MÉTHODES D'APPRENTISSAGE AUTOMATIQUE POUR LA SEGMENTATION DE TUMEURS AU CERVEAU

MACHINE LEARNING METHODS FOR BRAIN TUMOR SEGMENTATION

par

Seyed Mohammad Havaei

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 10 January 2017

Le 10 January 2017

Le jury a accepté le mémoire de Monsieur Seyed Mohammad Havaei dans sa version finale.

Membres du jury

Professeur Hugo Larochelle
Directeur de recherche
Département d'informatique, Université de Sherbrooke

Professeur Pierre-Marc Jodoin
Directeur de recherche
Département d'informatique, