used in the original paper is cross-entropy.

An improvement of the original architecture is proposed by Isensee et al. [5]. In this study, this modified U-Net is used to compare the performance of proposed loss function with other popular methods mentioned earlier in Section I. Further details about the network and corresponding dataset can be found in Section V-C.

B. Tools used

The study is conducted in two stages. The tools used for them are listed as following:

1) Comparing Point Clouds: 3DsMax and *Meshlab* are used to generate and convert point clouds in the required formats for comparison using waveloss. These comparisons are done with PyCharm IDE using Python 3. The main libraries used are: *numpy, scipy, SimpleITK, nibabel, point_cloud_utils*.

2) Training Neural Network: Python 3 is used to adapt and train the modified U-Net network architecture with the proposed metric. *Google Colaboratory* is used to train these networks as it offers a GPU environment which helps to significantly improve the training time. To develop and train the model, the high level *Keras API* is used with Tensorflow backend. Moreover, a few functions are also implemented in *Tensorflow*.

III. Related work

As discussed earlier, loss functions are used to attribute a cost to the difference between the network's output and the ground truth. It helps to simplify the problem representation and determines how well a specific algorithm models the given data. Further, it also aids in optimizing the network's performance.

Al-Afandi et al. [2] expose some deficiencies in the methods used in the 2D segmentation tasks. It is shown that most loss functions used are pixel-based. However, the authors argue that topographical information such as, the shape of segmentation and relative position of the misclassified pixels, should also be taken into consideration as they contain valuable information about the segmentation. To accomplish this task, they have introduced their own metric called 'Waveloss'. It is an extension of other pixel-base approaches. With this metric, both intensities and spatial differences are penalized through several parameters. First, intensity-based differences and then the topography-based differences are computed. Finally, both intensity and topographical differences are weighted by different penalties. The parameters can be tuned to amplify one characteristic over another one. For example, larger penalties can be applied to later iterations which ensure that the misclassified pixels farther away from the ground truth get higher penalty than the misclassified pixels closer to the ground truth. Further, this measure is flexible and with specific characterization, it can take the form of cross entropy or L1. The authors have effectively managed to show that topographic information can help increase the accuracy of commonly used image segmentation networks and provide faster convergence on a simple dataset inspired by CLEVR. They also managed to increase the overall accuracy of instance segmentation by 4% on the more complex dataset MS-COCO(2017) using the Mask_RCNN architecture with ResNet-101 backbone by changing the Smooth-L1 loss with the Waveloss.

In 3D image segmentation, the commonly used loss functions usually do not take topography into account. Two of the most commonly used ones are: Binary Cross Entropy and the Dice Coefficient Loss. They are described further in Section V, as well as their advantages and disadvantages.

Our work is inspired from [2]. A similar technique is proposed in this project to apply the Waveloss on 3D images and point clouds.

IV. Proposed approach

The aim of this paper is to investigate the use of topographical information through the Waveloss function introduced by Al-Afandi et al. [2] for the task of 3D image segmentation. To do so, the Waveloss was implemented to be functional on 3D objects. The Waveloss implementation can be found in Algorithm 1.

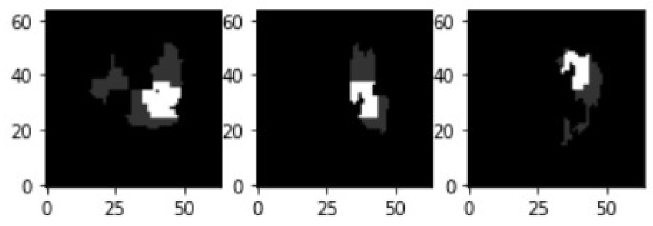
```
Data: image1, image2
Result: Waveloss
begin
    $ValInc \leftarrow 0.1$
    $SpaInc \leftarrow 10$
    $NumSteps \leftarrow int(1/ValInc)$
    $ValW \leftarrow np.arange(1, NumSteps + 1)/NumSteps$
    $SpaW \leftarrow np.arange(1, NumSteps+1)/NumSteps$
    $WaveLoss \leftarrow 0$
    $IntersSection \leftarrow min(image1, image2)$
    $Union \leftarrow max(image1, image2)$
    $CurrentWave \leftarrow min(image1, image2)$
    for $step \in range(NumSteps)$ do
        # Loss for intensity difference:
        $Growed \leftarrow CurrentWave + ValInc$
        $Growed \leftarrow min(Growed, Union)$
        $ValueDiff \leftarrow sum(Growed - CurrentWave)$
        # Spatial Propagation:
        $Growed \leftarrow$
          $max\_pool3d(Growed, SpaInc, 1, padding =' SAME')$
        $Growed \leftarrow min(Growed, Union)$
        $TopologyDiff \leftarrow$
          $sum(Growed - CurrentWave)$
        $CurrentWave \leftarrow Growed$
        $WaveLoss \leftarrow WaveLoss + ValW[step] *$
          $ValueDiff + SpaW[step] * TopologyDiff$
    end
end
```

