Classification, Skull Stripping and Segmentation of Brain Tumor MRI

A Project Report

In view of LIFPROJET course

Bachelor of Computer Science in *Faculty of Science and Technology*

by

EPHREM Enzo, DE CLERCQ Allan

FACULTY OF SCIENCE AND TECHNOLOGY, DEPARTMENT OF COMPUTER SCIENCE

UNIVERISTY CLAUDE BERNARD LYON 1

LYON

December 2022

Abstract

Brain imaging refers to the use of various techniques to create images of the brain and its structures. These techniques can include CT scans, MRI, PET, and others. Deep learning is a subfield of machine learning that involves the use of artificial neural networks to learn and recognize patterns in data. In the context of brain imaging, deep learning can be used to analyze and interpret brain images, potentially aiding in the diagnosis of brain disorders and diseases.

Contents

1 Introduction

Classification and segmentation of brain tumors are important tasks in the field of medical image analysis. These techniques are used to accurately identify and locate tumors within brain images, which can help doctors diagnose and treat patients with brain cancer. The classification of brain tumors involves assigning a specific type or category to a tumor based on its characteristics, such as its size, shape, and appearance. Segmentation, on the other hand, involves identifying the exact location and boundaries of a tumor within an image. Both tasks are typically performed using specialized algorithms and software tools that are designed to analyze medical images and extract relevant information. By accurately classifying and segmenting brain tumors, doctors can make more informed decisions about a patient's treatment and care. But to improve this segmentation and the doctors' decisions, it is necessary to process as much data as possible and remove what is not useful, such as the skull. The skull stripping allows first of all a better visual view of the brain but also to reduce the errors of the segmentation. It is usually the radiologist's job to do this, but it is a fastidious and repetitive job that requires time and concentration. And like any repetitive work, it is possible to program it.

2 Methodology of Prediction

The proposed system uses noise correction because brain MRIs are very sensitive to noise, classification and skull removal to preprocess the images. The goal is to maximize segmentation by improving the quality of MRIs.

2.0.1 Methodology of Classifier Prediction

For reasons of access to medical data and computer performance. We have realized a 2D Conventional Neural Network for the classifying. The classifier is able to detect 3 types of tumor or a healthy brain (section 3.1.1). We chose the FLAIR sequence of the MRI because we think it's the one that makes the tumors stand out the most visually. As the CNN is in 2D, We chose to select 9 slices per view (section 3.1.2). For all the views We select the middle slice then from the middle, slices 5, 10, 15 and 20 positively and negatively. Once the MRI is cut in 27 images, We crop the slices, We apply a Gaussian filter if it has not already been applied, because some MRI automatically reduce the noise. After saving the new images, we send them to the CNN. After having made the sum of the 27 predictions. If the tumor is predicted to be 95% healthy, then the study stops there and the MRI is declared holy. Otherwise the MRI is declared infected by the tumor with the highest percentage.

2.0.2 Methodology of Skull Stripping Prediction

In order to join the classifier to the segmentation, we realized that the classifier was trained with MRI images with skull and the segmentation without skull. We first wanted to strip the skull from the images and re-train the classifier. We tried to preprocess the images using the methodology in Detection and Classification of Brain tumors Using Deep Convolutional Neural Networks [2] by using Otsu's method then recovering the largest component and applying a closing transformation. Unfortunately we did not have good results (figure 1). So we decided to make a new CNN. However, we did not find enough data sets for all sequences. We are currently able to strip the skull of T1 sequence. Due to lack of time, we reused the preprocessing of and its work [1]. We have however slightly modified its CNN which gave us better results 4.4.

2.0.3 Methodology of Segmentation Prediction

The segmentation methodology (Figure 4) comes after classifying and skull-stripping the MRIs, which is a way of filtering out unnecessary segmentation of non-tumorous tissues. By focusing only on the Flair, T1ce and T2 sequences, and after going through the pre-processing, we are ready to give the brain tumor to the U-Net. After splits of seconds we get our prediction that should undergo post-process to make our result readable. Using multiple display functions provided, we can easly look at the accurate 3D mask generated.

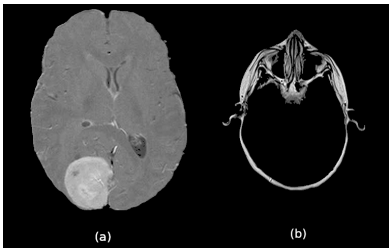


Figure 1: Illustration of skull stripping preprocessing : (a) good result, (b) bad result

