HAL
open science

# Fully Automatic Brain Tumor Segmentation using End-to-End Incremental Deep Neural Networks in MRI images

Mostefa Ben Naceur, Rachida Saouli, Mohamed Akil, Rostom Kachouri

## To cite this version:

Mostefa Ben Naceur, Rachida Saouli, Mohamed Akil, Rostom Kachouri. Fully Automatic Brain Tumor Segmentation using End-to-End Incremental Deep Neural Networks in MRI images. Computer Methods and Programs in Biomedicine, 2018, 166, pp.39 - 49. 10.1016/j.cmpb.2018.09.007 . hal-01893017

**HAL Id: hal-01893017**

**https://hal.science/hal-01893017**

Submitted on 11 Oct 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Fully Automatic Brain Tumor Segmentation using End-to-End Incremental Deep Neural Networks in MRI images

Mostefa Ben naceur[a,b,*], Rachida Saouli[a,*], Mohamed Akil[b], Rostom Kachouri[b]

[a] *Smart Computer Sciences Laboratory, Department of Computer Sciences, University of Biskra, Biskra, Algeria*
[b] *Gaspard Monge Computer Science Laboratory, ESIEE-Paris, University Paris-Est Marne-la-Vallée, France*

**Abstract**

Background and Objective: Nowadays, getting an efficient Brain Tumor Segmentation in Multi-Sequence MR images as soon as possible, gives an early clinical diagnosis, treatment and follow-up. The aim of this study is to develop a new deep learning model for the segmentation of brain tumors. The proposed models are used to segment the brain tumors of Glioblastomas (with both high and low grade). Glioblastomas have four properties: different sizes, shapes, contrasts, in addition, Glioblastomas appear anywhere in the brain.

Methods: In this paper, we propose three end-to-end Incremental Deep Convolutional Neural Networks models for fully automatic Brain Tumor Segmentation. Our proposed models are different from the other CNNs-based models that follow the technique of trial and error process which does not use any guided approach to get the suitable hyper-parameters. Moreover, we adopt the technique of Ensemble Learning to design a more efficient model. For solving the problem of training CNNs model, we propose a new training strategy which takes into account the most influencing hyper-parameters by bounding and setting a roof to these hyper-parameters to accelerate the training.

Results: Our experiment results reported on BRATS-2017 dataset. The proposed deep learning models achieve the state-of-the-art performance without any post-processing operations. Indeed, our models achieve in average 0.88 Dice score over the complete region. Moreover, the efficient design with the advantage of GPU implementation, allows our three deep learning models to achieve brain segmentation results in average 20.87 seconds.

Conclusions: The proposed deep learning models are effective for the segmentation of brain tumors and allow to obtain high accurate results. Moreover, the proposed models could help the physician experts to reduce the time of diagnostic.

*Keywords:* Deep learning, Fully automatic, Brain tumor segmentation, Convolutional neural networks, Training, Hyper-parameters.

*Corresponding author.

*Email addresses:* `mostefa.bennaceur@univ-biskra.dz` (Mostefa Ben naceur ), `rachida.saouli@esiee.fr` (Rachida Saouli ), `mohamed.akil@esiee.fr` (Mohamed Akil ), `rostom.kachouri@esiee.fr` (Rostom Kachouri)

## 1. Introduction

Brain tumor is a growing abnormal cell in the brain or central spin canal. In USA ([1]), National Brain Tumor Society estimates that every year 13.000 patients die, and 29.000 patients suffer from primary brain tumors (i.e. primary tumor is a tumor starts in the brain). In 2007, they expected in UK ([2]), more than 4.200 patients have a brain tumor each year. Thus, the number of experts is insufficient compared to the number of patients in the world. In addition, if we count also the potentiality of medical mistakes (i.e. false negatives: diagnostic result indicates that a person does not have a disease, but in fact he has) besides low survival rates of patients with life-threatening diseases such as Glioblastomas. Glioblastomas affect children between 5 and 10 years old, and adults between 30 and 40 years old. Moreover, Glioblastomas are the most frequent primary brain tumors in adults ([3]).

MRI is the investigation tool of choice for the segmentation of brain tumors such as Glioblastomas and for the evaluation of treatment response ([4]). Usually, an expert radiologist uses MRI technique to generat a sequence of images (Flair, T1, T1 contrast, T2, ...etc) to identify different regions of tumor. This variety of images helps radiologists and experts to extract different types of information about the tumor (shape, volume ...etc). In general, the manual segmentation in MRI is a time-consuming procedure and depends on skills of each expert ([5];[6];[7]) .

During the last three decades, the problem of brain tumor segmentation has attracted many researchers, in which the number of huge published papers as noted by ([8]) involving brain tumor segmentation increased exponentially. In this case, the creating of a smart brain tumor segmentation system has always been a highly needed option. In addition, instead of wasting time with manual diagnosis by the experts, this smart segmentation system decreases the needed time for the diagnosis process. This way gives to doctors more time with their patients in the process of treatment and follow-up. There are many researches that have been carried out for automating this task of radiologist (i.e. localization and segmentation of the whole tumor and its sub-regions), these algorithms are classified into two categories ([8]): *Generative models* ([9]), these models need a *prior knowledge* about tumor and its tissue and its appearance, but these models require recording several parameters about tumor's features. However, *Discriminative models* ([10]) directly learn the characteristics of tumor and how to segment it from manually annotated data (created by radiologists). Some algorithms need a huge training data such as Deep Learning algorithms. Convolutional Neural Networks during the ILSVRC [1] have proved that they are a powerful tool for feature detection and extraction. In 2012, this tool led to a big breakthrough in all computer vision tasks. A model based on Convolutional Neural Networks called AlexNet was developed by ([11]) achieved the best results in image classification. AlexNet is a bigger model of LeNet5 ([12]) (i.e. AlexNet added more layers than LeNet5) and now the most

[1] ILSVRC: ImageNet Large-Scale Visual Recognition Challenge

successful models on computer vision field are an extended version of the AlexNet model such as ZFNet ([13]), VGGNet ([14]), GoogleLetNet ([15]), RestNet ([16]). CNNs have outperformed several traditional machine learning algorithms in wide types of computer vision tasks, such as object detection ( [17]; [18]; [19]; [20]).

The aim of this paper is to develop a fully automatic (i.e. without any user-intervention) MRI Brain tumor segmentation using end-to-end incremental CNNs model. We propose three deep learning models (2CNet, 3CNet, EnsembleNet), 2CNet and 3CNet are End-to-End models, each of which has only one Network from the input image to the output instead of multiple boxes of treatment (or networks), EnsembleNet is the aggregation of these two End-to-End deep learning models. In general, the conventional machine learning approaches require many processing steps, but end-to-end deep learning usually replaces these steps by a single neural network. We build this end-to-end model incrementally to get the maximum advantage of each layer before moving on to the next one. These end-to-end deep learning models are applied to Glioblastomas tumors. With such Glioblastomas tumors that have four unpredictable properties: different sizes, shapes, contrasts, moreover, these tumors appear anywhere in the brain. Thus, the most suitable machine learning solution is Convolutional Neural Networks. The biggest challenge that we face is that: Glioblastomas have four sub-regions ([8]), there are respectively 4 types of intra-tumoral structures: edema, non-enhancing core and necrotic core, enhancing core, and healthy tissue.

In this paper, our main goal is to propose and develop a new fully automatic method for the segmentation of Brain tumor. For achieving this goal, our contributions are divided into four fold:

1. We propose an "Incremental XCNet" algorithm which generates CNNs models, i.e. from a base model (a limited number of layers) we obtain a deeper and scalable model through our technique of "Incremental XCNet".
2. We propose a new strategy called "ELOBA_$\lambda$" to train the CNNs models, in which it takes into account the most influencing hyper-parameters (Epochs, Learning rate, Optimizer, Batch size). Our training strategy "ELOBA_$\lambda$" bounds and sets a roof to these hyper-parameters to fast the training of the generated CNNs models.
3. We design End-to-End Incremental deep neural networks called "EnsembleNet" which combines on parallel the instances of "Incremental XCNet" to get the benefits of each model based on a new non-parametric fusion technique.
4. Our models segments the whole brain in average 20.87 seconds by exploiting the advantage of GPU implementation. Thus, our models are one order of magnitude faster than the other deep learning models.

After describing the CNNs pipeline, and the convolution operation, pooling layer, and the classifier, (in Section 3). We review the state-of-the-art in the field of brain tumor segmentation, machine learning based

methods, and the current models based on CNNs, (in Section 2). Then, we explain our approach for solving the problem of brain tumor segmentation and the problem of training CNNs-based models (in Section 4) and (Section 5) respectively. Finally, the results of our approach are reported and discussed (in Section 6). Conclusion and perspective described at the end, (in Section 8).