Algorithm 1: Waveloss

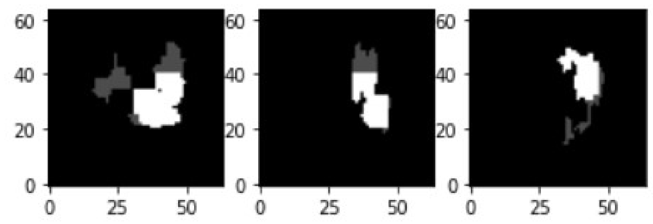
This function takes two point clouds, the output of the network and the ground truth. Here, both intensities and spatial differences are penalized through four parameters describes as follows:

1) *ValInc* is used to take intensity-based differences by determining the speed of wave during propagation along the intensity differences, number of iterations and therefore, the largest distance from the intersection where topographical differences are to be considered.
2) *SpaInc* is used to take topography-based differences, by determining the spatial propagation speed of the wave.
3) *ValW* contain the penalties for intensity differences.
4) *SpaW* contain the penalties for topographical differences.

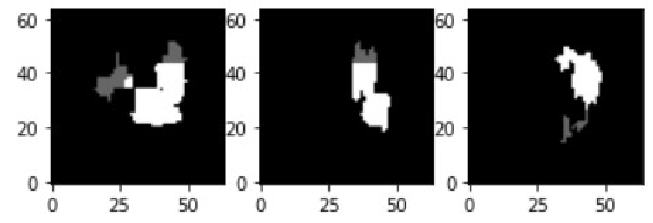
These parameters can be tuned to amplify one characteristic over another one.



(a) Iteration 1



(b) Iteration 2



(c) Iteration 3

Fig. 1: Illustration of the wave propagation

This function can be defined as the volume of an ascending wave starting from the intersection of the two segmentations and filling out the area defined by the union of the segmentations. A demonstration of this propagating wave can be seen in Figure 1. The images show cross-sections along the three axis of a tumor segmentations in three iteration steps. It can be seen that the 'wave' (white pixels) spread further from their initial position, this is the spatial propagation. The gray areas show the union of the segmentations. It can be seen that the intensity of the gray area becomes higher in each iteration, representing the intensity or value propagation of the waveloss.

This spatial and value propagation can be defined with the following equation:

$$Waveloss(A, B) = \int_{(x,y,z \in D_{Hm}(A,B))} D_{Hs}(x, y, z)$$

Where A, B are the input binary 3D images, *D_Hm* and *D_Hd* are the Hamming and Hausdorff distances defined in Section V-A. The equation represents the point-wise integration of local Hausdorff distances over every point in the disjunctive union of the two objects. This way it combines both the Hamming and Hausdorff distances since the maximal height of the propagating wave is proportional to the value of the Hausdorff metric for connected objects and the area of the wave is proportional to the Hamming distance. The slope which is the increase of the wave at each iteration, determines the connection between the topographic (Hausdorff) and the area based (Hamming) information.

The investigation is planned as follows:

- Create specific situations, evaluate the waveloss performance on them and compare the results with commonly used measures in order to highlight the shortcomings of the previously mentioned metrics (Section V-B)
- Train U-Nets on a real-life dataset with commonly used metrics as well as the waveloss and evaluate the results (Section V-C.)

V. Experimental study

In this section a comparison of the proposed loss function is presented with other commonly used loss functions. Most of these measures compute a pixel-wise difference between two objects disregarding their topographic properties. Through these comparisons, it is shown how these topographic properties can be leveraged to create a new loss function. Firstly, certain specific cases of point clouds are presented to demonstrate the behaviour of the proposed method. In the second part the performance of the described approach is presented in a real-world scenario and compared with other commonly used loss functions.

The loss function used for a specific application is always dependent on the problem itself. In case of 3D segmentation, three dimensional objects are compared to each other. Since they possess differences in structure, using the information about their shapes can be advantageous.

A. Loss functions used for comparison

This section presents the commonly used similarity measures used in recent works. These functions are used in this section for comparison. In the following sections the two 3D input images will be noted with A and B. They are binary images only containing ones and zeros.

1) Hamming distance: Hamming distance is a widely used distance measure and can be applied to a variety of problems in applications other than image processing as well. It was first used for developing error correction codes in 1950 [3]. In the domain of images, it determines the number of differing pixels, and can be computed as follows:

$$D_{Hm} = \sum (A \cup B)/(A \cap B)$$

The advantage of this measure is that it is easy and fast to calculate. However, it cannot differentiate topographic changes of an object. It only considers the volume difference between two objects. Further, it does not take into account the position of these differences either.

2) Hausdorff distance: The Hausdorff distance, on the other hand, only looks at the topographic differences between the two objects. It is computed as follows:

$$D_{Hd} = max(h(A, B), h(B, A))$$

where

$$h(A, B) = max_{a \in A} min_{b \in B} distance(a, b)$$

As it can be seen from the equations, the Hausdorff distance determines the largest difference between two pixels. There are two main problems with this method. First, by choosing only the largest difference it hides all other smaller differences without considering them at all. Secondly, it is sensitive to noise and outliers which makes it hard to use in real-life scenarios.

3) Binary cross entropy: Cross entropy is a popular choice for loss function, it measures the similarity between two vectors or distributions. In a typical multiclass classification application it is used to measure the similarity between the output of a softmax layer (the probability of object belonging to each class) and the one-hot-encoded ground truth label.

In this application for the binary segmentation a simplified version is used, called the binary cross entropy which is given by the following equation:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}(A_i \log B_i + (1 - A_i)\log(1 - B_i))$$

where N is the number of pixels in the images.

While this is a good general-purpose similarity measure it still does not consider the topography of the compared objects.

4) Dice coefficient loss: Dice coefficient is also widely used nowadays as it can be applied to a wide variety of problems. By default dice coefficient is an overlap measure which gives a value from the range [0,1] where 0 means no overlap and 1 means total overlap. In this form, it is used as a loss function to evaluate the performance of a segmentation for instance. It can easily be turned into a loss function by using $1 - dice$. As a loss function, the dice coefficient can be defined as follows:

$$Dl = 1 - \frac{\sum A_i B_i}{\sum A_i + \sum B_i}$$

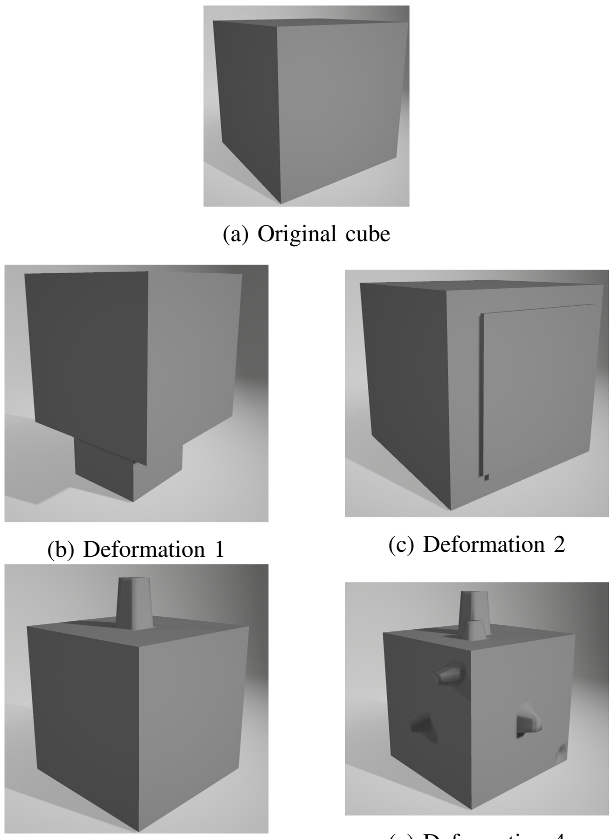
Once again, this measure does not take topography into account. In later examples it can be seen that in specific situations it might even perform inversely than expected.

B. Behaviour of waveloss compared to other measures

In order to demonstrate the behaviour and shortcomings of the above mentioned measures a few edge cases will be presented. For these, specific objects were created with which two sets of comparisons were conducted. In each, two deformed objects were compared to the original one and the results of these were evaluated.

1) Objects used for comparison: The base object is a simple cube (Figure. 2a). It is worth to mention that in this case it was a hollow cube to simulate a more point cloud like object, but the loss function should work the same with dense objects as well.

The first two deformations were designed in a way that they differ in the same number of voxels from the original cube. The distribution of these differing pixels is different. In the first deformation (Figure. 2b) these voxels are concentrated on a smaller area and they also stand out more, meaning they are farther from the center of the cube. In the second deformation (Figure. 2c) the same number of pixels is spread on a wider area and they are closer to the center than the previous deformation.


(a) Original cube

(b) Deformation 1

(c) Deformation 2

(d) Deformation 3

(e) Deformation 4

Fig. 2: Deformed cubes used for the comparison

	cube_1	cube_2	cube_3	cube_4
Waveloss	453.08	315.0	72.08	244.02
Dice	0.00306	0.00577	0.00126	0.00505
BCE	85.73023	85.73022	21.43256	90.92601
Hausdorff	1846.12012	6.49740	2504.73224	2504.732237
Hamming	1056.0	1056.0	264.0	1120.0

TABLE I: Comparison results

For the second set of comparisons two other deformations were created. Deformation 3 (Figure. 2d) contains a single protrusion compared to the original cube. Deformation 4 (Figure. 2e) contains that same protrusion along with multiple smaller ones. All of the smaller protrusion's furthest voxel is closer to the cube's center than the furthest voxel of the original protrusion.

	cube_1	cube_2	cube_3	cube_4
Waveloss	1	0.63759	0	0.45129
Dice	0.39930	1	0	0.84095
BCE	0.92523	0.92523	0	1
Hausdorff	0.73637	0	1	1
Hamming	0.92523	0.92523	0	1

TABLE II: Normalized comparison results

	cube_1	cube_2	cube_3	cube_4
Waveloss	1.43835	1	1	3.38541
Dice	0.53057	1	1	4.00722
BCE	1	1	1	4.24242
Hausdorff	284.13209	1	1	1
Hamming	1	1	1	4.24242

TABLE III: Ratio of change between deformations

2) Comparison results: It is worth noting that the Waveloss function is highly customizable and could be tuned to better handle these specific cases, but for generality these comparisons were made with simple parameters, meaning linear weights, a value step of 0.1 and spatial step of 10.

In the first set of comparisons, the original cube was compared with the first two deformed versions of the cube referred to as *cube_1* and *cube_2* in the result tables. For the analysis, *cube_2* is considered more similar to the original cube than *cube_1* as the deviation of the changed area is small. This is visually intuitive as well. The results of these comparisons can be seen in Table I. It is generally true for all these metrics that the smaller the value, the more similar the compared objects are, but they all operate in different domains, thus they were normalized. The normalized values can be seen in Table II. In this table it can be easily seen that the *Hamming* distance in both cases is the same. This was expected as, by definition, the Hamming distance represents the number of differing voxels, and the objects were created so that these results would be equal. *Binary Cross Entropy* also acts in a similar way, it cannot differentiate between the two cases. While the normalized values already contain valuable information, it is worth investigating further and look at the rate the values of each metric change. The ratio of change of the values of comparison with *cube_1* compared to *cube_2* can be seen in Table III. Once again, *Hamming* and *BCE* show no change, but what is interesting is that the *Hausdorff* distance shows orders of magnitude of higher change than the other metrics. While the change indeed is large, this value is still unexpected. The *Dice Coefficient* acts in an unexpected way too, it gives a lower value for the first deformation compared to the second one, contradicting the intuition that it should be the other way around. This case shows the advantage of the *Waveloss* function which is able to differentiate between the two cases, has an appropriately proportionate response and the values reflect the real-world situation.

The second set of comparisons was created with *cube_3* and *cube_4* corresponding to deformation 3 and 4. From the normalized results in Table II, it can be seen that the *Hausdorff* distance cannot differentiate between the two cases since the maximum deviation from the center is the same in both cases. It is interesting to see that all other metrics classified *cube_3* as the most similar to the original one, while *Hausdorff* considers it the most different one. *Cube_4* on the other hand is considered the most different one by *Hamming* and *BCE* too, while *Dice* and *Waveloss* ranks it differently. This will be further investigated in the next paragraph. Looking and the ratio changes in Table III, it can be seen that except for the *Hausdorff* distance, all other metrics have a similar change in their values, although *Waveloss* is somewhat smaller than the others. At this point, it is hard to decide whether this is advantageous, but it shows that it can handle this case as well.

Investigating the normalized results a bit further, it can be seen that while *Hamming*, *Hausdorff* and *BCE* results are concentrated on one end of the interval *Dice* and *Waveloss* give more evenly distributed results. And while the last two do not give the same order across the four comparisons both their results could be argued to be good since it is hard to give an objective order between these four cubes.

C. Application of waveloss in real-world scenarios

After seeing that the waveloss function can handle scenarios that other loss function might not, it was also tested in real-world application. For this a CNN was trained for semantic segmentation of brain tumors using the proposed function along with two other loss functions: Binary Cross Entropy and Weighted Dice Coefficient. The Hamming and Hausdorff distances proved highly unreliable in these situations so they were not evaluated further.

1) Network: Modified U-Net: U-Net is one of the most popular segmentation networks for medical imaging. It was first used for 2D segmentation [6] and later extended for 3D segmentation as well. As such a popular network it was used as a starting point for several other researches. For this experiment a modified version was used as presented in [5]. This network reached the 3rd place in the 2017 BraTS Challenge. It modifies several parts of the original network, but the main differences include an equally weighted dice coefficient, residual weights, and deep supervision. The implementation used was taken from the following open source repository: [8]. Its architecture can be seen in Figure 3.

2) Dataset: BraTS2018: The dataset used was taken from the MICCAI BraTS'18 Challenge [9] [10] [11], which aimed to evaluate state-of-the-art methods for brain tumor segmentation in multimodal magnetic resonance imaging (MRI) scans focusing on the segmentation of intrinsically heterogeneous brain tumors, gliomas specifically.

The dataset contains 285 scans each scan has four variants which are the four input channels: native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2) and T2 Fluid Attenuated Inversion Recovery (FLAIR) volumes. For each scan a three-channel segmentation is provided, the annotations containing the GD-enhancing tumor, the peritumoral edema, and the necrotic and non-enhancing tumor core.

The dataset was divided into training, validation and testing set, each containing 230, 26 and 29 scans respectively. The network was trained using the training set and it was evaluated in each epoch using the validation set. At the end, the resulting network was evaluated on the test set. The results will be detailed in in a following section.

Since at this point reaching the best results was not the focus but to rather just do comparisons, a few steps were simplified, like the preprocessing of the data. This way the images were rescaled to a smaller dimension of size (4, 64, 64, 64) and they were normalized to have zero mean and standard deviation of

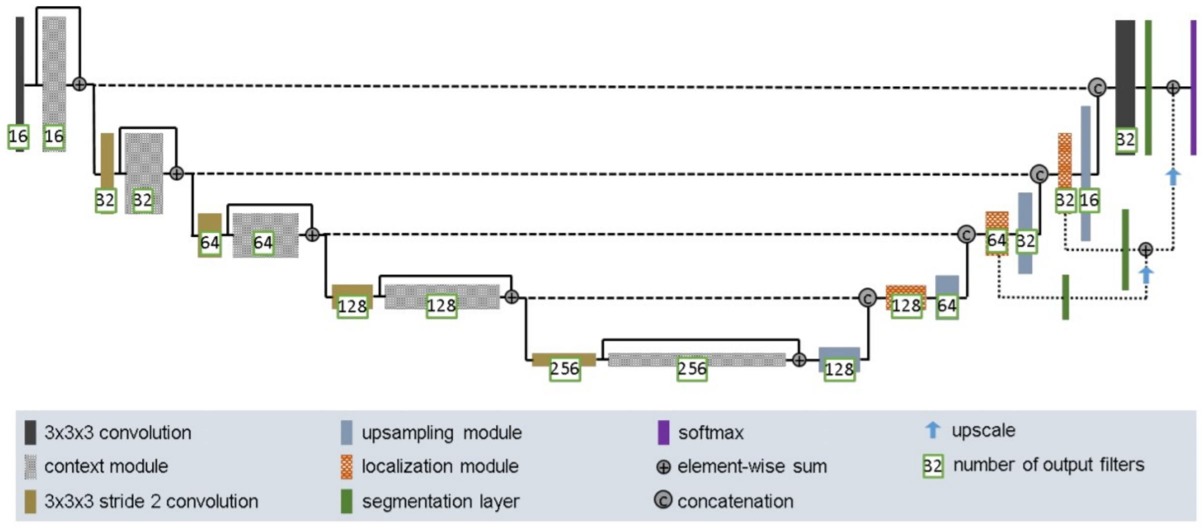


Fig. 3: Modified U-Net architecture proposed by Isensee et al. [5]

one. The same resizing was applied to the segmentation masks as well, along a morphological closing operation to correct some of the information lost due to resizing.

3) Parameters of the comparison: All the networks were trained using the same parameters. This means that the network itself was not changed at all, the same optimizer was used in all cases, which was the Adam optimizer with a constant learning rate of $5 \cdot 10^{-5}$. The same training, validation and test split of the data was used in all cases. All models were trained for 200 epochs using a batch size of 1. In all cases only the loss function was changed.

4) Results: In this section the result of training the above mentioned network will be discussed. Also, as recommended by the original paper of the network, instead of dice loss, a weighted dice-loss was to deal with the unbalanced nature of the data, meaning that in the segmentations there are significantly more background voxels than foreground. For the performance evaluation of the networks, the dice coefficient was used. The results on all the three splits of the data will be discussed but it is worth mentioning that the evaluation on the test dataset best reflects the performance of the given network. The results for these training can be found in Table IV. In the table, *max* refers to the maximum value from all epochs, *best* refers to the value from the best epoch (the epoch with the minimum validation loss is considered the best epoch) and *last* refers to the last epoch of the training. The *best epoch* column shows which epoch had the best validation loss out of the 200.

In a general overview it can be seen that all three functions result in a similar behaviour, as showed in Figures 4, 5, 6. In all cases, the dice coefficient converges in the first quarter of the training and reaches around the same values. Weighted dice converges (Figure 5b.) the fastest but this is highly dependent on the learning rate, so with other learning rates the other methods might converge this fast too. The loss values act similarly too. The waveloss shows some sudden changes in both the loss and the metric graphs (Figure 4.). This could be the result of the parameters used for the optimizer or for the loss itself, but it did not affect the overall performance. The loss for the weighted dice (Figure 5a.) shows that the learning rate was most suited for this function. In the graph representing the loss of the BCE (Figure 6a.), it can be seen that around the 50th epoch the validation loss starts to slightly increase indicating that it might be overfitting. While this increase is slow, it did result in worse performance in several cases compared to the other two. The other two models after starting to converge to their final validation dice values keep increasing the training values without decreasing their corresponding validation values. As it can be seen in the table in most of the cases this results in a small increase in performance compared to the best epoch.

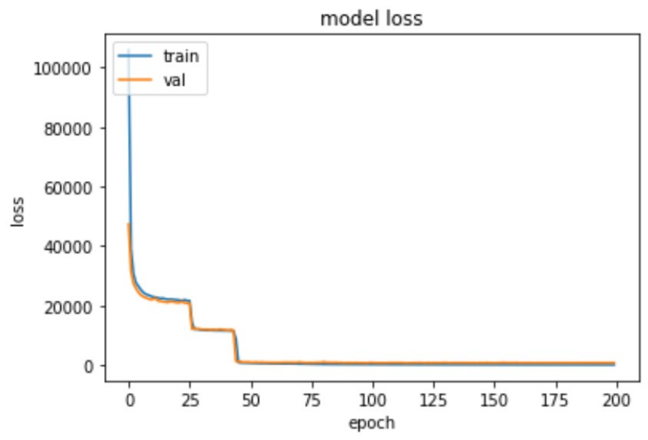
Looking at the values regarding the evaluation on the training set (Table IV.) it can be seen that all three methods reached very similar values with the exception of BCE when evaluating the best epoch. Here it shows a significantly lower value, which is probably due to the fact mentioned above that after reaching its peak in the validation loss the model is not able to improve the loss even though it improves the dice coefficient. This effect can be seen in the other splits of the data as well. This might mean that the validation loss is not the best way to determine the best epoch, but in all the other results the best epoch values are close to the maximum and the last ones indicating that the loss and the dice metric follow each other thus it should be a good value to determine the best epoch.

Evaluation on the validation split of the data is somewhat more representative. These results are closer to real-world performance of the models. Both Waveloss and WDice perform similarly, with WDice having a small advantage, but only one percent. BCE is the last in this case with 5-6 percent lower performance compared to the other two and an even higher difference in the best epoch case, but this is again due to the facts mentioned above.

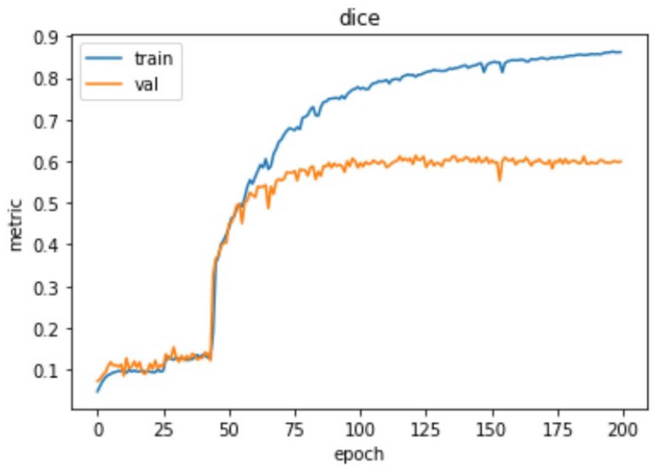
Results on the test set are the most representative ones. The model was only run once on these images after the

	Train			Validation			Test		
	max	last	best	max	last	best	best	last	best epoch
wave	0.863	0.862	0.824	0.614	0.600	0.612	0.633	0.641	136
wdice	0.854	0.852	0.825	0.622	0.613	0.622	0.618	0.618	119
bce	0.871	0.867	0.618	0.565	0.552	0.513	0.536	0.609	43

TABLE IV: Dice coefficients for the corresponding loss functions used for training

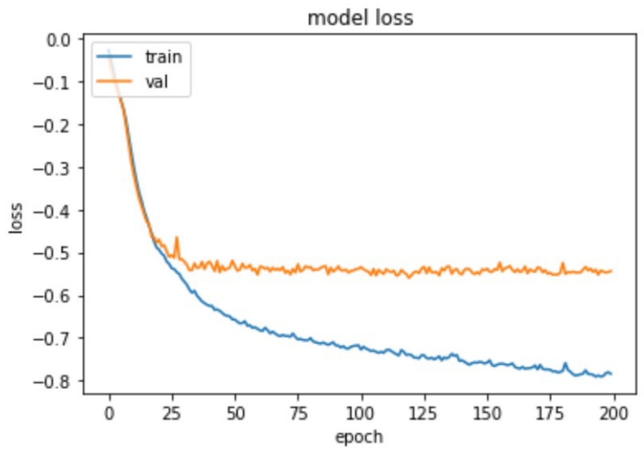


(a) Training and validation loss

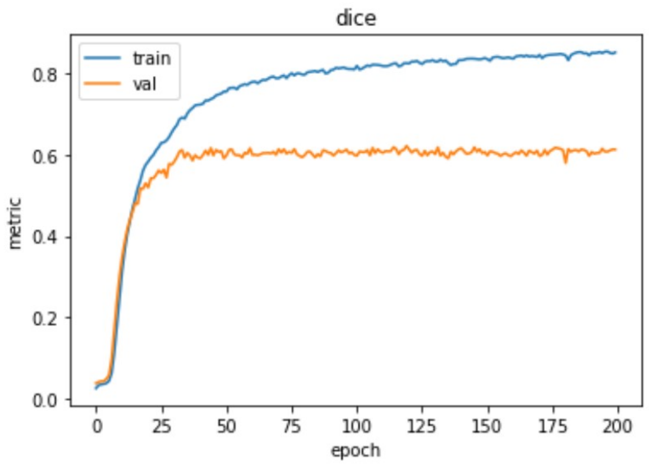


(b) Training and validation dice metric

Fig. 4: Training and validation graphs for Waveloss



(a) Training and validation loss



(b) Training and validation dice metric

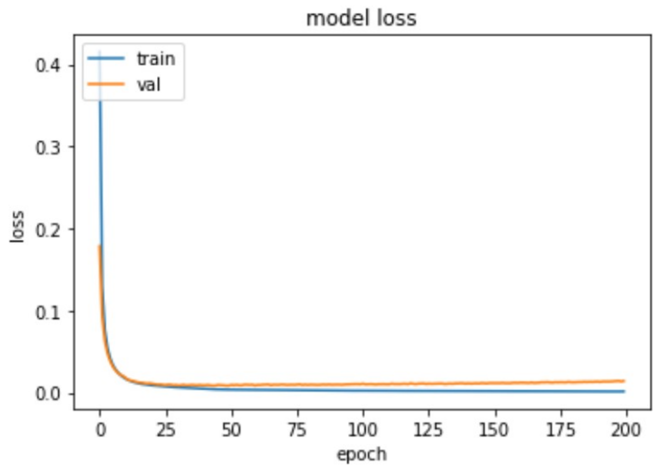
Fig. 5: Training and validation graphs for Weighted Dice

whole training process was finished. Once again, the best value for BCE is significantly lower than all the others, but the last epoch performance is comparable to them, even though it still comes as the worst performance. WDice has the same performance for best and last and is about the same as the validation performance. Waveloss shows increase in performance compared to the validation performance and this is enough to overtake the WDice that has better performance of about 3 percent than the waveloss.

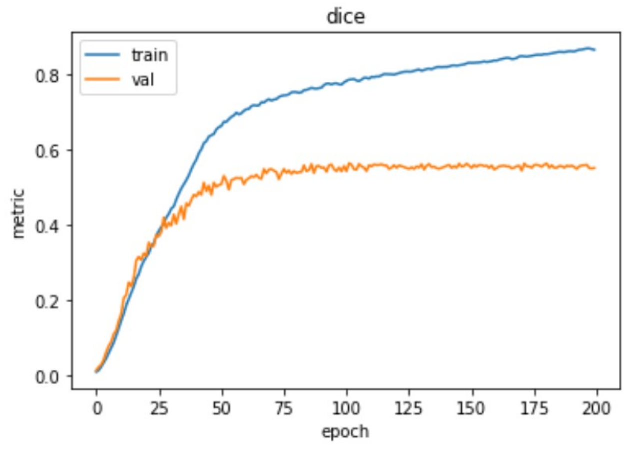
It has to be noted that while the Waveloss does outperform the other loss functions, it also adds to the computational cost of the overall training. However, the inference cost remains unaffected. While this is something to pay attention to, this effect can be reduced thanks to the many parametrization options of the function. For instance, by increasing the value increase at each step fewer iterations are required to calculate the loss, although might lose some accuracy. It all comes down to the problem-in-hand since it can be tuned according to the specific case. An example could be the case of pedestrian segmentation in a point cloud. In this case propagation along the z axis might be less important since in most of the cases they will have similar height.

VI. Conclusions

The Waveloss shows promising results on 3D brain MRI segmentation. The suggested approach is viable and can possibly outperform state-of-the-art methods. Performance and evaluation of qualitative and quantitative experiments confirm



(a) Training and validation loss



(b) Training and validation dice metric

Fig. 6: Training and validation graphs for Binary Cross Entropy

the viability of the waveloss as loss function for semantic segmentation of 3D images. The proof of concept is therefore validated.

In future work, several aspects can be explored further. A possible approach would be to achieve state-of-the-art segmentation results on the BraTS2018 or similar dataset. This would entail training a more complex network, without data simplification and fine tuning the parameters of the waveloss. Another option would entail applying the waveloss on a complex point cloud dataset, like the Lyft [12] dataset. This could provide more information about the capabilities of the waveloss. For better evaluation, the networks should be evaluated with more metrics besides the dice coefficient, like the meanIoU or Hausdorff95. Also, although reaching better segmentation performance is the main focus, the computational performance of the loss function can be evaluated and possibly optimized. These are possible options to continue this work and they will be explored in the next part of the project.

ACKNOWLEDGMENT

We would like to thank our supervisors András Horváth and Álvaro García Martinez for their help, insightful comments, explanations and recommendations. Their help and guidance were much appreciated throughout the whole project.

REFERENCES

[1] Long, J., Shelhamer, E., and Darrell, T. (2015). "Fully convolutional networks for semantic segmentation". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440)

[2] J. Al-Afandi and A. Horváth, "Application of the Nonlinear Wave Metric for Image Segmentation in Neural Networks", CNNA 2018; The 16th International Workshop on Cellular Nanoscale Networks and their Applications

[3] R.W.Hamming, "Error detecting and error correcting codes," Bell System technical journal, vol. 29, no. 2, pp. 147–160, 1950.

[4] F. Hausdorff "Grundzüge der Mengenlehre", 1914

[5] Isensee, F., Kickingereder, P., Wick, W., Bendszus, M., Maier-Hein, K.H.: Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge. BrainLes 2017, Springer LNCS 10670 (2018) 287– 297

[6] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: Med Image Comput Comput Assist Interv. Vol. 9351 of Lect Notes Comput Sci. pp. 234–241.

[7] Cicek, O., Abdulkadir, A., Lienkamp, S. S., Brox, T., Ronneberger, O., 2016. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In: Med Image Comput Comput Assist Interv. Vol. 9901 of Lect Notes Comput Sci. Springer, pp. 424– 432

[8] github.com/ellisdg/3DUnetCNN/blob/master/unet3d/model/isensee2017.py

[9] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694

[10] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[11] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[12] Kesten, R. and Usman, M. and Houston, J. and Pandya, T. and Nadhamuni, K. and Ferreira, A. and Yuan, M. and Low, B. and Jain, A. and Ondruska, P. and Omari, S. and Shah, S. and Kulkarni, A. and Kazakova, A. and Tao, C. and Platinsky, L. and Jiang, W. and Shet, V., Lyft Level 5 Perception Dataset 2020, https://level5.lyft.com/dataset/