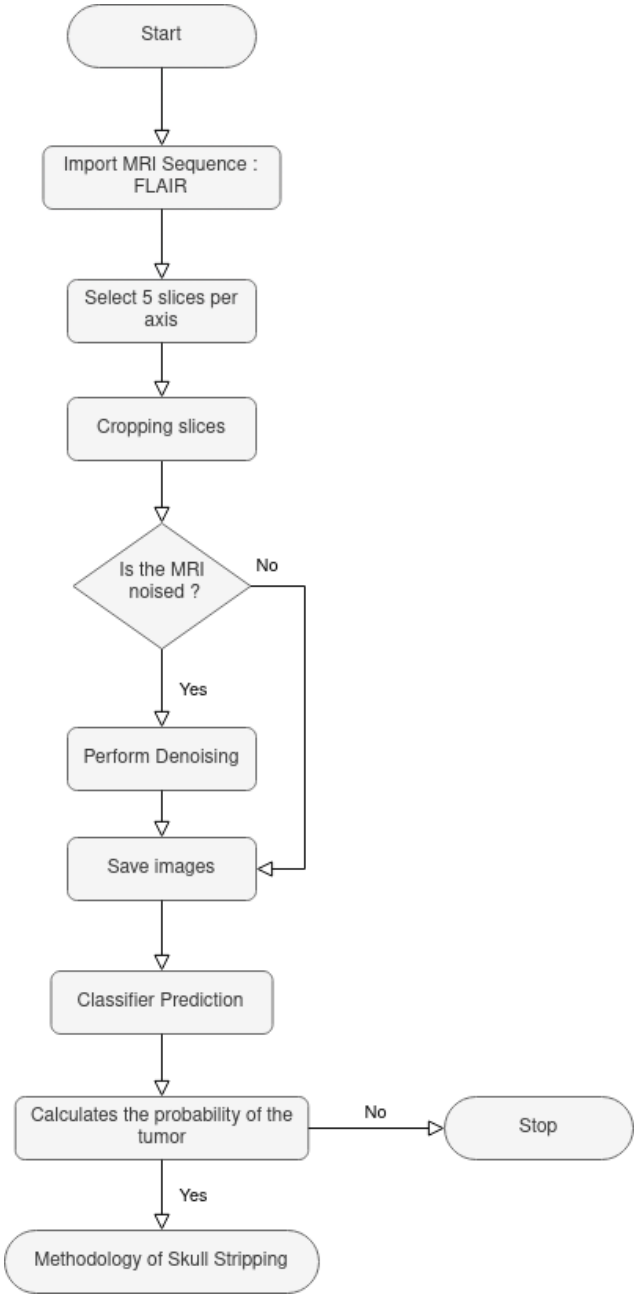


Figure 2: Methodology of Classifier Prediction Architecture

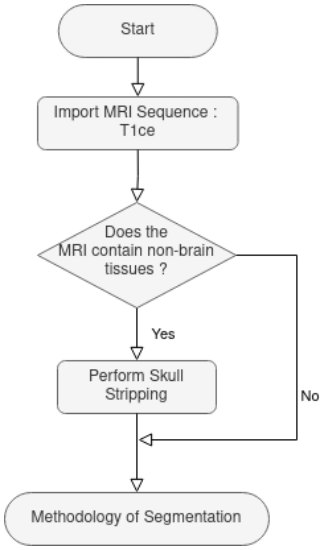


Figure 3: Methodology of Skull Stripping Prediction Architecture

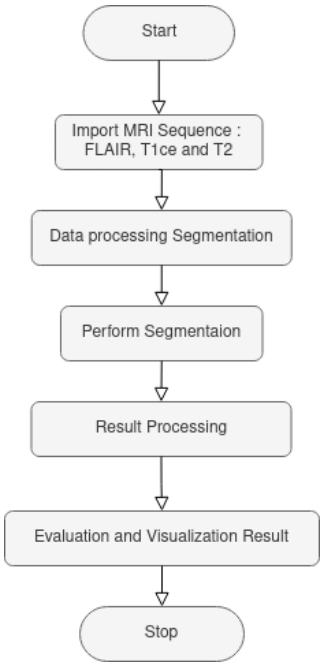


Figure 4: Methodology of Segmentation Prediction Architecture

3 Classification

3.1 Dataset Description

The dataset used is freely available on Kaggle [3]. The dataset consists of 3264 .jpeg MRI images of 3 brain tumors types : meningioma, glioma and pituitary tumors as well as images of healthy tumors. All images are resized to 150x150 dimensions. The MRI images in this dataset are from 3 different views: sagittal, coronal and axial. The number of MRI images without tumors being much lower were increased. The dataset dataset is composed of 500 MRIs of brain without tumor, 901 MRIs of 901 pituitary tumor MRI images, 937 meningioma tumor images and 926 glioma tumor MRI images.

3.1.1 Brain Tumors Types (Figure 5)

Meningioma tumor: Meningioma is a central nervous system tumor that starts from meninges that surround the brain and spinal cord. Normally, meningioma is the most common type of tumor that forms in the head.

Glioma tumor: Glioma is a central nervous system tumor that starts from glial cells in the brain or spinal cord. Glioma is a common type of brain and spinal cord tumor. It accounts for about 33 % of all brain tumors.

Pituitary tumor: Pituitary tumors are unusual growths that develop in the pituitary gland. This gland is an organ about the size of a pea. It's located behind the nose at the base of the brain. Some of these tumors cause the pituitary gland to make too much of certain hormones that control important body functions.

3.1.2 Different view of MRI (Figure 6)

Sagittal View: The sagittal view is an x-z plane that divides the body into left and right halves. It is called the sagittal plane because it passes through or is parallel to the sagittal suture,

Coronal View: The coronal view is an x-y plane that divides the body into front and back sections (also called dorsal and ventral).

Axial View: The axial (or horizontal) views, are x-z planes that are parallel to the ground and divide the body into upper and lower sections. The axial plane is also called the transverse plane.

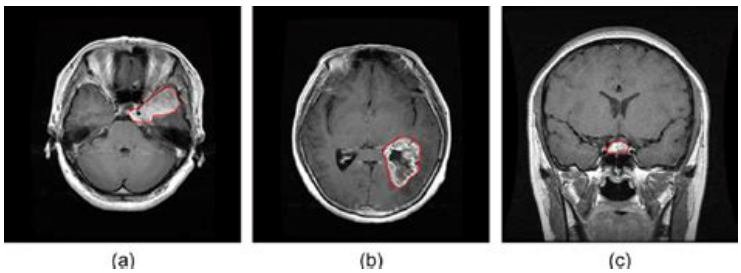


Figure 5: Illustrations of theses brain tumors types : (a) meningioma; (b) glioma; (c) pituitary

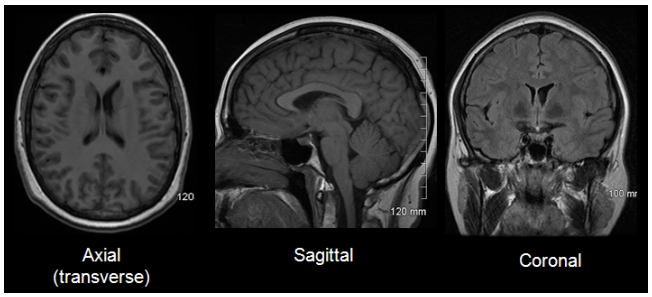


Figure 6: MRI images of different views

3.2 Environment

To classify it, we used Python and implemented it in jupyter Notebook. Python is an interpreted, multi-paradigm and multi-platform programming language. It supports structured, functional and object-oriented imperative programming. Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. We mainly used the following Python libraries: cv2, numpy, pathlib, matplotlib, tensorflow, keras, nibabel. The trainings were realized with Jupyter Notebooks on a device with an Intel i7-6700k 6th generation processor coupled with a AMD Radeon RX 5600 XT graphics card with 6 GB of RAM.

3.3 Model Settings Experiments Results

We chose to use a pre-trained model and then incorporate our CNN (section 3.5) to improve the results. Tumor detection is very important and fastidious, we can't afford to have a high rate of loss. We have chosen 4 different pre-trained models (ResNet50, VGG16, MobileNetV2, EfficientNetB0), 3 different loss function (categorical crossentropy, kl divergence, categorical hinge) and 3 different optimizer (Adam, Nadam, SGD). We we tested all the combinations to find the best one.

All the experiments were carried out with 15 epochs and a batch size of 2. The results are divided into 3 tables, one for each optimizer (Tables 1, 2, 3). Only the validation values will be represented because they are the ones we are most interested in. We are looking for the highest accuracy and the lowest loss.

Concerning its architecture it is very simple, with a simple initialization of the pre-model in 128x128x3 (image size), a GlobalAveragePooling2D layer and a Dense layer with 4 units (the number of classes: glioma, meningioma, notumor, pituitary) and the softmax activation function because we have several outputs and therefore a categorical model.

Table 1: Optimizer Adam Validation Results

Loss Function	Categorical Crossentropy		KL Divergence		Categorical Hinge	
Pre trained model	Val. loss	Val. accuracy	Val. loss	Val. accuracy	Val. loss	Val. accuracy
ResNet50	0.1718	0.9571	0.1673	0.9571	0.2717	0.8643
VGG16	0.1920	0.9746	0.2345	0.9343	1.0000	0.2703
MobileNetV2	0.2262	0.9448	0.1445	0.9562	0.2362	0.9265
EfficientNetB0	1.7897	0.3044	1.7647	0.2685	1.3397	0.3009

Table 2: Optimizer NAdam Validation Results

Loss Function	Categorical Crossentropy		KL Divergence		Categorical Hinge	
Pre trained model	Val. loss	Val. accuracy	Val. loss	Val. accuracy	Val. loss	Val. accuracy
ResNet50	0.1804	0.9623	0.1149	0.9711	0.1606	0.9203
VGG16	0.1217	0.9737	0.1507	0.9588	1.4593	0.2703
MobileNetV2	0.1264	0.9597	0.6589	0.8346	0.2829	0.8661
EfficientNetB0	1.4937	0.4033	7.3536	0.2265	1.4727	0.2633

Table 3: Optimizer SGD Validation Results

Loss Function	Categorical Crossentropy		KL Divergence		Categorical Hinge	
Pre trained model	Val. loss	Val. accuracy	Val. loss	Val. accuracy	Val. loss	Val. accuracy
ResNet50	0.2397	0.9256	0.2504	0.9265	0.2151	0.8967
VGG16	0.1353	0.9545	0.2300	0.9370	0.1298	0.9361
MobileNetV2	0.3668	0.8748	0.3231	0.8888	0.2965	0.8565
EfficientNetB0	1.3472	0.3508	1.2736	0.4286	1.0084	0.3998

By analyzing the results under 30 epochs, we can clearly determine in the figure 7 that the best pre-trained model to use is the VGG16. Analyzing the results of VGG16, we can easily see in the figure 8 that for the accurency, the maximum of the kl divergence function is the highest. with a very high median. and a very low loss per epochs too. So we will choose to use it

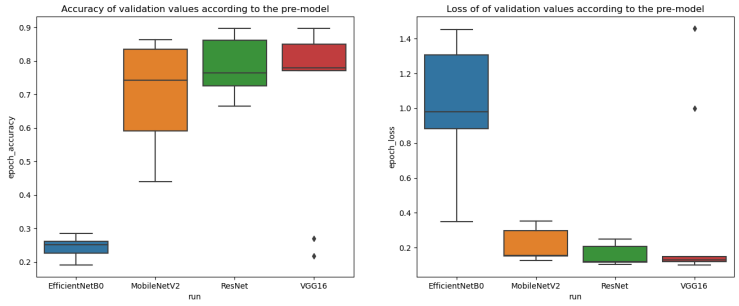


Figure 7: Box plot of pre trained models

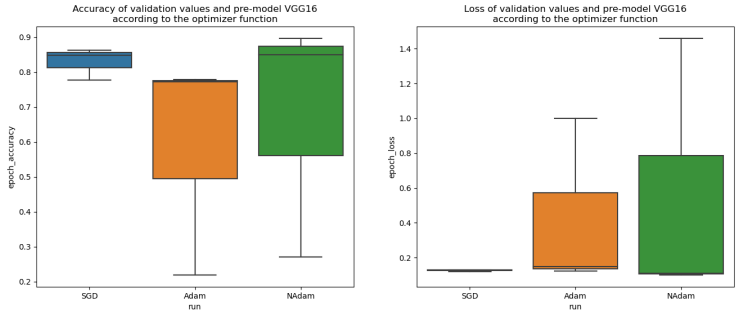


Figure 8: Box plot of optimizers

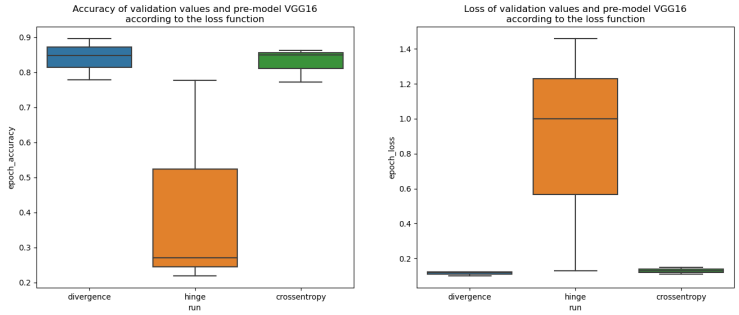


Figure 9: Box plot of losses functions

3.4 Model Settings

3.4.1 Pre-trained : VGG16

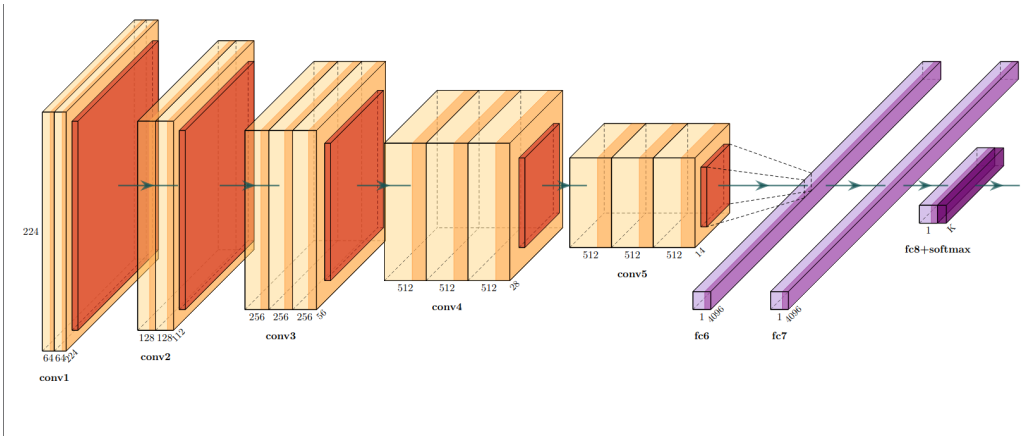


Figure 10: VGG16 Architecture

3.4.2 Loss Function : Kullback–Leibler divergence

$$D_{KL}(P\|Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad (1)$$

D_{KL} = Kullback–Leibler divergence loss function

$Q(x)$ = Prediction distribution, tensor inuput

$P(x)$ = True distribution, tensor target

$P(x)$ and $Q(x)$ are tensors of the same shape.

3.4.3 Optimizer : Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function with appropriate smoothing properties. SGD performs a parameter update for *each* training example x and each label y.

$$w_{t+1} \leftarrow w_t - \eta \nabla L(w_t) \quad (2)$$

3.5 Classifier Model Architecture

To improve the classifier, we decided to add to the output of the pre trained model: a layer of Dense then a layer of Dropout then 3 layers of Dense, see figure 11. It uses the section 3.4 parameters and has been trained under 50 epochs

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 128, 128, 3)]	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
global_average_pooling2d_8 (GlobalAveragePooling2D)	(None, 512)	0
dense_33 (Dense)	(None, 64)	32832
dropout_17 (Dropout)	(None, 64)	0
dense_34 (Dense)	(None, 32)	2080
dense_35 (Dense)	(None, 16)	528
dense_36 (Dense)	(None, 4)	68

Total params: 14,750,196
Trainable params: 14,750,196
Non-trainable params: 0

Figure 11: Classifier architecture

3.6 Advanced Classifier Results

3.6.1 Metrics

Model Name	Optional Skip	MSE Loss	Dice Score	IoU Score
MobileNetV2_SGD_kl_divergence_50_2.h5	True	0.019821	98.0179 %	96.1247 %

3.6.2 Loss and Accuracy

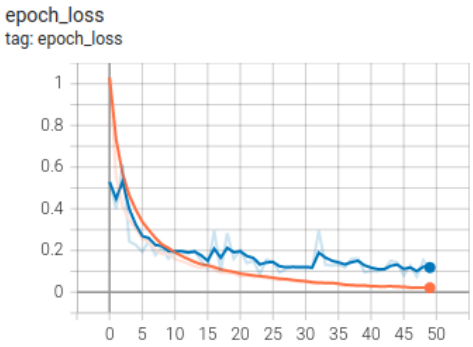


Figure 12: Advanced classifier : Loss per epochs

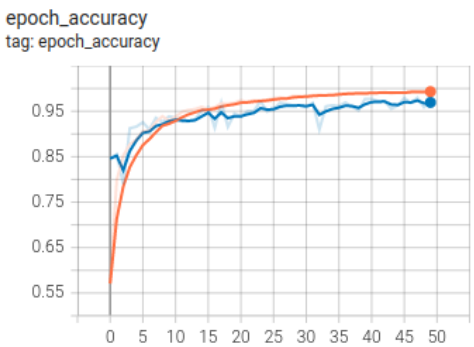


Figure 13: Advanced classifier : Accurency per epochs

3.6.3 Convolution Matrix

We can be observed on the convolution matrix figure **??** that the classifier has no problem to recognize MRI images without tumors with 100% of accuracy. This does not mean that the model will be true for every MRI image, there is always a possibility of improvement.

It is also very good at recognizing pituitary and meningioma tumors. We believe that this is because these tumors have precise location, shape and size, see figure 5. While glioma tumors do not have a specific location they can be in different places of different size and shape so more complicated to spot.

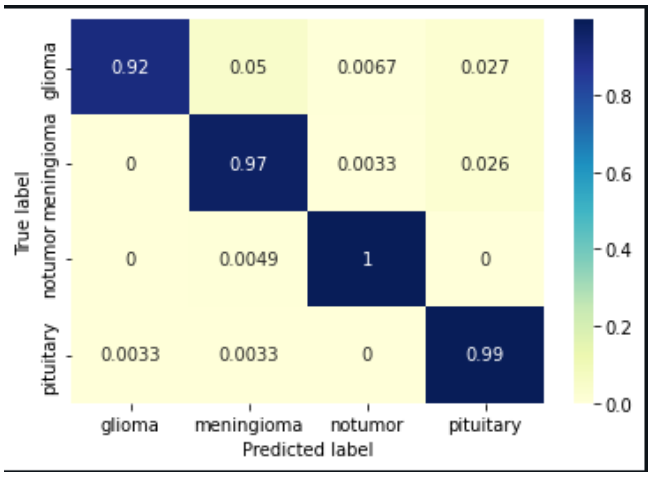


Figure 14: Convolution matrix of the advanced classifier

4 Skull Stripping

4.1 Environment

To strip it, we used Python and implemented it in Jupyter Notebook . See 3.2 for description of Python and Jupyter Notebook. This time we use Pytorch instead of Tensorflow. The trainings were realized with Jupyter Notebooks on a Google Cloab PRO+ server with an NVIDIA gpu coupled with 89.6 GB of RAM.

4.2 Dataset Description

For the segmentation of t1w MRI scans, we used the Neurofeedback Skull-stripped repository (NFBS) [5], which is a database of 125 anatomical T1-weighted MRI scans that have been manually stripped. We split the data set in 2 with 80% (100 mri) for the training phase and 20% (25 mri) for the test phase. It contains data from participants, aged 21 to 45 years, with a variety of clinical and subclinical psychiatric symptoms. For each participant, the repository contains:

- Anonymized T1-weighted image
- Skull image
- Brain mask

The resolution of the images is 1 mm3 and each file is in NiFTI (.nii.gz) format.

4.3 Primary model of skull stripping

4.3.1 Description and architecture

The model we used is Muraligm Akshay's Residual UNET 3D model [1] which is a custom image segmentation architecture inspired by Residual Networks and UNET for 3D images. Below figure 15 the architecture diagram of the Residual Block, figure 16 the architecture diagram of the Upscale Block which reuses Residual Block . As well as figure 17 the architecture diagram of the Residual UNET 3D which uses the 2 blocks. The model has been trained with 6 epochs and a batch size of 6 using the Standard

Segmentation Loss function and the Adam optimizer

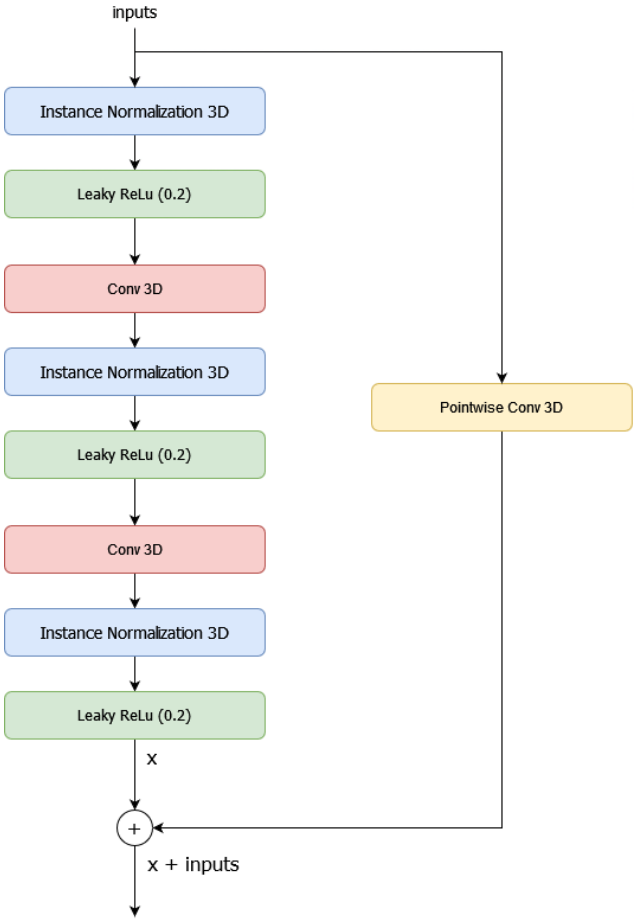


Figure 15: Architecture of the Residual Block

List of used layers :

- **Conv 3D** : See section 5.3
- **Instance Normalization 3D** : Also known as contrast normalization, this is a normalization layer that prevents instance-specific mean and covariance shifting, which simplifies the learning process.
- **Leaky ReLu** : Leaky ReLU, is a type of activation function based on the ReLU function, but it has a small slope for negative values instead of the flat slope of the ReLU, look at figure 18.
- **Pointwise Conv 3D** : It's a Conv 3D for shortcut tensor.
- **Sigmoid** : It's an application of the sigmoid function, look at figure 19.

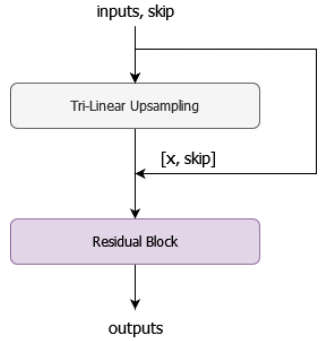


Figure 16: Architecture of the Upscale Block

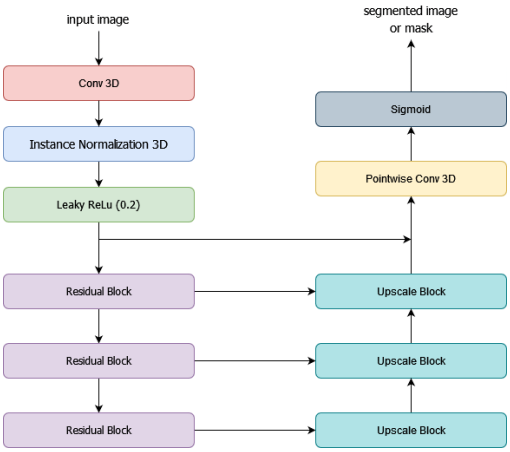


Figure 17: Architecture of the Residual UNET 3D

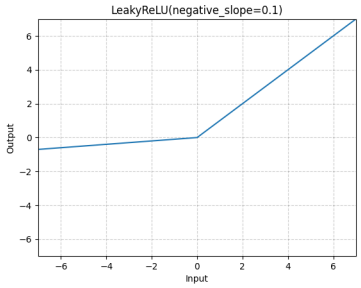


Figure 18: Leaky ReLu function

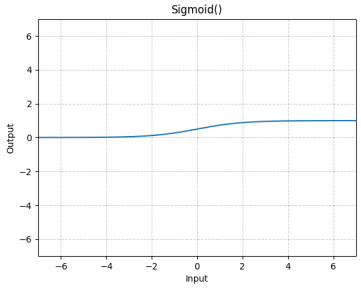


Figure 19: Sigmoid function

4.3.2 Results

Table 4: Muraligm Akshay's Results of Skull Stripping T1w

Model Name	Optional Skip	MSE Loss	Dice Score	IoU Score
residual_unet3d_MSE_2.pth	False	0.008068	91.76 %	87.53 %
residual_unet3d_MSE_3.pth	True	0.010058	91.13 %	86.11 %

4.4 Advanced model of skull stripping

4.4.1 Description and architecture

We didn't want to simply copy the model but to improve it. First we decided to modify the model parameters by increasing the number of epochs from 6 to 12. We also increased the batch size to 8.

In the article "SynthStrip: Skull-Stripping for Any Brain Image" [6] the scientists used the loss function: DiceLoss and obtained good results. So we have also decided to use this function. However, they talk about another even better loss function: SDT (Signed Distance Transform) but we did not manage to apply it.

We also kept the Adam optimizer which is very efficient for segmentation.

We noticed that there was no Dropout layer in the model. A DropOut layer randomly sets some of the elements of the input tensor to zero with probability p using samples from a Bernoulli distribution. Each is set to zero independently of each dropout call.

As described in the article "Improving neural networks by preventing co-adaptation of feature detectors" [7], it is used to prevent co-adaptation of neurons and thus avoid overtraining. As we only have 125 different MRIs, we thought it was useful to add drop out layers to improve the model.

The modifications have been made in the Residual Block. At the entry in the block, we apply a normalization layer and then the LeakyRelu function. Then a 3d convolution layer and again the normalization and LeakyRelu layer exactly like the first model. It is at this moment that we apply a DropOut of 0,2. We continue by reapplying the sub-block (Conv3D, Normalization, LeakyRelu), we apply a 2nd DropOut of 0.1 this time and we add a 3rd sub-block not to end with a DropOut. See figure 20

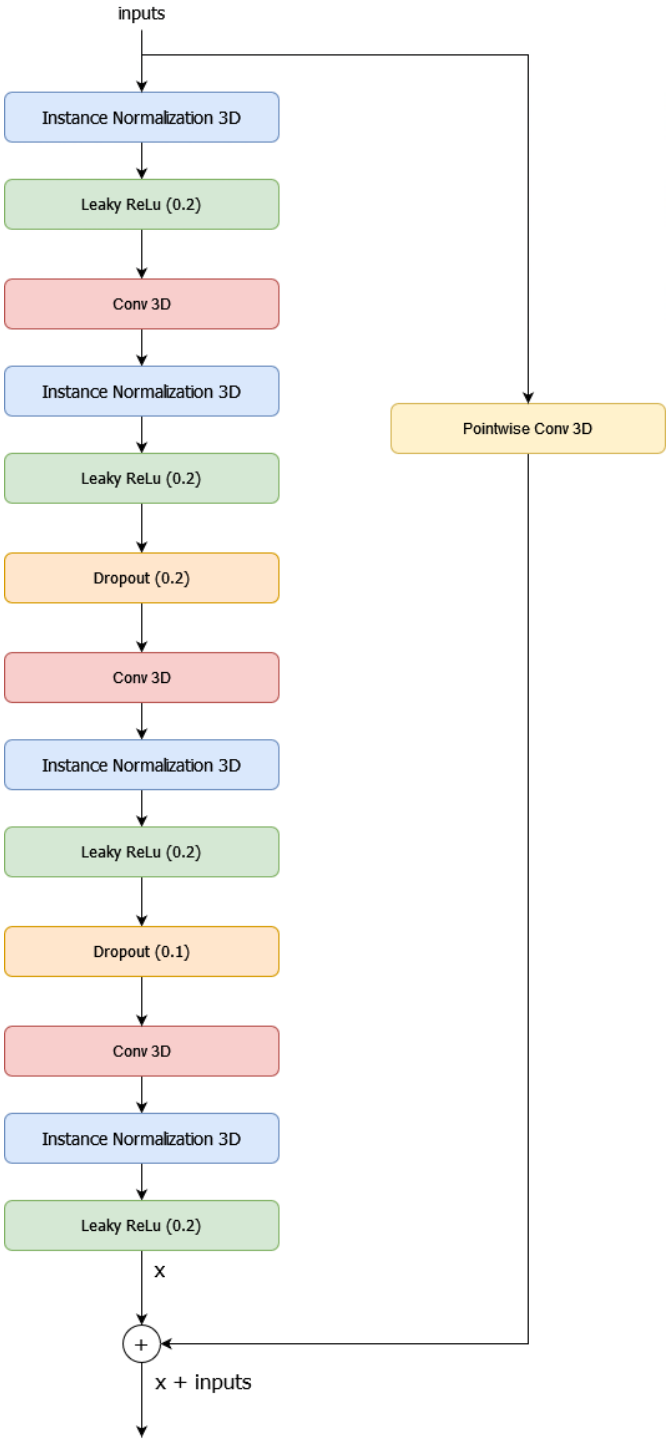


Figure 20: New Architecture of the Residual Block

4.4.2 Results

Concerning the T1w skull stripping model, our new model has a smaller MSE Loss, 0.019821 against 0.010058 for the first model. But we get a better Dice Score with 98.0179% against 91.13%. And to finish a much better IoU Score with 96.1247% against 86.11%.

Table 5: Results of the Advanced model of skull stripping T1w

Model Name	Optional Skip	MSE Loss	Dice Score	IoU Score
ResidualUNET3D_Adam_10_3.pth	True	0.019821	98.0179 %	96.1247 %

5 Segmentation

Brain tumor segmentation is the process of identifying and locating tumor tissue in medical imaging of the brain. This is an important task in medical diagnosis and treatment planning for patients with brain tumors. In the context of brain tumor segmentation, deep learning algorithms can be trained to recognize and segment tumor tissue from medical images of the brain, such as in particular magnetic resonance imaging (MRI) scans. (limits bounding box region)

5.1 Data

To train a CNN for brain tumor segmentation, a large dataset of medical images with corresponding ground truth segmentation masks is needed. These masks define the boundaries of the tumor tissue in the images and are used to train the CNN to recognize the appearance of tumor tissue. Provided in the kaggle BraTS2020 Dataset (Training + Validation) [4] in the validation folder, for each of the 369 patients, 4 sequences of MRI (Flair, T1, T1ce and T2) and one corresponding mask.

5.1.1 Schema

```
dataset
├── BraTS20_Training_001/
│   ├── BraTS20_Training_001_flair.nii
│   ├── BraTS20_Training_001_seg.nii
│   ├── BraTS20_Training_001_t1.nii
│   ├── BraTS20_Training_001_t1ce.nii
│   └── BraTS20_Training_001_t2.nii
├── BraTS20_Training_{ID}/
│   ├── BraTS20_Training_{ID}_flair.nii
│   ├── BraTS20_Training_{ID}_seg.nii
│   ├── BraTS20_Training_{ID}_t1.nii
│   ├── BraTS20_Training_{ID}_t1ce.nii
│   └── BraTS20_Training_{ID}_t2.nii
└── ....
```

5.1.2 Sequences

There are various types of brain MRI sequences (different MRI parameters) that are used to obtain different types of information about the brain.

T1-weighted

This sequence produces images that highlight the different types of tissue in the brain, such as gray matter, white matter, and cerebrospinal fluid. It is commonly used to diagnose brain tumors and other abnormalities.

T1ce (T1-weighted contrast-enhanced)

A recovery of the T1-weighted sequence but with a contrast agent to make certain structures in the brain more visible on the resulting images.

T2-weighted

This sequence produces images that highlight areas of the brain where there is fluid accumulation, such as in the case of brain swelling or inflammation. It is commonly used to diagnose conditions such as hydrocephalus and multiple sclerosis.

FLAIR (fluid-attenuated inversion recovery)

This sequence is similar to T2-weighted imaging, but it produces more detailed images of the brain's fluid-filled spaces. It is commonly used to diagnose conditions such as brain tumors and cerebral edema.

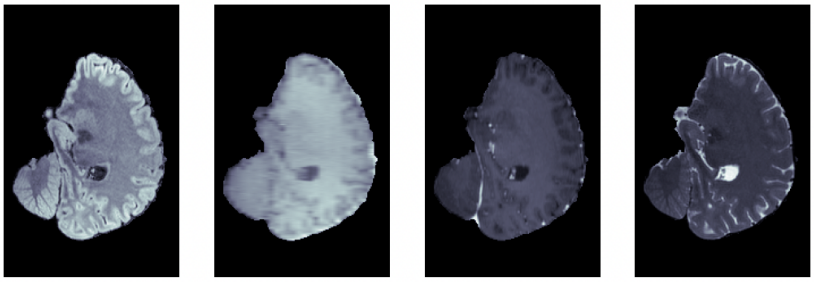


Figure 21: Different types of sequences in brain MRI : (a) Flair; (b) T1; (c) T1ce; (d) T2

5.1.3 Masks

A mask is an image that is used to identify the location and boundaries of a tumor in a medical image of the brain. By creating a mask of the tumor region, doctors and other medical professionals can more easily identify the tumor, its location, size and shape within the brain. This can be useful in planning treatment and monitoring the progress of the tumor over time. In our dataset the mask images are labeled to identify different tissue types within the brain tumor. Each voxel (the contraction of volume and pixel), has a value of 0, 1, 2 or 4. Label 0 represents a healthy tissue

Label 1: Necrosis

When a brain tumor is present, the abnormal growth of cells can lead to necrosis in the surrounding tissue. This can happen if the tumor grows too large and begins to press on nearby brain tissue, restricting the blood flow and oxygen supply to the affected area. The lack of oxygen and nutrients can cause the cells in the surrounding tissue to die, leading to necrosis. In some cases, the tumor itself can undergo necrosis, which can cause the tumor to shrink in size. However, necrosis can also cause the release of harmful substances into the surrounding tissue, leading to further damage and potentially causing complications.

Label 2: Enhancing tumor

An enhancing tumor in a brain tumor is a type of tumor that appears to be growing or becoming more active on imaging tests, such as MRI scans. When a brain tumor is present, the abnormal growth of cells can cause changes in the surrounding tissue, including the formation of new blood vessels. These new blood vessels can appear as a ”halo” or ”ring” of bright spots on an MRI scan, indicating that the tumor is growing or becoming more active. This is known as enhancement, and it can be a sign that the tumor is growing or that treatment is not working effectively. It is important to monitor for signs of enhancement in a brain tumor, as it can help doctors determine the best course of treatment.

Label 4: Edema

Edema is the medical term for swelling caused by an accumulation of fluid in the body's tissues. In the case of a brain tumor, edema can occur if the tumor grows and begins to press on nearby brain tissue. This can restrict the flow of cerebrospinal fluid, the clear liquid that surrounds and protects the brain and spinal cord. The buildup of fluid in the brain can cause the tissue to swell, leading to edema. Edema in the brain can be a serious condition and can cause a range of symptoms, including headache, nausea, and difficulty with balance and coordination. It is important to monitor for signs of edema in a brain tumor and to seek medical treatment if it occurs.

Figure 22: Mask Labels : (a) Whole Tumor; (b) Necrosis; (c) Enhancing tumor; (d) Edema

5.2 Data processing

Pre-processing and preparing brain MRI images and masks before training the model will benefit in numerous ways. Scans in the dataset (brain and mask) have a shape of 240x240x155. and have a .nii extension (.nii for a file type NIfTI-1).

NiBabel

NiBabel is a library that support common medical and neuroimaging file formats including .nii files. It gives full or selective access to header (meta) information and provides a numpy interface NumPy of the scans.

5.2.1 Brain MRI pre-processing

First of all it is important to try and reduce the neural network input data size given that it is considerably high (240x240x155 = 8 928 00 floats, amounting to approximately 34 MB per channel if using float32) especially with low computational resources at hand. The T1 channel can be discarded as the T1ce channel contains the same information in an enhanced way. Furthermore the borders are mostly 0-valued and can be cropped reducing the shape to 128x128x128 per channel. The second step is scaling each MRI image using the min-max scaling. Min-max scaling Eq. 3 is a method used to normalize data. It works by transforming the data so that all the values lie between 0 and 1. This is often useful when working with neural networks, as it can help to improve the convergence of the algorithm and can also make it easier to compare different features of the data. It also reduces bias triggered by relatively large range of values, and can help with computing time.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

5.2.2 Mask MRI pre-processing

It is important to remember that there are few values that masks can take for given coordinates Section 5.1.3. Since the label 3 is empty, replacing the label 4 by the label 3 before categorizing the masks would help eliminating an unnecessary class. Post-processing (processing the prediction) will handle switching all label 3 to 4 afterwards. Before handing out our labels to our model, to match the shape of the brain images, we also crop it from 240x240x155 to 128x128x128.

5.3 Model architecture

In deep learning, a model architecture refers to the specific design of a neural network, including the number of layers, the number of neurons in each layer, and the connections between them. The architecture of a model determines its ability to learn from data and make predictions, so choosing an appropriate architecture is an important part of building a successful deep learning model. A U-Net [12] is a type of convolutional neural network (CNN) that is often used for image segmentation tasks. It is called a U-Net because it has a ”U” shaped architecture that is made up of two parts: an encoder that processes the input image and extracts features from it, and a decoder that expands the feature map and produces the segmented output. The U-Net is typically used for tasks such as medical image segmentation, where it can

be trained to identify and segment specific structures in medical images. This is why it will be our chosen architecture for the segmentation of the tumor. The following diagram was made with PlotNeuralNet

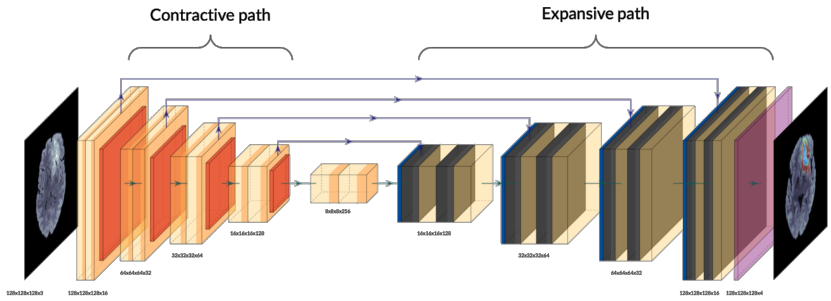


Figure 23: U-Net model architecture for segmentation

The encoding part, is made of 4 convolutional blocks. Knowing that a convolutional block is made of 2 Conv3D with a ReLu activation and a MaxPooling3D. And the decoding part 4 blocks of, 1 Conv3DTranspose and a combination of 2 Conv3D with a ReLu activation as well. Connecting the encoding and decoding parts, are 2 Conv3D called the bottleneck. Note that dropout layers are present throughout the model, and that the model ends with a Conv3D layers with a soft max activation. Finally arrows in Fig 23 represents concatenations.

Conv3D

As the name suggests, a Conv3D layer performs 3D convolutions over the input data. The purpose of a Conv3D layer is to learn spatial relationships in the input data by applying filters (also referred as kernels). The filters are typically small 3D volumes, and they slide across the input data, performing dot multiplications and summing the results to produce a new 3D volume. By applying multiple filters to the input data, a Conv3D layer can learn to detect a variety of spatial patterns in the data. In the suggested neural network, all Conv3D have kernels equal to (3, 3, 3) except the last one (1, 1, 1). The number of filters applied depends on if we are in the encoding or decoding part and how deep.

MaxPooling3D

MaxPooling3D is a technique used to reduce the size of the 3D input tensor. The input tensor is divided into a set of non-overlapping 3D patches, and for each patch, the maximum value is taken and used as the output of the maxpooling layer for that patch. The goal is to down-sample the feature maps, which can help to extract more robust features from the input data and improve the performance of the model. The output tensor has a smaller size than the input tensor, as it has fewer patches. In our neural network, having a pool_size of (2, 2, 2) the input tensor will be halved.

Conv3DTranspose

Conv3DTranspose is used to upsample the input tensor, which means it increases the spatial resolution of the feature maps produced by the layer. It is typically used to generate high-resolution output from low-resolution input. Like a Conv3D, the Conv3DTranpose takes a number of filters and a 3D kernel. The chosen kernel is (2, 2, 2) and the number of filter matches the number of filters in the Conv3D of the same block.

5.4 Segmentation loss function and metrics

A loss function is a function that is used to measure the performance of a model on a given dataset. The goal of training a deep learning model is to find the set of model parameters that minimizes the loss function, so that the model can make accurate predictions on new data. Metrics, on the other hand, are used to evaluate the performance of a model on a given dataset. Unlike the loss function, which is used during training to update the model's parameters, metrics are used to

evaluate the model's performance after training is complete. There are many different types of metrics that can be used in deep learning. In general, the loss function and metrics are used together to evaluate the performance of a deep learning model and to guide the training process.

5.4.1 Loss function

The loss function used for the model is a combination of two well known loss functions. The Dice Loss Eq. 5, and the Categorical Focal Loss Eq. 6. Both provided by the segmentation model 3D library [13].

$$LossFunc = DiceLoss + (1 * FocalLoss) \tag{4}$$

Dice Loss

The dice loss is commonly used while training any model for segmentation. It follows a simple equation:

$$\begin{gathered} DiceLoss(gt, pr) = 1 - DiceCoef(gt, pr) \\ \text{where } gt = \text{ground truth}, \\ pr = \text{prediction} \end{gathered} \tag{5}$$

The next section will get deeper into what the Dice coefficient is, Section 5.4.2.

Categorical Focal Loss

Categorical focal loss is a loss function that can be used in the training of deep learning models for classification tasks. It is designed to address the issue of class imbalance, which occurs when the number of examples for some classes is much larger than the number of examples for other classes. This can lead to a model that performs poorly on the underrepresented classes, as it is more likely to predict the more frequently occurring classes. Having a multi-class segmentation and an imbalance in the types of tissue in the tumor (Necrosis, Enhancing tumor, Edema), justifies the contribution of the Categorical Focal Loss in our model's loss function.

$$\begin{gathered} FocalLoss(gt, pr) = -gt \cdot \alpha \cdot (1 - pr)^{\gamma} \cdot \log(pr) \\ \text{where } gt = \text{ground truth}, \\ pr = \text{prediction}, \\ \alpha = 0.25, \gamma = 2.0 \end{gathered} \tag{6}$$

5.4.2 Dice coefficient

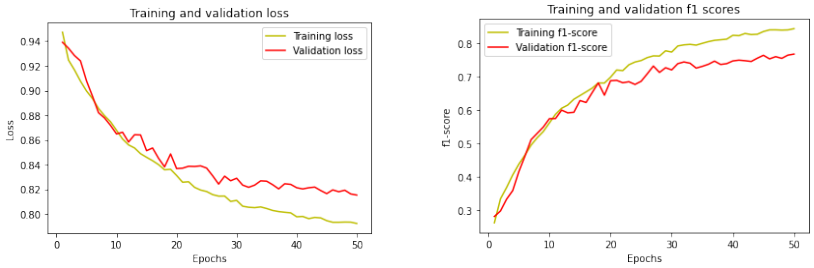
Used as the main metric, the Dice Coefficient has proven to be a very good indicator of segmentation correctness. The Dice coefficient is a measure of the similarity between two sets. It is defined as the ratio of the size of the intersection of the sets to the size of the union of the sets. The Dice coefficient is often used to evaluate the performance of image segmentation, where the sets being compared are the sets of pixels that are predicted to belong to a particular class and the set of pixels that are actually in that class. The Dice coefficient can range from 0, indicating no overlap between the sets, to 1, indicating complete overlap. It is calculated as follows:

$$\begin{gathered} DiceCoeff(gt, pr) = 2 * \frac{|gt \cap pr|}{|gt + pr|} \\ \text{where } gt = \text{ground truth}, \\ pr = \text{prediction}, \\ |gt \cap pr| = \text{set of elements to both gt and pr} \end{gathered} \tag{7}$$

5.5 Results and Analysis

5.5.1 Training

The model was trained on 295 of the 369 patients, in batches of 2. The other 74 were used as validation. Figure 24 (a) and Figure 24 (b) respectively shows the variation of the loss function and the FScore (Dice Coefficient) during the 50 epochs.



(a) Loss Function (b) FScore

Figure 24: Evaluation of the model

5.5.2 BraTS Challenge validation results

The Brain Tumor Segmentation (BraTS) challenge is an annual competition organized by the Multimodal Brain Tumor Segmentation (BRATS) consortium. It is designed to evaluate and compare the performance of different algorithms for segmenting brain tumors in magnetic resonance imaging (MRI) scans. In the BraTS challenge, participants are given a dataset of MRI scans of brain tumors and are asked to develop algorithms to automatically segment the tumors in the scans. The algorithms are then evaluated on their ability to accurately identify and outline the tumor regions in the MRI scans. In the scope of this report, a validation zip file of the predicted masks to the validation dataset of 2021 has been submitted as a validation evaluation to the model. The table that follows shows results of the submission (submission was made with a model trained for 30 epochs).

	Dice_ET	Dice_TC	Dice_WT
mean	0.578	0.387	0.745
median	0.696	0.387	0.776
25quantile	0.459	0.245	0.682
75quantile	0.801	0.515	0.853

Table 6: Dice scores of ET-enhancing tumor, TC-tumor core (necrosis), WT-Whole tumor

Based on Table 6 the model is capable of accurately predicting the shape of the whole tumor, but fails to accurately distinguish between the core tumor and the enhancing tumor.

6 Conclusion and Future Scope

In conclusion of this project, we can conclude that we have exceeded our expectations. We did not expect to obtain such high results. The skull stripping model has exceeded the results of Muraligm Akshay. The segmentation results are in the upper range of the BRATS competition.

The only thing we regret is that we did not succeed in fully linking our project. We missed a few days to realize several skull stripping models according to T2 and FLAIR sequences. This would have simply required a dataset search, which can sometimes be very long, the transfer of the T1 sequence to T1ce by applying filters and several tens of hours of training.

If we could review the methodology of our project. We would have placed the skull stripping first, then a new 3D classifier that would take as input a MRI sequence and finally the segmentation.

Unfortunately we could not do the 3D classifier because of lack of data. It is difficult to get MRI of the brain and even more difficult to get healthy MRI.

References

[1] Skull Stripping and ICA, Muraligm Akshay, Machine Learning Engineer https://github.com/aksh-ai/skull-stripping-and-ica

[2] G. Balaji, R. Sen, H. Kirty, "Detection and Classification of Brain tumors Using Deep Convolutional Neural Networks", 4.1.4 Skull Stripping, arXiv preprint https://arxiv.org/pdf/2208.13264.pdf

[3] Kaggle, "Brain Tumor MRI Dataset", https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

[4] Kaggle, "BraTS2020 Dataset (Training + Validation)", https://www.kaggle.com/datasets/awsaf49/brats20-dataset-training-validation

[5] Neurofeedback Skull-stripped (NFBS) repository , http://preprocessed-connectomes-project.org/NFB_skullstripped/

[6] A. Hoopes, J. Mora, A. Dalca, B. Fischl and M. Hoffmann, "SynthStrip: Skull-Stripping for Any Brain Image", arXiv preprint https://arxiv.org/abs/2203.09974

[7] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv preprint https://arxiv.org/abs/1207.0580

[8] 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: https://ieeexplore.ieee.org/document/6975210

[9] 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: https://www.nature.com/articles/sdata2017117

[10] 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 https://arxiv.org/abs/1811.02629

[11] U.Baid, et al., The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification, arXiv:2107.02314, 2021. arXiv preprint, https://arxiv.org/abs/2107.02314

[12] Ronneberger, Olaf and Fischer, Philipp and Brox, Thomas, "U-Net: Convolutional Networks for Biomedical Image Segmentation" (2015), arXiv preprint https://arxiv.org/abs/1505.04597

[13] Solovyev, Roman and Kalinin, Alexandr A and Gabruseva, Tatiana, "3D convolutional neural networks for stalled brain capillary detection", Computers in Biology and Medicine 2022, doi: 10.1016/j.compbiomed.2021.105089 https://github.com/ZFTurbo/segmentation_models_3D

List of Figures

List of Tables