## 2. Related Work

There are several methods that have been developed for Brain Tumor Detection and Segmentation: threshold-based methods ([21]; [22]), Region-based methods ([23]; [24]; [25]), Edge-based methods ([26]; [27]; [28]), Atlas-based methods ([29]; [30]; [31]), Classification and Clustering methods ([32]; [33]; [34]; [35]). These methods are not computationally expensive but they generally produce poor results and need a user interaction (e.g. region growing) most of the time. Moreover, they need a *prior knowledge* from experts and feature engineering. Therefore, these techniques cannot be fully automatic algorithms. As the goal is to develop a fully automatic method for the segmentation of brain tumors, obviously a new method is needed.

After the breakthrough in 2012 of the AlexNet ([11]) model which is based on the Convolutional Neural Networks, many CNNs-based approaches from 2014 to the present have proposed for the segmentation of brain magnetic resonance (MR) images. Furthermore, we have investigated several CNNs architectures which are implemented for BraTS[2] challenge.

Zikic et al. [36] proposed a brain tumor segmentation method based on CNNs. The input of their algorithm is a (4*19*19) 2D-patch matrix (i.e. four channels: T1, T2, T1c, and Flair). The CNNs network is used to segment the MRI images to five classes: non-tumor, necrosis, edema, non-enhancing tumor and enhancing tumor. Their CNNs model is a sequential network contains 5 layers.

Another work used CNNs introduced by Urban et al. [37] for brain tumor segmentation, their approach is different. They used a sequential 3D-CNNs model, in which the input data are 3D patches of size (9*9*9) voxels with four channels (i.e. T1, T1c, T2, FLAIR). Moreover, as known in several CNNs models, they did not use the pooling layers. Also, they used a post-processing step where they removed all regions of less than 3000 voxels. In addition, this method takes one minute to segment the whole brain using GPU implementation.

Axel et al. [38] used CNNs for the segmentation of brain images. The input data of the network are 2D patches of size (32*32) pixels. The network is divided into two parallel pathways; the first pathway is a CNNs architecture (i.e. two Maxout convolutional layers -Maxout is the result of merging two or more feature maps-), and the second pathway is a fully connected Maxout layer, then these two pathways are concatenated at the end into a softmax layer. Moreover, this model takes 20 minutes to segment the whole

[2]BraTS 2017 challenge braintumorsegmentation.org/

brain using GPU implementation.

Pereira et al. [39] developed a method based on CNNs for brain tumor segmentation. They used two CNNs architectures for each type of Glioblastomas (i.e., High Grade HGG and Low Grade LGG). Their algorithms take as inputs 2D patches of size (33*33) pixels with four channels (i.e. four MRI sequences). The last step implemented in their algorithms is post-processing, they applied a morphological filter to delete isolated regions. In addition, this model takes 10 minutes to segment the complete brain using GPU implementation.

Havaei et al. [40] extended the previous work of Axel et al. [38], where their CNNs architecture has two pathways, each of which takes different 2D patches size with four MR sequences as channels. Moreover, the output of the first pathway is used as an additional input to the second network. Also, they applied a post-processing step, which they removed flat blobs near to the skull based on connected components. In addition, this method takes three minutes to segment the whole brain using GPU implementation .

In this study, we have been motivated by the recent deep learning models especially when the performance of brain tumor segmentation methods becomes more accurate and the inference time is decreased. The performance of a model on all tumor regions is measured by various metric evaluations (see section 6.3). Inference time represents the needed time to make prediction on a new data set, which means the quantity of operations that needed to execute the model on the platform (device). Thus, for achieving these two goals (i.e. performance, inference time), we propose different models. First, to reduce inference time, the sequential CNNs architecture is the best choice. Because the sequential CNNs architecture takes less time than the parallel CNNs architecture due to losing time during the concatenation period. Based on this rule, we develop our 2CNet and 3CNet models (see section 4). Second, we have noticed during the ILSVRC challenge, year after year, that each model outperformed the previews state-of-the-art model, in which each model used a deeper and bigger model: ALexNet [11] used 8 layers, VGGNet [14] used 19 layers, GoogleLetNet [15] used 22 layers, RestNet [16] used 152 layers. Moreover, one of the successful ways to create a deep learning model and improve its performance ([15]) is by increasing the depth (i.e., the number of layers) and the width (i.e., the number of units per layer). Thus, our algorithm "Incremental XCNet" (see section 4.1 ) is created based on these rules, in which it attempts to improve the performance by exploring the depth response of CNNs model to the test pattern. Third, our models (2CNet, 3CNet, EnsembleNet) do not use any post-processing step and even the pre-processing we use the minimum as used in many image processing applications. This way allows the models to be independent of data's nature and to reduce the needed time to segment the images. In addition, the most used technique to develop a deep learning model is: trial and error. This technique depends on creating a deep learning model then train this model, if the model failed then we develop another model and so on until one of models succeed. However, the algorithm of "Incremental XCNet" applies a new technique differs from the conventional CNNs-based models that use

the technique of trial and error. "Incremental XCNet" creates and trains the CNNs model at the same time using a new training strategy called ELOBA_$\lambda$ (see section 5.1). Thus, with this method of "Incremental XCNet" we can reduce the development and training time of CNNs models.

## 3. Convolutional Neural Networks

Convolutional Neural Networks have been proven to be a powerful tool for feature detection and extraction in many computer vision tasks. In general, the CNN-based pipeline begins with a fixed-size matrix for example an input image, in our case (32 * 32) matrix. Additionally, the images have a spatially correlated nature, in which, the image's features are distributed across the whole image. So, CNNs have exploited these properties and added a new layer called convolution. This convolution layer is applied to the input images (raw pixels). The idea of adding convolution layer before classifier helped to extract useful features at multiple locations. By stacking many convolution layers (pooling layer can be added alternatively with convolution), kernels in CNNs play a role of a feature detector. So, by choosing a specific kernel we can detect a specific feature that we want it. The kernel parameters represent the weights that need to update to get the most suitable configuration, in another word, the kernel tuned during the training process for pushing it to respond strongly to a specific feature. This updating operation of weights is often done by the Back-propagation ([41]) algorithm. Where lower kernels parameters are trained to detect features such as edges, corners, and higher kernels parameters are trained to detect more complex features or high-level features such as shapes and object parts. The output of convolution operation called feature map which is calculated as follows:

$$Y_i = b_i + \sum_j W_{ij} * X_j \tag{1}$$

Where $Y_i$ is the $i^{th}$ feature map, $b_i$ is a trainable parameter bias, $W_{ij}$ is a trainable parameter weights. In CNNs models $W_{ij}$ are the kernels parameters. $X_j$ is the input, * is the convolution operator.

Convolution layer reduces the number of dot-product operations by using the property of sparse interactions which is in contrast with the traditional feed forward neural network.

In general, the most activation function used in deep learning models is Rectified Linear Unit (ReLU) ([42]), it solved the *vanishing gradient* problem. In addition, ReLU is used for introducing the non-linearity because convolution is a linear operation:

$$F_k = Max(0, Z) \tag{2}$$

Where Z is the input to the neuron k.

Pooling layer lets the images lose some information, and this leads to reduce the size of feature map, but this operation makes the images less sensitive and invariant to small translations (resisting to local translation). MaxPooling operation takes the maximum value of each non-overlapping rectangle (sub-region) of feature map. MaxPooling ($G_{ij}$) is computed as follows:

$$G_{ij} = max_s Y_{i+s,j+s} \tag{3}$$

Where S is the size of sub-region.

The last module in CNNs pipeline is a classifier, usually Multi-layers Perceptron is used.

With this kind of deep learning model appeared several issues like the problem of over-fitting which is fixed by using regularization techniques such as Dropout ([43]); Dropout ignores some neurons with a probability during the training phase, this technique allows each neuron to learn several independent representations of the same data.

## 4. The proposed brain tumor segmentation method

In this section, we explain our three CNNs models (2CNet, 3CNet, EnsembleNet) for solving the problem of brain tumor segmentation. The main idea of our CNNs architecture is based on the development of a generic algorithm, which for each different parameter (the number of convolution layers per bloc), it generates a different CNNs model. The method of generating CNNs model is based on an incremental technique, in which we create a base model (each base model called bloc) then we train this bloc using ELOBA_$\lambda$ (see algorithm 2) until the bloc stops learning new relevant features. At this time we add another bloc depends on the results (Dice score) then we go back to the beginning and so on until we obtain an efficient model (i.e. model gives more accurate results). Our algorithm for generating CNNs models called Incremental XCNet (see algorithm 1) which is different than the other CNNs models. Incremental XCNet merges between creating and training deep learning model at the same time. After getting trained models we adopt the technique of Ensemble Learning which is a process of combining several models to obtain a better predictive performance. In our case, Ensemble Learning aggregates the results of different models (2CNet, 3CNet) to give us a new model called EnsembleNet which gives more accurate results. We develop each CNNs model (2CNet, 3CNet) in parallel then at test time we combine the results of these trained CNNs models using a non-parametric fusion equation (see the equation 4).

Our methods Incremental XCNet (see algorithm 1) and ELOBA_$\lambda$ (see algorithm 2) are different from the trial-and-error process; in the field of machine learning, trial-and-error process takes a model then try all possible configurations of hyper-parameters without any guided approach, but our methods and our experiments suggest to put a roof to each of these hyper-parameters to avoid the combinatorial explosion problem and more branching in the search space. Using the trial-and-error process leads to getting a

high complexity and maybe we will not reach to the model that has a high performance. The method of Incremental XCNet and ELOBA_$\lambda$ belong to the new field of Artificial Intelligence called automated machine learning ([44], [45]). Automated machine learning attempts to design a new machine learning model without the intervention of users. In this paper, we propose a method Incremental XCNet with an optimizer (ELOBA_$\lambda$) for generating deep learning models with minimal user intervention.

All our models (2CNet and 3CNet) use a technique based on a 2D image patch, in which these models predict the pixel's class which is the center of the 2D patch. In addition, after several experiments on the size of the patch (e.g. patches with 28*28 pixels) that gives the best result, we found that the size (32*32) pixels works better with our CNNs models. The algorithm of Incremental XCNet takes as an input a (4*32*32) patch matrix in which the width*height is (32*32) and the number of channels is four: flair, T1, T1c, T2. The outputs are four different classes: healthy tissue, edema, non-enhancing (solid) core and necrotic (or fluid-filled) core, enhancing core. Our models (2CNet, 3CNet) are designed and trained from scratch based on the general rule that is taken from ILSVRC challenge and from ([15]): every time we add more layers, we will obtain high-performance models. Thus, instead of using kernels with large receptive field, we choose to use small filters. Small filters (3 * 3) have proved in ([14]) that they keep a lot of information, and at the same time they reduce the number of operations. Also, we double the number of filters after each pooling layer, these techniques allow the model to be more deeper in terms of layers. Additionally, detecting a high level and more complicated features needs more filters than the previews layer.

### *4.1. Incremental XCNet Algorithm*

The main idea of the Incremental XCNet Algorithm is based on adding a new bloc (bloc is a set of convolution and pooling layers) after each training phase. The algorithm of Incremental XCNet is described as follows:

The user sets the first base model (first bloc) that will be accumulated using the algorithm 1 Incremental XCNet. Then, the algorithm 1 in turn generates another larger model automatically based on the first base model. For example, the first base model is two convolution (2C) layers and one max-pooling layer (1P), so the first base model is 2C+1P, after the first iteration of training using the algorithm 2 ELOBA_$\lambda$, the algorithm 1 will add another two convolution layers and one max-pooling layer to the first base model, so the model will become 2C+1P+2C+1P, and so on for the next iterations.

The algorithm of Incremental XCNet is related with the algorithm of ElOBA_$\lambda$ (algorithm 2), in which unlike other CNNs-based models that define the architecture at the beginning then train it. Here, we focus on creating a CNNs model to give a high predictive performance, and at the same time, designing an optimized architecture in terms of layers. For achieving these goals, we follow a simple technique: in Algorithm 1:line 1 we create a base model (bloc) with a number of convolutions, i.e. X is the number of convolution layers and it is always fixed at the beginning. Then in Algorithm 1:line 2, we train this base model with the algorithm

**Algorithm 1: Incremental XCNet**

**Data:** X: the number of convolution layers, $S \in \{0, 1\}$: the number of pooling layers, F: the number of filters, FC: the number of fully connected layers

**Result:** M: a CNNs trained model

1 Create a bloc (X , S) and FC: $M_0 \leftarrow (X, S) + FC$ # Creating the first bloc

2 $M_0 = ELOBA_\lambda$ ( $M_0$ ) # Run the training using ELOBA_$\lambda$ to Adjust the Network's weights

3 **while** *(The new added bloc is not null)* **do**

4 Remove FC and freeze all layers of $M_{i-1(i\in\mathbb{N})} : M_{i-1(i\in\mathbb{N})} \leftarrow (X, S)$

5 Double the number of filters

6 Adding a new bloc (X , S) and FC: $M_{i(i\in\mathbb{N})} \leftarrow M_{i-1(i\in\mathbb{N})} + (X, S) + FC$

7 $M_{i(i\in\mathbb{N})} = ELOBA_\lambda$ ( $M_{i(i\in\mathbb{N})}$ )

8 **end while**

of ELOBA_$\lambda$, the first execution of (Algorithm 1:line 1 and line 2) from the algorithm Incremental XCNet, gives $M_0 = (X, S) + FC$, i.e. we have a known number of convolution and pooling layers and a number of fully connected layers. Then in Algorithm 1:line 4, we remove the fully connected layers (i.e. classifier) and freeze all layers of the last base model $M_i$ (i.e. the weights inside the layers will not update any more). Then, in Algorithm 1:line 5 we double the number of filters, i.e. when the number of features is small (and this number depends on the model's size) then S=0 (see figure 1 (a) bloc (4)), otherwise S=1. In Algorithm 1:line 6 we add a new bloc and fully connected layers to the last base model. Then in Algorithm 1:line 7, we continue the training using ELOBA_$\lambda$, after the first execution of (Algorithm 1:line 6 and line 7) we will get $M_1 = M_0 + (X, S) + FC$, i.e. besides $M_0$ we have a known number of convolution and pooling layers and a number of fully connected layers.

### *4.2. Extended Models*

For the development of our models, 2CNet (figure 1(a)) and 3CNet (figure 1(b)) are instances of the algorithm Incremental XCNet. Both models (2CNet and 3CNet respectively) produce complete predicted images (h and g respectively with size n*n). Then, next step is image fusion, in which we combine these two images (h and g) to get more accurate result (i.e. the final image). The non-parametric fusion function is performed using a new equation as follows:

$$f(i,j) = \begin{cases} I_{h(i,j)} & \text{if} \quad I_{h(i,j)} = I_{g(i,j)} \\ min(I_{h(i,j)}, I_{g(i,j)}) & \text{if} \quad I_{h(i,j)} \neq I_{g(i,j)} \end{cases} \tag{4}$$

where $I_{h(i,j)}$, (respectively $I_{g(i,j)}$) is the intensity of pixel h(i,j) (respectively g(i,j)), in which $\forall (i,j) \in [1,n]^2$.

The combination of Incremental XCNet's instances (i.e 2CNet and 3CNet) and the fusion operation, gives us at the end the EnsembleNet (figure 1(c)) model.

In this paper, we propose 3 Deep Learning models 2CNet, 3CNet, and EnsembleNet which aggregates the segmentation results that are obtained from 2CNet and 3CNet models by using a non-parametric fusion function. Moreover, the fusion function is used after getting the segmentation results not during the training time, the reason why we use the fusion function after getting the segmentation results is as follows:

- The fusion of two models (2CNet and 3CNet) in the fully connected layer during the training time (or testing time) will lead to obtaining a huge model with millions of parameters (weights), which obviously the model will require in this case a lot of computation power to train, otherwise, it will take a long time to converge (or to give the prediction results). Thus, the solution is to use the fusion function after getting the segmentation results to minimize the training and the prediction time.

- Glioblastomas have many connected regions (blobs) with different sizes and shapes, in which some research (see [37], in the related work section ) propose a fixed threshold (global threshold) operation as a post-processing to remove these false blobs (i.e. non-tumor region classified as tumor region) which are generated by a model but this fixed threshold also removes some small tumor regions. Overall, this fixed threshold could improve the segmentation results, but it is not a good method for patient-specific (i.e. for each patient), e.g. patient ID = Brats17_2013_13_1 from BraTS 2017 dataset, has 11 tumor regions inside his 3D MR volume ( 1 region with 20442 voxels, 7 regions with 1 voxels, 1 region with 2 voxels, 1 region with 4 voxels, 1 region with 32 voxels), if we apply a fixed threshold = 3000 voxels as used in [37], we will get only the first region. Thus, to address this issue of false blobs automatically we propose in this paper a non-parametric fusion function that aggregates predicted pixels that are true (tumor regions) with a high rate from other deep learning models 2CNet and 3CNet.

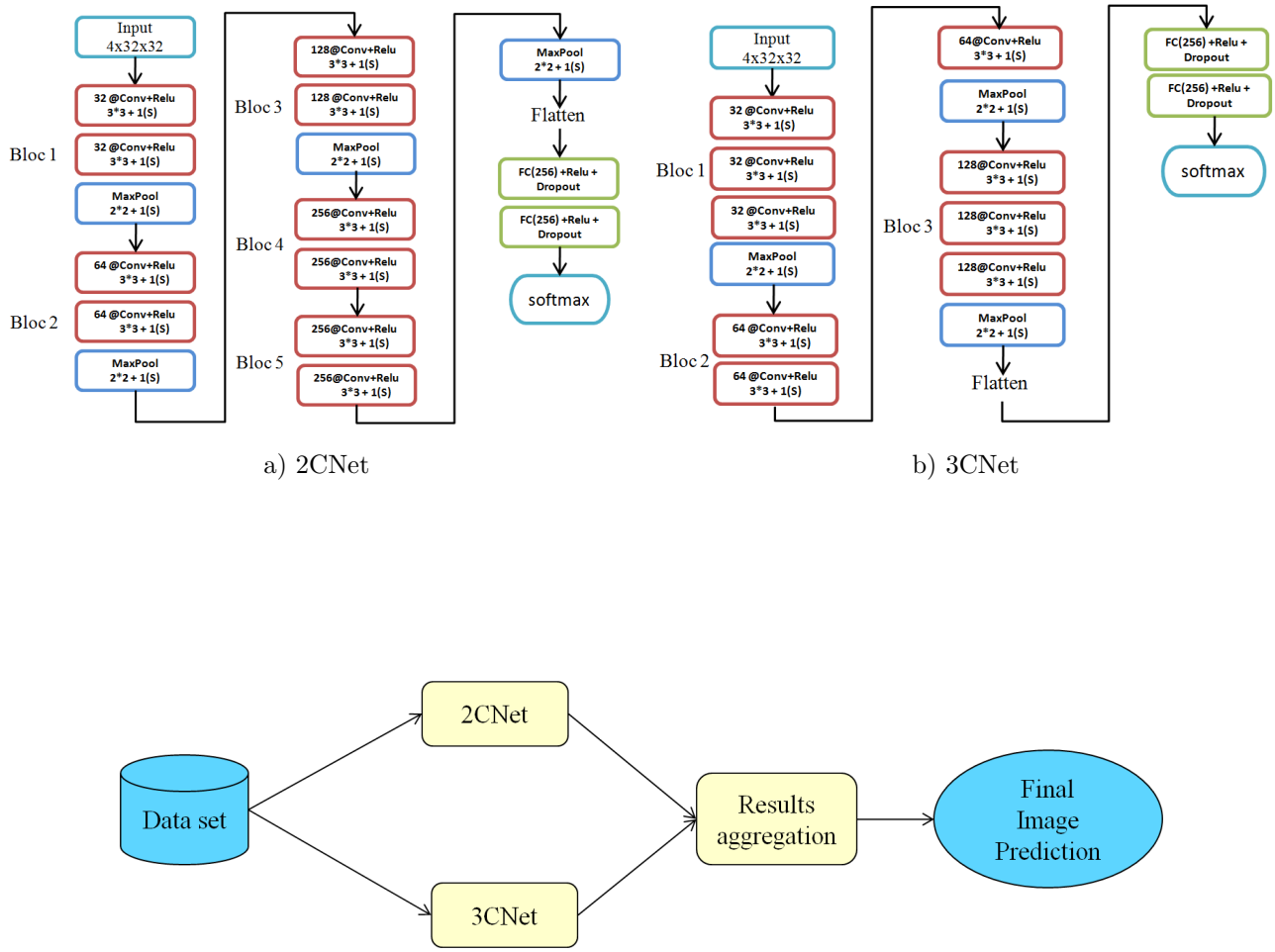


Figure 1: Schematic representation of our proposed CNNs architectures. (a) is the architecture of 2CNet, (b) is the architecture of 3CNet, (c) is the architecture of EnsembleNet. The symbols used inside the boxes indicate: Conv: convolution, #@: is the number of filters, Relu: is the non-linear function Relu, #*#: is the size of filter (or the size of sub-window in the case of pooling), S: stride, FC: fully connected layer.

## 5. The proposed training strategy

During the investigation of all methods that are used for the training of deep learning models, we noticed that there is no a specific method for the training of CNNs models. The users have many choices (i.e. hyper-parameters) that should be selected before creating the CNNs model with no guarantee that this CNNs model will succeed or fail. In addition, the used training method is not usually completely described. They use different hyper-parameters (i.e. the number of epochs, batch size, learning rate, optimizer...etc) for each CNNs model. For these reasons, we propose a new strategy to unify the training method of CNNs models that will be called: ELOBA_$\lambda$ (E: Epochs, L: Learning rate, O: Optimizer, B: Batch size, A: Architecture, $\lambda$ is the number of epochs). ELOBA_$\lambda$ is used for training our 2CNet and 3CNet models.

### *5.1. ELOBA_$\lambda$ Algorithm*

The algorithm of ELOBA_$\lambda$ (algorithm 2) takes into account the most influencing hyper-parameters (i.e. Epochs, Batch size, Learning rate, Optimizer). We create a base model architecture by algorithm 1, then we train (algorithm 2:line 5) this base model. Next step (algorithm 2:line 6) is calculating a Dice coefficient (see section 6.3), we mention that the Dice coefficient in the first execution is initialized to 0. If we get a better Dice score than the last iteration (algorithm 2:line 7), then we save the model (algorithm 2:line 9) and continue the training for another $\lambda$ epochs. If we get a worse Dice score than the last iteration, then we back to the hyper-parameters: we increase the Batch size (algorithm 2:line 4) then continue the training for $\lambda$ epochs and so on until the Batch size becomes greater than bs_max. In the same way, we decrease the Learning rate (algorithm 2:line 3) and continue the training. When the Learning rate becomes very small less than r_min (i.e. because small steps need a lot of time to converge), then we should change the optimizer[3] to another one (e.g. RMSprop, SGD) (algorithm 2:line 2) and initialize the Learning rate (algorithm 2:line 3) and the Batch size (algorithm 2:line 4) to the first state. The end criteria of the ELOBA_$\lambda$ algorithm is done when there is no improvement while trying all types of optimizers consecutively (i.e. Adam, RMSprop, SGD). In this case, we need to change the CNNs architecture which is done using the algorithm 1.

The algorithm of ELOBA_$\lambda$ is described as follows:

[3] See section 6.4 for further discussion on this point

**Algorithm 2: ELOBA_$\lambda$**

**Data:** $M$: an untrained CNNs base model, $\lambda$: Epochs, Bs: Batch size $\in [bs_min, bs_max]$, K: a positive integer , Lr: Learning rate $\in [r_min, r_max]$, R: a positive number, Op: $\in$ Optimizer = {Adam, RMSprop, SGD}

**Result:** $M$: a *trained* CNNs base model

```
for (j = 1 : j ← j+1 : j = 3) do
    Op = Optimiser (j)
    for (Lr = r_max : Lr ← Lr/R : Lr = r_min) do
        for (Bs = bs_min : Bs ← Bs + K : Bs = bs_max) do
            Training(M, λ, Bs, Lr, Op)   # Run the back-propagation with the hyper-parameters (λ,
             Bs, Lr, Op)
            Dice_t(M)                    # Compute the Dice coefficient
            while (Dice_t(M) > Dice_{t-1}(M) )   # Where Dice_{t-1}(M) is the Dice coefficient of the
             last iteration, in the first execution (Dice_0(M)) is equal to 0
            do
                Save the model M and its weights
                Training(M, λ, Bs, Lr, Op)
                Dice_t(M)
            end while
        end for
    end for
end for
```

### 5.2. ELOBA_$\lambda$ training tree

The main idea of the ELOBA_$\lambda$ Algorithm is based on exploring different search spaces: Optimizers, Batch size, Learning rate. This way helps to converge more toward the global minimum by trying different search techniques. In fact, ELOBA_$\lambda$ is used to unify the methods of training and to get the optimal hyper-parameters (i.e. epochs, batch size, learning rate, optimizer) to deep learning models.

Figure 2 shows the training method of ELOBA_$\lambda$ in order to explore different search spaces. As we can see, from a base model architecture at the higher-level node (i.e. $M$) each node (e.g. optimizer = Adam) after that discovers the range of lower-level nodes (e.g. Learning rate = r_max) to the leaf nodes (e.g. Batch size = bs_min). The algorithm of ELOBA_$\lambda$ attempts to find the global minimum by exploring three search spaces: optimizers, batch size, learning rate. As each path represents a potential hyper-parameter, so ELOBA_$\lambda$ tests the response of each path in this tree.

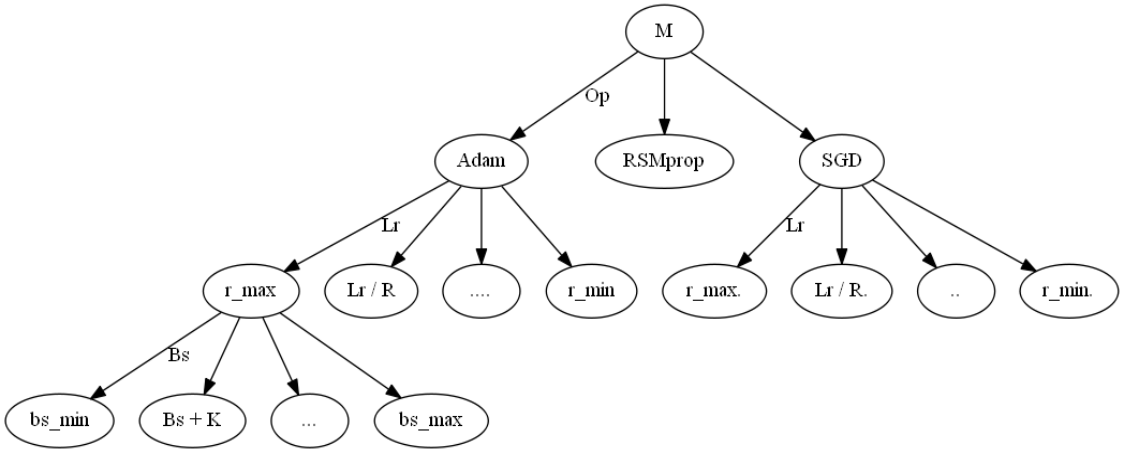


Figure 2: illustration of the training strategy ELOBA_$\lambda$. Where "M" is the model architecture, "Op" is the optimizer, "Lr" is the learning rate, "Bs" is the batch size.

## 6. Data description and experimental results

### *6.1. Dataset*

We evaluate our experiments on real patient's data, in which we have got this dataset from the BraTS-2017. BraTS's dataset has 210 patient's brain with high grade (i.e. High Grade HGG) and 75 patient's brain with low grade (i.e. Low Grade LGG). Each patient's brain image comes with 4 MR sequences (i.e. T1, T1c, T2, flair) and the Ground truth labels (figure 3) (made by experts).

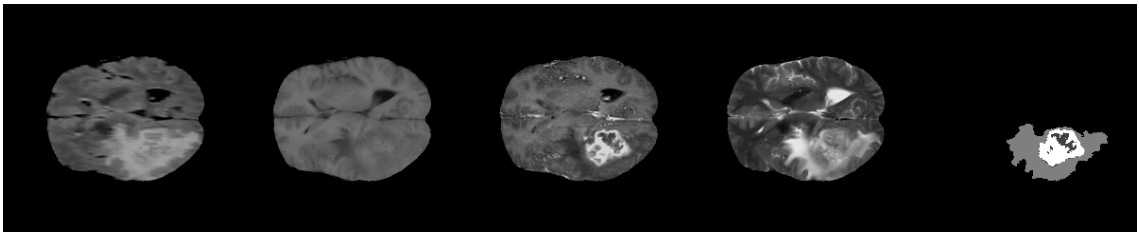

Figure 3: illustration of the four MRI modalities (i.e. four channels) used as inputs to our CNNs-based models (i.e. 2CNet, 3CNet). From left to right: FLAIR, T1, T1c, T2 and the last image is the Ground Truth labels: Healthy tissue (label 0), Non-enhancing core and necrotic core (label 1), Edema (label 2), Enhancing core (label 4)

### *6.2. Implementation*

For the implementation of our models (2CNet, 3CNet, EnsembleNet), we use Keras [4] and Theano as a backend. Keras is a high-level open source deep learning library that runs on top of Theano which can

[4]Keras:https://keras.io/

benefit from a massive parallel architecture such as GPU [5] to optimize the deep learning models. For the pre-processing step, we apply the minimum pre-processing as used in many image processing applications. The normalization of the MRI image intensities is performed as follows:

1. Remove the 1% highest and lowest intensities.
2. Subtract the mean value and divide by the standard deviation in all channels.
3. Select the slices that are greater than 100 pixels representatives of each class among the four classes. After several experiments to select the best number of pixels representatives, we chose 100 pixels because it provides the best results (in these slices, tumors appear in their largest size).

For the post processing, we do not have any post-processing step, in other words, the final image is the predicted image by the EnsembleNet model. Moreover, the training is done on Nvidia Quadro K1200.

### *6.3. Evaluation*

To obtain better and more efficient CNNs models, we put our models under a serie of different tests and different scenarios: image patch size, different pre-processing, and the number of patches used for training. For the evaluation of our models tumor segmentation performance, we use the evaluation metrics that used in BraTS: complete (i.e. necrosis and non-enhancing tumor, edema, enhancing tumor), core (i.e. necrosis and non-enhancing tumor, enhancing tumor), enhancing (i.e. enhancing tumor), the evaluation metrics are calculated as follows:

$$\text{Dice} = \frac{|P_1 \wedge T_1|}{(|P_1| + |T_1|)/2}, \quad \text{Sensitivity} = \frac{|P_1 \wedge T_1|}{|T_1|}, \quad \text{Specifisity} = \frac{|P_0 \wedge T_0|}{|T_0|}, \quad \text{Hausdorff} = \underset{a \in P_1}{max} \{ \underset{b \in T_1}{min} \{ d(a\ ,\ b) \}\}$$

Where $\wedge$ is the logical AND operator, P is the model predictions and T is the ground truth labels. $T_1$ and $T_0$ represent the true lesion area and the remaining normal area respectively. $P_1$ and $P_0$ represent the predicted lesion area and the predicted to be normal respectively. $(|.|)$ is the number of pixels belonging to this class. Moreover, 'a' and 'b' are points of sets $P_1$ and $T_1$ respectively, and d(a,b) is the distance (e.g. the Euclidian distance) between point 'a' and 'b'.

### *6.4. ELOBA_$\lambda$ hyper-parameters*

Based on many experiments, we obtain the following hyper-parameters:

- The optimizer (Op): "Adam", at the beginning of the ELOBA_$\lambda$ algorithm, we used "Adam" [46] as an optimizer because it is very fast than the other optimizers for converging toward the global minimum. The other optimizers that we used too: RMSprop and the last one is stochastic gradient descent (SGD) with Nesterov's Momentum.

[5] GPU: Graphics Processing Unit

- The learning rate (Lr): R= $10^{-1}$, R is the decreasing rate used to change the learning rate. r_max= $10^{-3}$, r_min= $10^{-8}$, the range [r_min, r_max] represents the search space (i.e. the space of all possible solutions) for the optimization algorithms. Thus, based on our experiments, the three chosen optimizers do not give guaranteed solutions out the range [r_min, r_max].

- The batch size (Bs): bs_min=32, bs_max=256, k=32, batch size is the number of training examples for one forward/backward pass. With the problem of non-convex functions, it is hard to get rid of the local minima points. Moreover, one of the techniques that influences the optimization of non-convex functions is the batch size, in which the use of small batch size leads to introduce some noise; this will helps to escapes from saddle points. And the use of big batch size (more than 256) leads to consume a lot of memory space. Thus, the best solution is the use of batch size in the range [32-256].

- The number of epochs $\lambda$= 20, where according to our experiments:

  - The algorithm 2 "ELOBA_$\lambda$" gives 20 epochs to each configuration of hyper-parameters. Thus, the first configuration of hyper-parameters for example, has 20 epochs, after getting a better result, the successful model will continue for another training with 20 epochs and so on, but if this model gets a worse Dice score then we update the hyper-parameters for the same model then we continue the training with the new hyper-parameters and so on. Thus, the algorithm 2 orients the training phase by testing the results (Dice score of the model) after every 20 epochs.

  - One of the biggest problems that we face with Deep Neural Network is the weights initialization. To address this issue, the algorithm 2 starts the training of a new deep learning model where the last model has stopped, in other words, the new model benefits from the last model in the initialization of weights. Thus, the algorithm 2 uses the first model to initialize the larger models.

  - The algorithm 2 rewards successful models that show an improvement of the Dice score during their first 20 epochs, and it also reduces the training phase of models with a low performance.

  - Using a high number of epochs with a decreasing rate (Lr/10 ) will lead the learning rate (Lr) to become infinitesimally small, obviously the model will take a long time to converge, and this point of weakness is well known in the Adagrad [47] algorithm. Thus, after many experiments we found that 20 epochs is enough for the new model to obtain a small improvement compared to the last model because the new model will start the training where the last model has stopped, not from scratch, if the new model obtains a better result (Dice score) it will get another 20 epochs with the same hyper-parameters to converge more and to give a better result and so on (see the algorithm 2 "ELOBA_$\lambda$", line 7 to line 12)

### 6.5. Results

In this section, we evaluate our models on a public dataset 2017 of BraTS. We show the results of our 3 models with the state-of-the-art models measuring both the segmentation performance and the inference time using three described metrics (see section 6.3). Unlike other CNNs-based models, our sequential architectures 2CNet, 3CNet have deep architectures which help to extract relevant and very high-level features. Moreover, EnsembleNet also takes the advantage of these deep architectures to build a parallel architecture that brings together the most important features of the same subject. Thanks to the ELOBA_$\lambda$ training strategy that uses a simple and practical idea based on the use of many optimizers over a bounded range of training space instead of using one optimizer as used in the conventional CNNs models.

The EnsembleNet's outputs are the "valid" pixels or pixels that are true with a high rate from both architectures 2CNet and 3CNet. Moreover, we have tested many non-parametric fusion functions, and the non-parametric function that gives more accurate results is described in Section 4. Overall, our non-parametric fusion function (see the equation 4) increases the Dice score over the complete region but at the same time it gives results less accurate over the core and the enhancing regions due to the fact that after getting the segmentation results by 2CNet and 3CNet, EnsembleNet model aggregates the results of 2CNet and 3CNets models using a non-parametric fusion function, this fusion function does not take any prior knowledge or any priority for any region in the predicted image, empirically this fusion function has improved the complete region compared to other regions because:

- The complete region holds all the other regions (core, enhancing). Thus, any improvement of any region will remains under the complete region and it will be also counted for the complete region.
- BraTS dataset is an unbalanced dataset, in which 98% of data are healthy tissue with label 0 and 1.1% of data are Edema with label 2 . In addition, it is well known that the healthy tissue class (label 0) and the edema class (label 2) do not belong to core or enhancing regions. Thus, any improvement of class two will be counted for the complete region.
- The Minimum function of the equation 4 updates any confusion between labels of the same pixel to the label that has the minimum value, which obviously the zero is the minimum value between all labels.
- The EnsembleNet model has been influenced toward the classes that have minority of existence in the dataset because the 4 classes are not equivalent in any subject (input MRI image). In addition, the class zero (healthy tissue) and the class two (edema) always win when there is a confusion of the same pixel.

Table 1 illustrates the results of the segmentation performance of different methods based on CNNs. As we can see, our models (2CNet, 3CNet, EnsembleNet) improved the Dice score compared to the state-of-the-art models such as the models of [36], [38], and [39]. Moreover, EnsembleNet outperforms the method of [40],

and [37] in terms of Dice score and it obtains competitive results on other evaluation metrics (Specificity, Sensitivity). Our three models are also evaluated on the Hausdorff distance metric, it can be observed that our models achieve good results (i.e. 13 mm to 20 mm), in which the Hausdorff distance: [14.62 mm - 17.59 mm], [17.40 mm - 20.25 mm], [12.04 mm - 16.39 mm] for whole tumor, tumor core, enhancing tumor core respectively. Based on the comparison of EnsembleNet with the state-of-the-art models, EnsembleNet gives the highest Dice score.

Table 1: Segmentation results of our three proposed models which are noted by adding "*" to the architecture's name with the state-of-the-art CNNS methods. WT, TC, ET denote Whole Tumor (complete), Tumor Core, Enhancing Tumor core respectively. Fields with (- , -/-) are not mentioned (given with a specific number) in the published work.

| Methods | Dice | | | Specificity | | | Sensitivity | | | Hausdorff | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WT | TC | ET | WT | TC | ET | WT | TC | ET | WT | TC | ET |
| **2CNet*** | **0.88** | **0.80** | **0.83** | **0.74** | **0.76** | **0.77** | **0.88** | **0.86** | **0.78** | **17.59** | **20.25** | **16.39** |
| **3CNet*** | **0.87** | **0.81** | **0.83** | **0.74** | **0.78** | **0.80** | **0.89** | **0.84** | **0.74** | **16.86** | **18,99** | **13.67** |
| **EnsembleNet*** | **0.89** | **0.76** | **0.81** | **0.74** | **0.77** | **0.78** | **0.82** | **0.82** | **0.69** | **14.62** | **17.40** | **12.94** |
| Zikic et al. [36] | 0.84 | 0.74 | 0.69 | - | - | - | - | - | - | - | - | - |
| Urban et al. [37] | 0.88 | 0.83 | 0.72 | - | - | - | - | - | - | - | - | - |
| Axel et al. [38] | 0.79 | 0.68 | 0.57 | - | - | - | 0.79 | 0.67 | 0.63 | - | - | - |
| Pereira et al. [39] | 0.87 | 0.73 | 0.68 | - | - | - | 0.86 | 0.77 | 0.70 | - | - | - |
| Havaei et al. [40] | 0.88 | 0.79 | 0.73 | 0.89 | 0.79 | 0.68 | 0.87 | 0.79 | 0.80 | -/- | -/- | -/- |

Figure 5 shows the pixel classification results obtained by 2CNet, 3CNet, EnsembleNet for the four MRI modalities in Figure 4. As we can see our CNNs models detect the tumor region and its three subregions: Red: necrosis and non-enhancing tumor, Green: edema, Yellow: enhancing tumor. This representative segmentation gives very promising results, in which we can improve it more in the future by adding more post-processing operations.

Figure 6 shows examples of the segmentation results for proving the utility of using non-parametric fusion function. As we can see 2CNet, 3CNet models detect many regions as tumor regions, in addition these models also generates small regions with different sizes which are classfied as tumor regions (red circles). To address this issue, we use a simple but effective method called non-parametric fusion function, by using this fusion function we are able to detect which region of these regions are actually tumor regions. Non-parametric fusion function acts as a voting function which elects only the tumor region with a high probability. EnsembleNet model (right column) aggregates the segmentation results of 2CNet and 3CNet

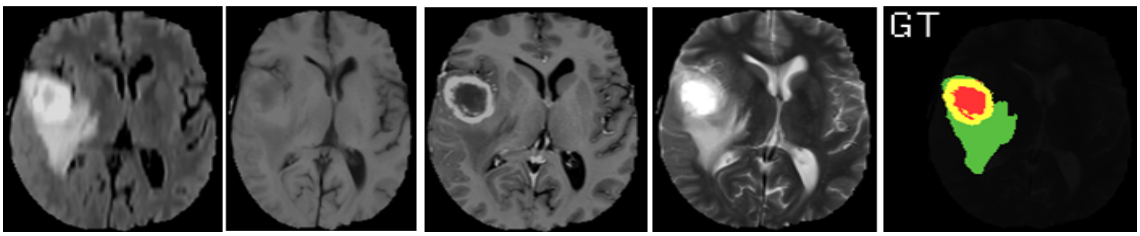


Figure 4: The four MRI modalities of a patient with Glioblastomas disease used as inputs to our CNN-based models. From left to right: Flair, T1, T1c, T2, GT (ground truth labels). The color is used to indicate to each region of the tumors: ■ necrosis and non-enhancing tumor, ■ edema, ■ enhancing tumor, ■ everything else (healthy tissue).

Figure 5: Visualization results of the pixels classification obtained by our CNNs models: 2CNet, 3CNet, EnsembleNet respectively.

models using this fusion function, as it can be observed that EnsembleNet detects only the tumor regions (blue circles) that are predicted by 2CNet and 3CNet in the first stage.

After the training phase, it comes the production phase where the trained model is used to infer/predict a new data. for benchmarking the models, inference time is used, i.e. as we said before inference time represents the needed time to make prediction on a new data. Thus, to deploy deep learning on a large scale, it is necessary to improve the deep learning inference time. EnsembleNet is the fastest method among all other CNNs-based models based on the inference time (see Table 2).

Table 2 presents a comparison of our models (2CNet, 3CNet and EnsembleNet) with the state-of-the-art CNNs models. We notice that our CNNs models (2CNet, 3CNet, EnsembleNet) outperform all the CNNs models cited in the table in terms of inference time (21.39, 19.7, 21.49 respectively), in which the time is decreased (21.49 seconds, almost eight times faster) compared to the state-of-the-art model of [40] (three minutes), and about three times faster than [37]. Furthermore, we cannot compare our models with the model of [36] because it is not mentioned in the published work. Moreover, we mention that each model in the table 2 uses a different GPU architecture with the exception of our models, in which we use the same GPU.

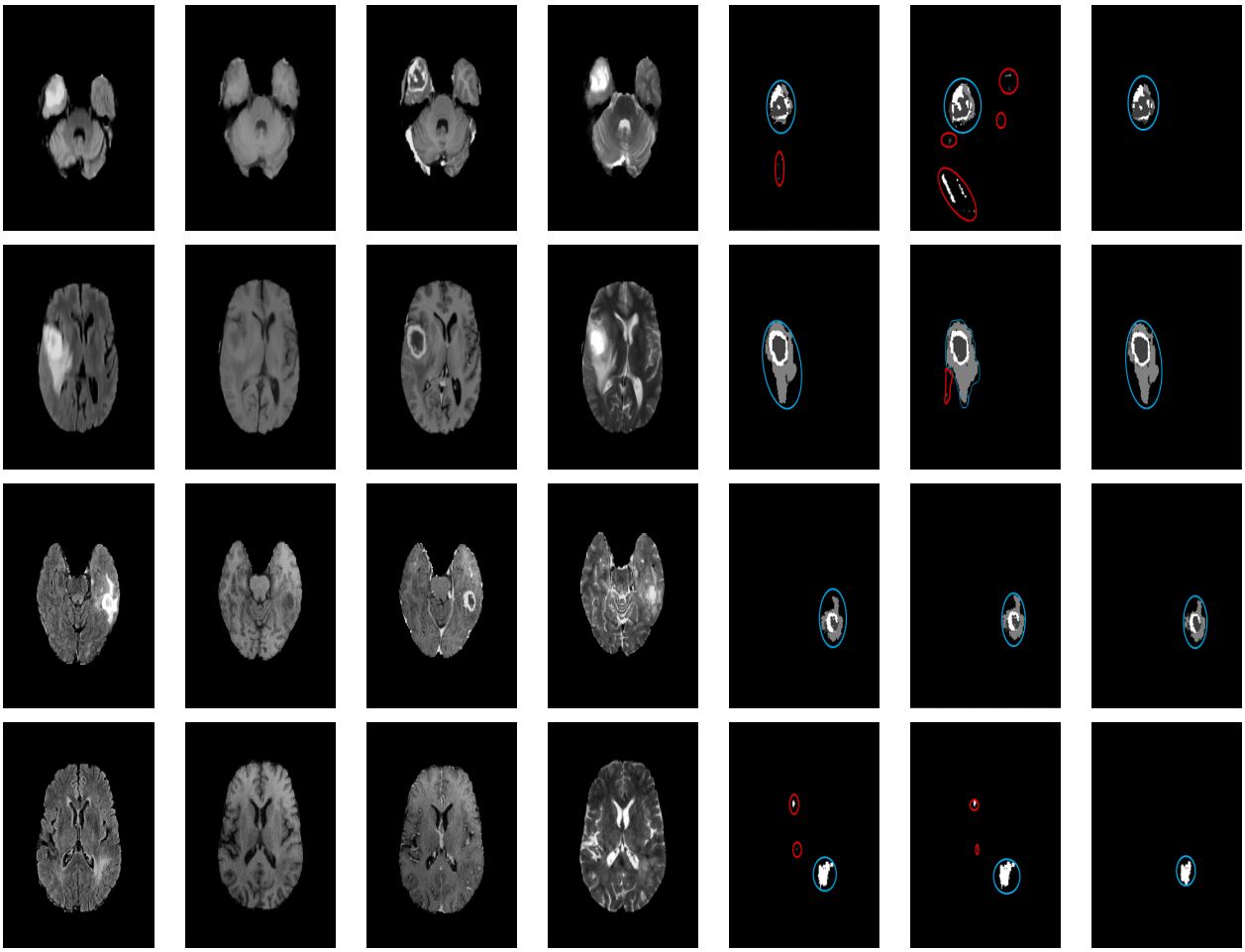

Figure 6: Examples of the segmentation results on the BRATS 2017 dataset to show the utility of using non-parametric fusion funcion. Each row represents segmentation results of a different slice from the axial view. The first and second row for the 42th and 77th slice of Subject "$Brats17_2013_10_1$". The third and fourth row for the 50th and 80th slice of Subject "$Brats17_CBICA_ABM_1$". From left to right: Flair, T1, T1c, T2, segmentation results of 2CNet, segmentation results of 3CNet, segmentation results of EnsembleNet. Red circle: non-tumor classified as tumor, blue circle: tumor regions.

Table 2: evaluation results of the state-of-the-art CNNs methods with our 3 proposed models which are noted by adding "*" to the architecture's name.

| Methods | Inference time (seconds) (architecture) |
|---|---|
| **2CNet*** | **21.39 (GPU)** |
| **3CNet*** | **19.72 (GPU)** |
| **EnsembleNet*** | **21.49 (GPU)** |
| Zikic et al. [36] | - (-) |
| Urban et al. [37] | 60 (GPU) |
| Axel et al. [38] | 1200 (GPU) |
| Pereira et al. [39] | 600 (GPU) |
| Havaei et al. [40] | 180 (GPU) |

## 7. Discussion

In this study, by using the algorithm 1 and algorithm 2, we developed different prediction models (2CNet, 3CNet, EnsembleNet) from the previous prediction models ([40], [37]). The model of [40] used an architecture based on deep neural networks in which this model exploits the local and the global features of the patches using two pathways. By using these two pathways in a CNNs model, their model leads to losing time during the first and second stages of fusion. However, our model EnsembleNet uses the parallel architecture technique but without losing time, in which EnsembleNet is the main process and it has 2 threads (2CNet, 3CNet) which run in parallel, and each model computes a different predicted image for the same subject (input image), then we aggregate these two images (image for each model) by using a non-parametric fusion function. Thus, the timing required by EnsembleNet is the maximum between the timing required by 2CNet (21.39 seconds) and the timing required by 3CNet (19.72 seconds) then we add the maximum to the timing required by the non-parametric fusion function (0.1 seconds). Thus, the timing required by EnsembleNet is 21.49 (21.39 + 0.1) seconds. Moreover, EnsembleNet needs only 21.49 seconds to segment the whole tumor and its subregions (see Table 2).

The development of 3D-CNNs model as used in [37], leads to getting a huge model that is hard to train it (i.e. during training, they need to compute the gradient for each 3D filter per pass), and it uses too much memory and a massive computing. In addition, this model needs 40 hours for training as they declared. While our model 2CNet (see section 4) reached the same complete Dice score with 2D patches instead of 3D patches.

Moreover, the use of specific operations in the preprocessing step: N4ITK bias correction [48] as used in ([40], [37] ), and the use the post-processing step: removing flat blobs near the skull based on connected components as used in the model of [40], leads to obtaining models depend significantly on the nature of data.

Thus, the performance and the efficient of these models are also based and depend on the pre-processing and on post-processing stages. However, by using the algorithm of Incremental XCnet (algorithm 1) and the algorithm of training ELOBA_$\lambda$ (algorithm 2), our model EnsembleNet uses the valid pixel-wise classification with a high rate from other models (2CNet, 3CNet) to predict the new image. Thus, the technique of valid pixel with a high rate replaces the operation of post-processing inside the algorithm without the need to use more operations after the obtaining of predicted image.

## 8. Conclusion

In this paper, we proposed three fully automatic methods (2CNet, 3CNet, EnsembleNet) for the segmentation of brain tumors using Deep Convolutional Neural Networks, in which we adopted the technique of ensemble learning with the power of CNNs, and the result is a new model we termed it as "EnsembleNet". The advantage of these three models: (i) these models are new incremental architectures; from a base model that is fed into our training strategy ELOBA_$\lambda$, we obtain a deeper model with a high performance. (ii) EnsembleNet takes the advantage of parallel architecture without losing time; 2CNet and 3CNet extract features at the same time from the same Glioblastomas MRI image then EnsembleNet aggregates the results of both models (2CNet and 3CNet) to get more accurate result. (iii) optimized architectures; Incremental XCNet algorithm allows our models to get the suitable architecture in terms of layers. (iv) these three models are end-to-end deep learning approaches, and fully automatic.

In addition, we developed a new training strategy ELOBA_$\lambda$. First, ELOBA_$\lambda$ proved itself as an effective strategy and a simple method, in which it is the first strategy to our knowledge, shows how to train a CNNs model. Second, our training strategy ELOBA_$\lambda$ is created for giving us the exact time when we should change the model's architecture. Finally, ELOBA_$\lambda$ is a dynamic and adaptable method, in which we can change all the hyper-parameters (i.e. the number of epochs, learning rate, batch size, optimizers) based on the workstation computing power and the architecture's size.

The most important part is that 2CNet, 3CNet, EnsembleNet models have achieved high accurate results: 0.88, 0.87, 0.89 Dice score respectively over the complete region, moreover, these models improved the evaluation metrics compared to the state-of-the-art, from the first side and from the other side the efficient design with the advantage of GPU implementation, allows our three deep learning models to segment the whole brain and save the patient's life in average 20.87 seconds. Thus, with these three models, we outperformed most state-of-the-art models in terms of accuracy and speed. As a perspective of this research, we intend to further investigate the ELOBA_$\lambda$ strategy, we will do more experiments on the other hyper-parameters. On the other hand, we will try other techniques of image fusion for combining several CNNs models. Moreover, we would like to see the effect of changing the outputs of 2Cnet and 3CNet models for different segmentation objective, i.e. changing the models from multi-classification to binary classification then retrain these models

again with 2 outputs instead of 4 outputs.

## 9. References

### References


[1] Amarjot Singh, Shivesh Bajpai, Srikrishna Karanam, Akash Choubey, and Thaluru Raviteja. Malignant brain tumor detection. *International Journal of Computer Theory and Engineering*, 4(6):1002, 2012.

[2] T Logeswari and M Karnan. An improved implementation of brain tumor detection using segmentation based on soft computing. *Journal of Cancer Research and Experimental Oncology*, 2(1):006–014, 2009.

[3] Eric C Holland. Progenitor cells and glioma formation. *Current opinion in neurology*, 14(6):683–688, 2001.

[4] Lev Bangiyev, Maria Camilla Rossi Espagnet, Robert Young, Timothy Shepherd, Edmond Knopp, Kent Friedman, Fernando Boada, and Girish M Fatterpekar. Adult brain tumor imaging: state of the art. In *Seminars in roentgenology*, volume 49, pages 39–52. Elsevier, 2014.

[5] Stephanie J Chiu, Xiao T Li, Peter Nicholas, Cynthia A Toth, Joseph A Izatt, and Sina Farsiu. Automatic segmentation of seven retinal layers in sdoct images congruent with expert manual segmentation. *Optics express*, 18(18):19413–19428, 2010.

[6] Vanderson Dill, Alexandre Rosa Franco, and Márcio Sarroglia Pinho. Automated methods for hippocampus segmentation: the evolution and a review of the state of the art. *Neuroinformatics*, 13(2): 133–150, 2015.

[7] Li Shen, Hiram A Firpi, Andrew J Saykin, and John D West. Parametric surface modeling and registration for comparison of manual and automated segmentation of the hippocampus. *Hippocampus*, 19 (6):588–595, 2009.

[8] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2015.

[9] Marcel Prastawa, Elizabeth Bullitt, Sean Ho, and Guido Gerig. A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8(3):275–283, 2004.

[10] Darko Zikic, Ben Glocker, Ender Konukoglu, Antonio Criminisi, C Demiralp, Jamie Shotton, Owen M Thomas, Tilak Das, Raj Jena, and Stephen J Price. Decision forests for tissue-specific segmentation of

high-grade gliomas in multi-channel mr. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 369–376. Springer, 2012.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. 2014.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015: 1–9, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298594.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 2016.

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[19] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 International Conference on Computer Vision, ICCV 2015, pages 1440–1448. IEEE, dec 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.169.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[21] Andreas Stadlbauer, Ewald Moser, Stephan Gruber, Rolf Buslei, Christopher Nimsky, Rudolf Fahlbusch, and Oliver Ganslandt. Improved delineation of brain tumors: an automated method for segmentation based on pathologic changes of 1h-mrsi metabolites in gliomas. *Neuroimage*, 23(2):454–461, 2004.

[22] Peter Gibbs, David L Buckley, Stephen J Blackband, and Anthony Horsman. Tumour volume determination from mr images by morphological segmentation. *Physics in Medicine & Biology*, 41(11):2437, 1996.

[23] Michael R Kaus, Simon K Warfield, Arya Nabavi, Peter M Black, Ferenc A Jolesz, and Ron Kikinis. Automated segmentation of mr images of brain tumors. *Radiology*, 218(2):586–591, 2001.

[24] Joshua E Cates, Ross T Whitaker, and Greg M Jones. Case study: an evaluation of user-assisted hierarchical watershed segmentation. *Medical Image Analysis*, 9(6):566–578, 2005.

[25] Marloes MJ Letteboer, Ole F Olsen, Erik B Dam, Peter WA Willems, Max A Viergever, and Wiro J Niessen. Segmentation of tumors in magnetic resonance brain images using an interactive multiscale watershed algorithm1. *Academic Radiology*, 11(10):1125–1138, 2004.

[26] Vicent Caselles, Francine Catté, Tomeu Coll, and Françoise Dibos. A geometric model for active contours in image processing. *Numerische mathematik*, 66(1):1–31, 1993.

[27] Joshua E Cates, Aaron E Lefohn, and Ross T Whitaker. Gist: an interactive, gpu-based level set segmentation tool for 3d medical images. *Medical image analysis*, 8(3):217–231, 2004.

[28] Aaron E Lefohn, Joshua E Cates, and Ross T Whitaker. Interactive, gpu-based level sets for 3d segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 564–572. Springer, 2003.

[29] Bjoern H Menze, Koen Van Leemput, Danial Lashkari, Marc-André Weber, Nicholas Ayache, and Polina Golland. A generative model for brain tumor segmentation in multi-modal images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 151–159. Springer, 2010.

[30] Nathan Moon, Elizabeth Bullitt, Koen Van Leemput, and Guido Gerig. Model-based brain and tumor segmentation. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 528–531. IEEE, 2002.

[31] Marcel Prastawa, Elizabeth Bullitt, Nathan Moon, Koen Van Leemput, and Guido Gerig. Automatic brain tumor segmentation by subject specific modification of atlas priors1. *Academic radiology*, 10(12): 1341–1348, 2003.

[32] Ezequiel Geremia, Olivier Clatz, Bjoern H Menze, Ender Konukoglu, Antonio Criminisi, and Nicholas Ayache. Spatial decision forests for ms lesion segmentation in multi-channel magnetic resonance images. *NeuroImage*, 57(2):378–390, 2011.

[33] Mehmet Ozkan, Benoit M Dawant, and Robert J Maciunas. Neural-network-based segmentation of multi-modal medical images: a comparative and prospective study. *IEEE transactions on Medical Imaging*, 12(3):534–544, 1993.

[34] Matthew C Clark, Lawrence O Hall, Dmitry B Goldgof, Robert Velthuizen, F Reed Murtagh, and Martin S. Silbiger. Automatic tumor segmentation using knowledge-based techniques. *IEEE transactions on medical imaging*, 17(2):187–201, 1998.

[35] Lynn M Fletcher-Heath, Lawrence O Hall, Dmitry B Goldgof, and F Reed Murtagh. Automatic segmentation of non-enhancing brain tumors in magnetic resonance images. *Artificial intelligence in medicine*, 21(1-3):43–63, 2001.

[36] Darko Zikic, Yani Ioannou, Matthew Brown, and Antonio Criminisi. Segmentation of brain tumor tissues with convolutional neural networks. *Proceedings MICCAI-BRATS*, pages 36–39, 2014.

[37] Gregor Urban, M Bendszus, F Hamprecht, and J Kleesiek. Multi-modal brain tumor segmentation using deep convolutional neural networks. *MICCAI BraTS (Brain Tumor Segmentation) Challenge. Proceedings, winning contribution*, pages 31–35, 2014.

[38] Davy Axel, Havaei Mohammad, Warde-Farley David, Biard Antoine, Tran Lam, Jodoin Pierre-Marc, Courville Aaron, Larochelle Hugo, Pal Chris, and Bengio Yoshua. Brain tumor segmentation with deep neural networks. *Proceedings MICCAI-BRATS*, pages 01–05, 2014.

[39] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A Silva. Deep convolutional neural networks for the segmentation of gliomas in multi-sequence mri. *Proceedings MICCAI-BRATS*, pages 52–55, 2015.

[40] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31, 2017.

[41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[42] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[43] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.

[45] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[48] Nicholas J Tustison, Brian B Avants, Philip A Cook, Yuanjie Zheng, Alexander Egan, Paul A Yushkevich, and James C Gee. N4itk: improved n3 bias correction. *IEEE transactions on medical imaging*, 29 (6):1310–1320, 2010.