INFO8010: Image segmentation to identify brain tumors

Rompen Jade,[1] **Bellefroid Sarah**,[2] and **Gómez Herrera María Andrea Liliana**[3]
[1]*Jade.rompen@student.uliege.be (s172422)*
[2]*sarah.bellefroid@student.uliege.be (s154557)*
[3]*mariaandrealiliana.gomezherrera@student.uliege.be (s198387)*

I. INTRODUCTION

Image segmentation refers to the task of breaking down an image into different semantic categories, these fragments are its meaningful elements, in order to process the image and allows to understand the boundaries and shapes of the objects contained. Image segmentation aims to label each pixel of an image with the corresponding class that represents. One of the most important applications of image segmentation is medical image diagnosis, where tissues are classified in a fastest way and have shown great results. Other applications of image segmentation are: geo-sensing, self-driving cars and facial segmentation.

In short, image segmentation works as follows: an image is given as input and a segmentation map will be given as output, this mask pixels are going to be labeled and indicate which class or instance every pixel belongs to. In order to get an overview of the predicted classes, the target vectors can be overlapped into the original image, this allows to identify the different regions in the image.

In this project the image segments are processed to identify tumors on brain images. When performing image segmentation the typical architectures of neural networks used are based on an encoder-decoder architecture, where the encoder extracts the features of the image and the decoder generates the segmentation map, an U-Net architecture has been chosen since it was primarily developed for medical image analysis and has been proved to be of good utility when performing segmentation tasks in medical imaging and nowadays is widely used while working with CT scans and MRI and X-rays. "This neural network is particularly useful since it creates highly detailed segmentation maps using very limited trading samples and it is faster to train than other segmentation models given to its context-based learning"[8].

We have used the "Br35H :: Brain Tumor Detection 2020" dataset that allows image classification, in brain images that either contain a tumor or not, and image segmentation. Once that the model based in an U-net is trained, we have choose to evaluate it with the following metrics: the precision, the recall, the Dice score and the Intersection Over Union score. Regarding the results, we have that the chosen architecture performs well for compact tumors and with a simple contour, although if the tumor's edges are complex and the tumor is not condensed in the same region of the brain, the results are different.

II. RELATED WORK

Medical diagnosis and treatment have benefited from medical imaging, image segmentation have improved the detection, diagnosis and monitoring procedures in diagnostic medicine and clinical oncology.

In order to ameliorate the medical diagnosis and treatment of tumors, a great variety of image segmentation methods have been developed. Starting from the use of a typical convolutional neural network, since pattern recognition is an innate feature of this kind of neural networks, to some methods more sophisticated like the two-path CNN that is build by passing the input through two streams in the network, where the first one stream has small kernels to get local information from the image and the second one has kernels with large receptive field, focusing more on contextual information or the three-path CNN that contains three layers in parallel, which are combined using a convolution layer, The parallel paths use different convolution kernels detecting multitude of features [12].

Another method is the usage of a MobileNet that is a depthwise deparable convolution neural network algorithm which is an improvement of a convolution neural network as it has a greater number of filters to detect the patterns in the magnetic resonance image (MRI) [1].

The paper "U-Net: Convolutional Networks for Biomedical Image Segmentation" [11] states that U-Net architectures provide more precise segmentation with few images than most architectures do. U-Net are composed of a sequence of encoder blocks linked together with max pooling layers then a sequence of decoder blocks. Down-sampling happens on the encoder side while up-sampling happens on the decoder side.

There exists quite a few loss functions that are used to train networks used to do image segmentation. [17] compares the results of the training of networks trained with different loss functions. Cross entropy loss is one of the most widely used loss function in deep learning and can be used in image segmentation problems. Focal loss is a variant of the binary cross entropy loss that addresses the issue of class imbalance. Dice loss is the most commonly used metric for evaluating segmentation accuracy. Tverski loss is related to dice loss but enables optimisation for output imbalance. Models are evaluated with four metrics: dice score, intersection over union, precision and recall.

III. METHODS

A. Preparing the data set

Before starting anything we needed to find a proper data set to work with. A proper data set for the task of image segmentation needs to contain: i) the images on which to perform the segmentation, ii) a groundtruth mask or some annotations that would lead to building it.

The data set we are working with is called *"Br35H::Brain Tumor Detection 2020"* and can be found on the Kaggle platform [5]. This data set is made up of two separate data sets, one allows for image segmentation and the other for tumor detection. For the segmentation part we are provided with three separate folders, each being respectively for the task of training, validating and testing the model. There are 500 images for training, 200 for validating and 100 for testing. The ground-truth mask are not provided but instead the regions of interests for all images are provided as annotations and are gathered in a JSON file. We use the library OpenCV to transform those annotations into the groundtruth masks. Those masks are binary mask, where the pixels labelled as 0 correspond to background or healthy tissues and those labelled as 1 correspond to tumorous tissues. The annotations are delineations of the tumour bounds. The regions can be recorded as polygons, eclipses and also as circles.

Before using the network the images are converted to gray-scale and resized to match a size of 512 x 512. This size was chosen because the model we are using requires a multiple of 32 and works best with a square image. We chose the largest size we could use with our resources to have smaller scaling error effects at the end. We also tried to first pad the image to a square then resize it to keep the aspect ratio. However, no major improvements were observed thus we only used the resize function to easier back transformations later.

B. Building the model

Originally neural convolution networks are good at predicting images but only if the same class is assigned to all pixels in the image. However here the aim is to perform pixel-based classification. In our case we perform image segmentation to detect all pixels that correspond to our region of interest, the tumors.

We decided to build a UNet whose architecture (see Figure 1) builds upon fully convolutional networks. This model was originally invented for biomedical uses [6] and is a popular model when it comes to image segmentation. The main reason why we chose this network is for its capability of working with relatively small data-set while still being efficient.

The architecture of the UNet can seen as an encoder network followed by decoder network. "The encoder network halves the spatial dimensions and doubles the num-

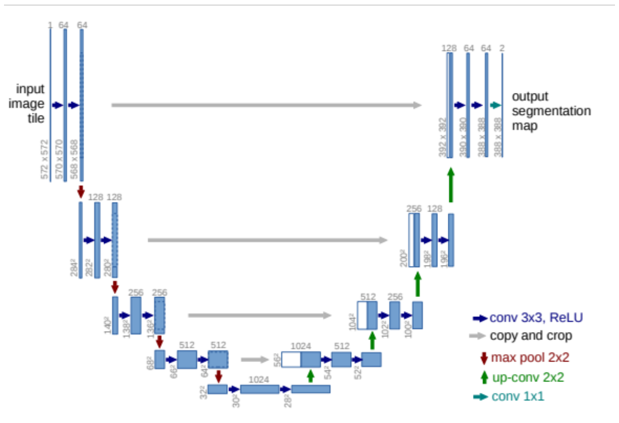


FIG. 1: U-Net architecture [11]

ber of filters. Likewise, the decoder doubles the spatial dimensions and halves the number of filters"[15].

"The encoder network acts as the feature extractor and learns an abstract representation of the input image through a sequence of the encoder blocks."[15] The encoder network is composed of smaller sub-units, called encoder blocks, linked together by max pooling layers. The encoder blocks contains two convolutional layers, each followed by a ReLU activation function, and finishes with a batch normalisation layer.

"The decoder network is used to take the abstract representation and generate a semantic segmentation mask."[15] The decoder network also works with blocks. First we use a deconvolutional layer then we concatenate the output with the output of the encoder block of the same depth. At the end of the block we use two convolutional layers.

Our model slightly differs from the normal UNet architecture as we decided to have the output size being the same as the input size. To do that we only had to use padding on the convolutional layers. A detailed view of our architecture can be found in the annex VI A.

C. Loss Function

The first lost function we decided to use is the Cross-Entropy loss as it is a recurrent choice in the literature. The purpose of the Cross Entropy is to take the output probabilities of the network and measure the distance from the truth values [7]. It is defined as

$$L_{CE} = -\sum_{i=1}^{n} t_i \; log(p_i)$$

where n is the number of classes, t_i is the truth label and p_i is the Softmax probability for the i^{th} class [7].

Once we were finished with using the Cross-Entropy loss we tried using the Dice loss. Compared to the Cross-Entropy which focuses on the similarity between probability distributions, the Dice loss focuses on the overlay between two segmentation masks. The Dice loss should improve the overall performance of the model but more particularly on image where the region of interest is small. The Dice loss is defined[2] as

$$Dice = \frac{2\sum_i^N p_i\ g_i}{\sum_i^N p_i^2 \sum_i^N g_i^2}$$

where N is the number of pixels in the image and p_i and g_i are the values from the prediction mask and the groundtruth mask. They are equal to 0 for the background and 1 for the region of interest.

This report will focus on the work built with the Cross-Entropy loss since we'll see later that we did not manage to get predictions with the Dice loss. The implementation that we used can be found here [16]. One-hot encoding of the groundtruth mask had to be computed in order to use the Dice loss.

D. Making a prediction

To generate the prediction mask we take the output from the network and pass it through a Softmax layer to generate the probabilities for all pixels. From there, all the pixels that are above a certain threshold value will be labelled as 1, meaning they belong to the region of interest. Since there are only two classes to predict in our problem, we choose a threshold of $0,5$.

E. Evaluation metrics

Once we have trained model we want to evaluate it, we want to know how well the model performs. To evaluate the model we use four metrics: the precision, the recall, the Dice score and the Intersection Over Union score.

Before defining the mentioned metrics we need to compute four values: the number of true positives, true negatives, false positives and false negatives. For that we use a code written by a discord user under the username *the-bass* [13]. The prediction matrix is divided by the groundtruth matrix. The true positives are obtained by summing all the elements labelled as 1, the false positives by summing up all the elements labelled as 'inf', the true negatives all the element labelled as 'nan' and the false negatives all the elements labelled as 0. We will use the abbreviations tp, fp, tn and fn to refer to the true positives, the false positives, the true negatives and the false negatives respectively.

The precision, also called positive predictive value, is defined as

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

The recall, also called sensitivity, is defined as

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

An ideal system would have an accuracy and recall of 1. However, in practice a compromise between these two values is necessary. If the recall value is high, the number of false positives is likely to be higher. Furthermore, if the precision is high, the system may not detect certain events. This is why, in general, the F1 score is used because it allows the two previous evaluation metrics to be combined. [10]In our case it is essential to use the binomial precision and recall to evaluate the performance because we have a dataset that is unbalanced. Indeed, there is a greater number of healthy pixels than pixels representing a tumour. []

The Dice score, which is also called the F1 score, is defined as

$$Dice = \frac{tp}{tp + fp + fn}$$

The Dice score is a score that evaluates the similarity between 2 sets. When it is 0 it indicates that there is no spatial overlap and conversely when it is 1 the spatial overlap is complete.[3] [14]

The Intersection Over Union is defined as

$$IoU = \frac{2tp}{2tp + fp + fn}$$

This measure corresponds to the superposition of the predicted region and the ground truth. When it is 0 it means that no superposition is present, while when it is 1 the superposition is total.[14]

F. Visualising the results

To visually compare two mask together we build a function that generate a new "mask" where each pixel is labelled as tp, tn, fp and fn. It will allow us to see what the network predicts correctly, what it predicted that it should not have, and what it should have predicted but it has not . Figure 2 represents an example of such a mask comparison. The pixels in blue represents the false negatives, those in orange the false positives, those in white and black represent the true positives and true negatives respectively.

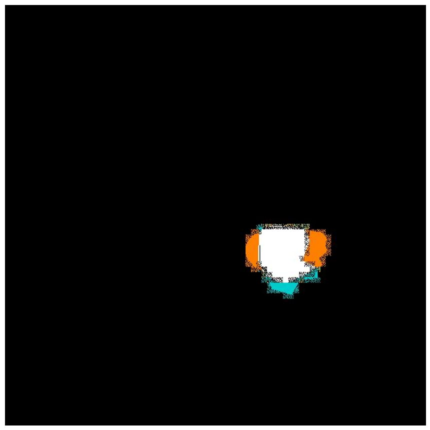


FIG. 2: Example of comparison between two masks

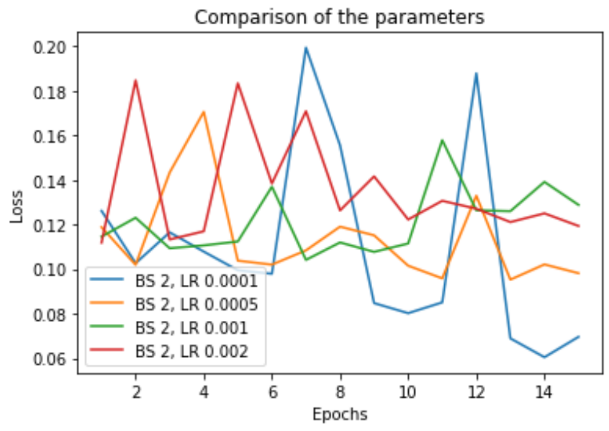


FIG. 3: Validating losses with a batch size of 2 for different learning rates

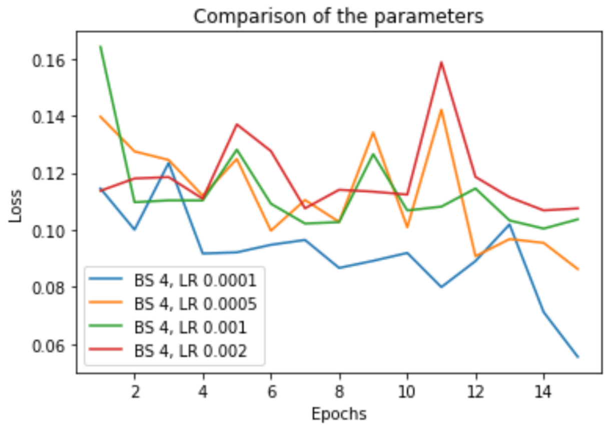


FIG. 4: Validating losses with a batch size of 4 for different learning rates

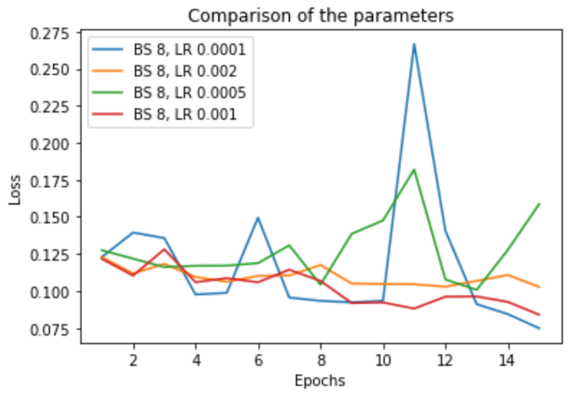


FIG. 5: Validating losses with a batch size of 8 for different learning rates

IV. RESULTS

A. Optimisation

To optimise the final performances we study the evolution of the network under different sets of parameters. In our model there are 3 parameters we can play with: the batch size, the learning rate, the number of epochs and the weight decay. Here we study the effects that the batch size and learning rate have.

Figure 3 represents the evolution of the validation loss for a batch size of 2 and four different learning rates (0.002, 0.001, 0.0005, 0.0001). Since the validation losses are oscillating over time it is harder to draw conclusions from those results. However, we observe that the lower the learning rate is and the lower the validation loss is overall. We also notice that with a learning rate of 0.0001 the model seems to improve over the epochs, leaving aside the oscillations.

Figure 4 represents the evolution of the validation loss for a batch size of 4 and the same learning rates as previously stated. It might be harder to see but once again the smaller the learning rate and the smaller the validation loss is overall. For the two learning rates 0.0001 and 0.0005 the model seems to still improve over the epochs, leaving aside the oscillations. Once again the learning rate that allows the model to have the smaller validation loss is the learning rate 0.0001.

The same manipulation was repeated with a batch size of 8. The results are shown in figure 5. In this instance it is harder to draw conclusion but once again the best validation loss is achieved with a learning rate of 0.0001.

We then compare the best runs for each value of the batch size in figure 6. We notice how the batch size does not seem to influence the validation loss. However, for a batch size of 4, the amplitude of the oscillations are smaller and the validation loss is slightly lower over time. Thus, the parameters we choose to train our model with are a batch size of 4 and a learning rate of 0.0001.

Now, one question still remains, after how many epochs should the model stop training? Usually we plot the validation loss curve and stop the training when the curve stops decreasing. We run the model for a consequent number of epochs to have a global view of the evolution

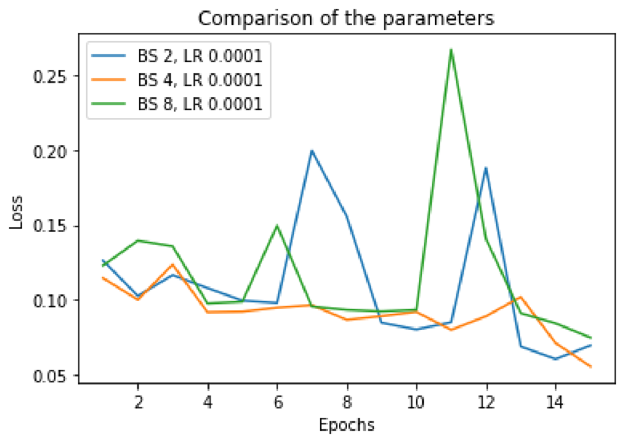


FIG. 6: Comparison of the best validation losses for a batch size of 2, 4, and 8

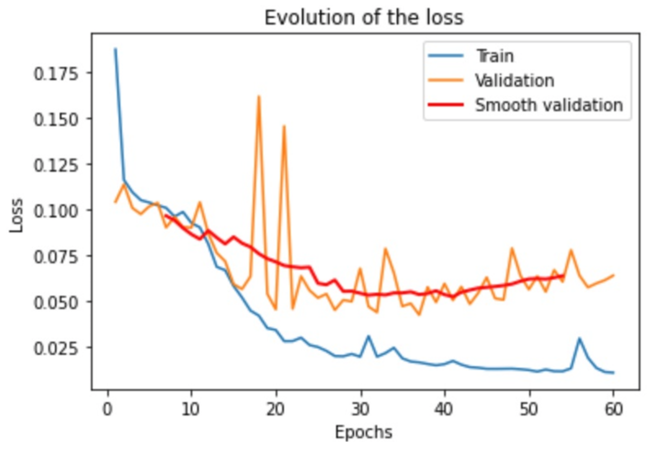


FIG. 7: Evolution of the training and validation losses through the epochs

of the loss, see figure 7. Since the validation loss oscillates we smooth the curve to have a better overview of what is happening. We observe that loss stops decreasing between 30 and 40 epochs. The smallest validation loss happens at epoch 37 with a validation loss of 0.0426. We know use the model as it was at the end of the epoch 37 to evaluate our model and make predictions.

We intended to repeat the same process using the Dice loss. However, most of the learning rates we used did not show any improvement of both the training and validation losses. The first curves that showed improvement of the losses can be seen in Figure 8. We were curious to see how our model was doing on the testing set but the results were disappointing. The model would not predict anything but background pixels. Since we did not manage to get any improvements we do not include further analysis on the Dice loss workflow.

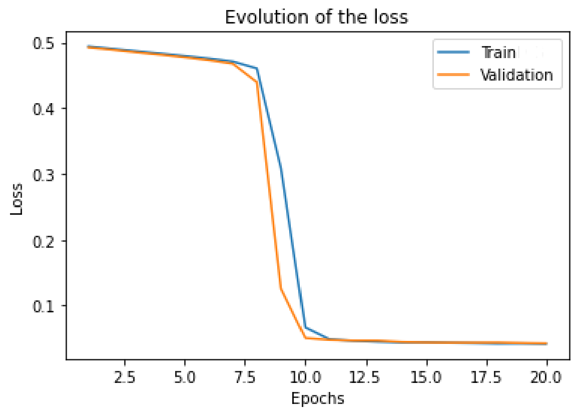


FIG. 8: Training and validation losses using Dice loss with a learning rate of 5 10^{-7}

	Mean	Min	Max	90% interval
Precision	0.913	0	1	[0.52, 1]
Recall	0.452	0	0.759	[0.128 0.711]
Dice	0.586	0	0.85	[0.226 0.829]
IoU	0.439	0	0.739	[0.128 0.708]

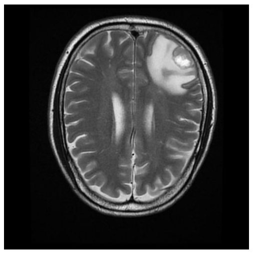
TABLE I: Performances of the model on the testing set

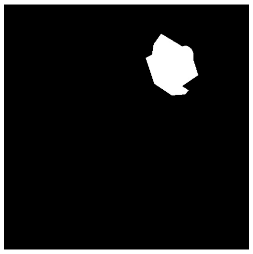
B. Quantitative analysis

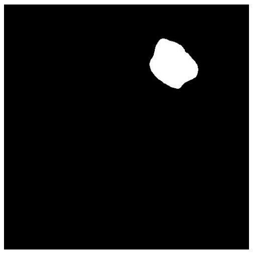
We use our model to generate all prediction masks and compute the precision, recall, dice score, and intersection over union score between the predicted mask and the groundtruth masks. We report the mean, minimum, maximum and 90% confidence interval of all those values. The computed values are reported in the table I

We observe that in average 91% of our predictions about the region of interest are correct. A precision of 1 means that there is no false positive in our prediction. A value of 1 either means that all the pixels predicted as the region of interest are actually background pixels either nothing is predicted as the region of interest. Our precision might be good but our recall is low. In average we recognize 45% of the original region of interest. Our Dice score and Intersection over Union score are quite low too. Furthermore, the fact that the accuracy is high and the recall is relatively low shows that the network detects few pixels with a tumour. However, the detected tumours are well placed.[9]

All the confidence intervals are quite large meaning that the mean values are not good representatives of out total performances. A large interval demonstrates a small degree of precision of our model.[4]


(a) Original image


(b) Groundtruth mask


(c) Prediction mask

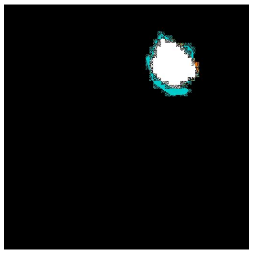
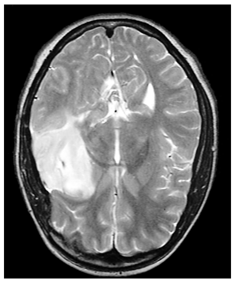
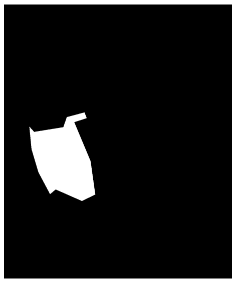
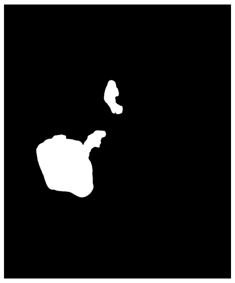
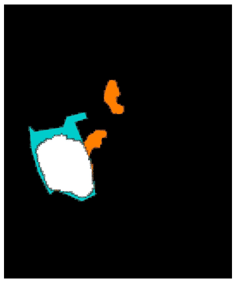

(d) Comparison of the two masks

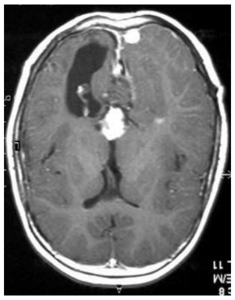
FIG. 9: Predicted mask with the highest Dice and IoU score.
a) The image to segment.
b) The groundtruth mask.
c) The prediction from the network.
d) Visual comparison of the two masks

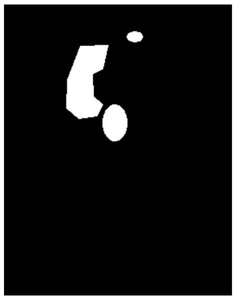

(e) Original image

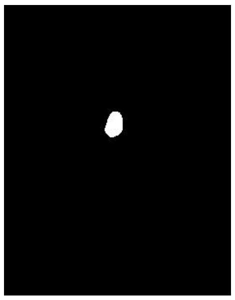

(f) Groundtruth mask


(g) Predicted mask


(h) Visual comparison


(i) Original image


(j) Groundtruth mask


(k) Predicted mask

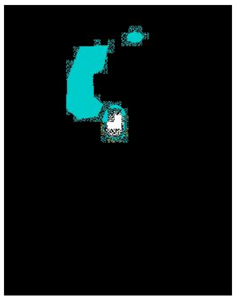

(l) Visual comparison

FIG. 10: Examples of predictions

C. Qualitative analysis

In this section we compare the predicted masks to the groundtruth masks.

Figure 9 represents the image from the testing set that has the highest Intersection over Union and Dice Score. Figures 10 and 11 represents four images taken at random in the testing set. Each figure is composed of the original brain image, the ground-truth mask, the predicted mask and the comparison of the two masks. Let's recall that the pixels in black are true negatives, those in white are true positives, in blue the false negatives and in orange the false positives.

The first observation that can be made from the following figures9d 9a 9c 9b is that the model predicts almost only diseased cells in the region where the cells are actually diseased. There are therefore very few false positives detected. This observation is in concordance with the definition of the Dice and Intersection over Union score since the smaller the number of false positives and

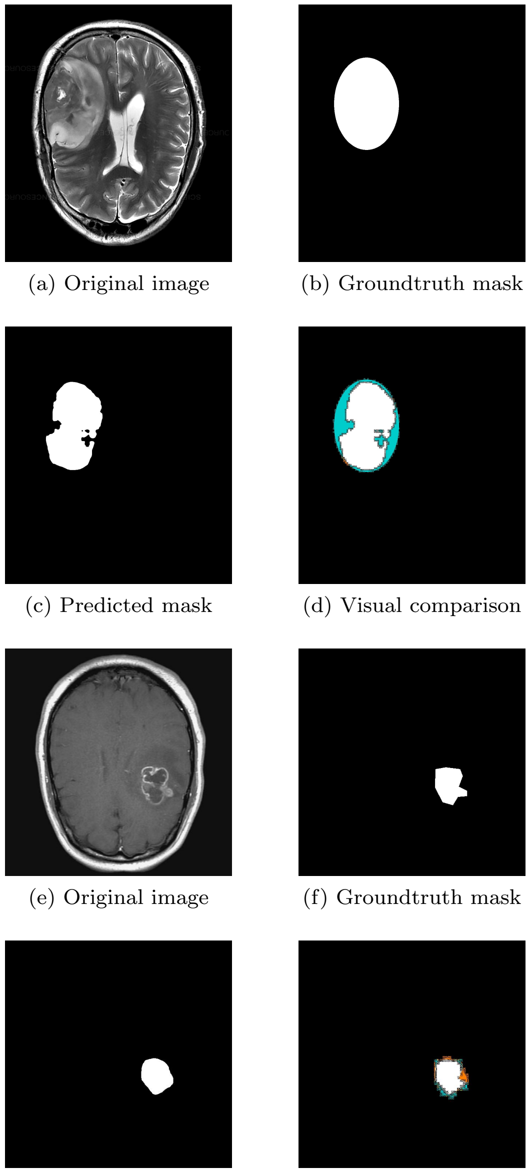

(a) Original image (b) Groundtruth mask (c) Predicted mask (d) Visual comparison (e) Original image (f) Groundtruth mask (g) Predicted mask (h) Visual comparison

FIG. 11: Other examples of predictions

false negative and the higher and the higher the scores. When looking at the overall predictions we observe that the model performs relatively well for tumours that are quite compact and have a simple contour. However, it can be noticed that the network is less efficient when the tumour edges are complex and the tumour is not condensed in one place. These observations are therefore consistent with the precision and recall results and their interpretation.

We are more satisfied at the model when analysing it in a qualitative way then quantitative. Indeed we observe that the model performs relatively well with relatively big and simple regions of interest. For the more complex regions of interest the model could do with some improvements. However, we are happy to see that the current predictions for those images are going in a right direction. Indeed, even if the edge of the tumour is not always predicted perfectly, the network detects the location of the tumour. For some applications this might be sufficient. Moreover, in the field of medicine it is important not to have too many false positive because it could be dangerous to start a treatment when the tissue is healthy. From this point of view our model is adequate because it predicts very few false positives.

V. DISCUSSION

The model trained using the cross entropy loss does not show satisfying results in terms of recall, Dice score and Intersection over Union score. However, when looking at the predicted masks the results are a bit more acceptable. The network might have difficulties with more complex regions of interest, it still shows potential. The network is also not perfect for simpler form which is not too concerning. Indeed, the groundtruth regions of interest are often represented with basic shapes and thus may not be precise. Thus we are not too concerned when false negatives appears at the edges. Overall, we conclude that the network is efficient in finding the area where the tumors are while having some trouble in detected all pixels belonging to them.

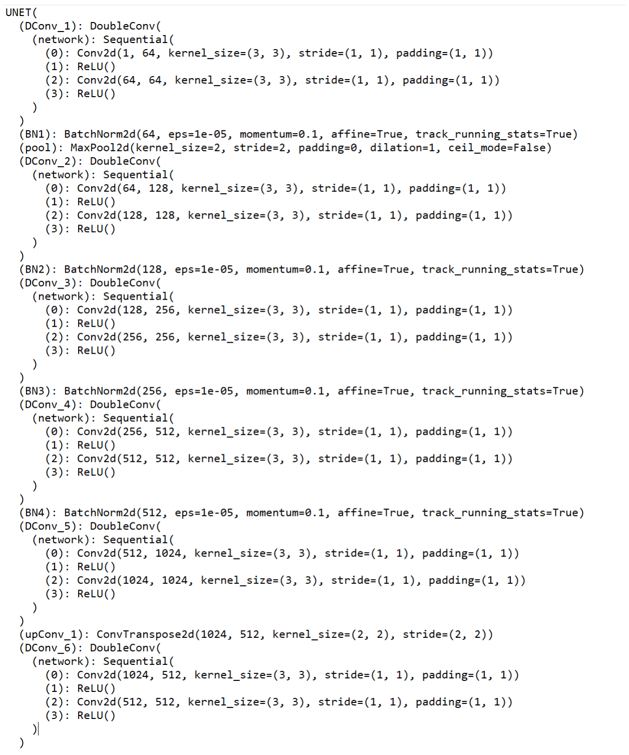
To improve the performances other loss functions could be studied as the cross entropy focuses on what the models predicts correctly and not what it misses. Among the other losses that exist we think of the Focal loss. It could potentially be beneficial in this case as it deals with the fact that the 2 classes present in the data set have very different proportions. We explained earlier that we tried using the Dice loss to train our model but did not yield any results. Therefore, before trying multiple loss function we should start by understanding why our run using the Dice loss did not perform well.

When observing the predicted mask we said that the network was less performing when the tumors shape where more complex. In particular we noticed that the predictions for images with multiple tumors were very poor. While it is not a problem to have only a few images of this type in the testing set (4%) it is more troublesome if they are rare in the training set. The training set has 500 images but only 19 have multiple tumors. If the class of image with multiple tumors was more represented in the training set, the predictions would be improved as well.

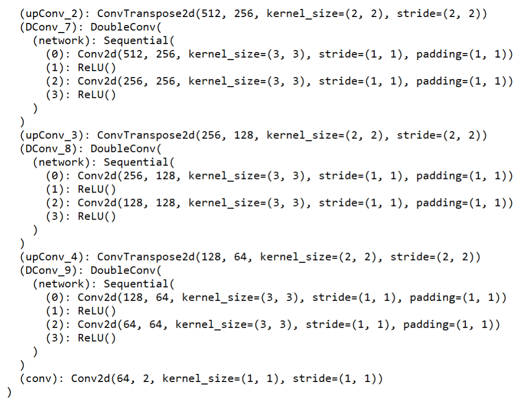
Another element that could be interesting to work on would be the oscillations in the validation loss. While completely eliminating the oscillations might be hardly feasible, it would be interesting to dampen them. Using smaller learning rates would indeed dampen to the detriment of fast learning. Smaller learning rates means that the networks takes more epochs to learn. Another possible path to follow could be to study the entire datasets and use cross-validation to determine the best split for our different sets. We took the actual splitting and assumed it was valid for our project while it is possible that it was not really split with care prior to this project.

VI. ANNEXES

A. Detailed model



```
UNET(
  (DConv_1): DoubleConv(
    (network): Sequential(
      (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (BN1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (DConv_2): DoubleConv(
    (network): Sequential(
      (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (BN2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (DConv_3): DoubleConv(
    (network): Sequential(
      (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (BN3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (DConv_4): DoubleConv(
    (network): Sequential(
      (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (BN4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (DConv_5): DoubleConv(
    (network): Sequential(
      (0): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (upConv_1): ConvTranspose2d(1024, 512, kernel_size=(2, 2), stride=(2, 2))
  (DConv_6): DoubleConv(
    (network): Sequential(
      (0): Conv2d(1024, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
```



```
  (upConv_2): ConvTranspose2d(512, 256, kernel_size=(2, 2), stride=(2, 2))
  (DConv_7): DoubleConv(
    (network): Sequential(
      (0): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (upConv_3): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2))
  (DConv_8): DoubleConv(
    (network): Sequential(
      (0): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (upConv_4): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2))
  (DConv_9): DoubleConv(
    (network): Sequential(
      (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU()
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU()
    )
  )
  (conv): Conv2d(64, 2, kernel_size=(1, 1), stride=(1, 1))
)
```

REFERENCES

[1] Kavita Bathe et al. *Brain Tumor Detection Using Deep Learning Techniques.* URL: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3867216`.

[2] Shuchen Du. *Understanding Dice Loss for Crisp Boundary Detection.* URL: `https://medium.com/ai-salon/understanding-dice-loss-for-crisp-boundary-detection-bb30c2e5f62b`.

[3] Esther Fleming. *What is Dice coefficient in image segmentation?* URL: https://www.sidmartinbio.org/what-is-dice-coefficient-in-image-segmentation/.

[4] Rangelands Gateway. *Confidence Interval.* URL: https://rangelandsgateway.org/inventorymonitoring/confidence.

[5] Ahmed Hamada. *Br35H :: Brain Tumor Detection 2020.* URL: https://www.kaggle.com/ahmedhamada0/brain-tumor-detection.

[6] *How U-net works?* URL: https://developers.arcgis.com/python/guide/how-unet-works/.

[7] Kiprono Elijah Koech. *Cross-Entropy Loss Function.* URL: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[8] Colin Elkin Nahian Siddique Paheding Sidike and Vijay Devabhaktuni. *U-Net and its variants for medical image segmentation: theory and applications.* URL: https://arxiv.org/ftp/arxiv/papers/2011/2011.01118.pdf.

[9] *Precision-Recall.* URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#:~:text=A%5C%20system%5C%20with%5C%20high%5C%20precision,with%5C%20all%5C%20results%5C%20labeled%5C%20correctly..

[10] *Precision-Recall-Classification Evaluation Framework: Application to Depth Estimation on Single Images.* URL: https://imatge.upc.edu/web/sites/default/files/pub/cPalou14.pdf.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* URL: https://arxiv.org/abs/1505.04597.

[12] Sidra Sajid, Saddam Hussain, and Amna Sarwar. *Brain Tumor Detection and Segmentation in MR Images Using Deep Learning.* URL: https://link.springer.com/article/10.1007/s13369-019-03967-8.

[13] the-bass. *Confusion matrix between two pytorch tensors.* URL: https://gist.github.com/the-bass/cae9f3976866776dea17a5049013258d.

[14] Ekin Tiu. *Metrics to Evaluate your Semantic Segmentation Model.* URL: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2.

[15] Nikhil Tomar. *What is UNET?* URL: https://medium.com/analytics-vidhya/what-is-unet-157314c87634.

[16] RNA Kaggle User. *Loss Function Library - Keras Py-Torch.* URL: https://www.kaggle.com/code/bigironsphere/loss-function-library-keras-pytorch/comments.

[17] Michael Yeung et al. *Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation.* URL: https://arxiv.org/abs/2102.04525.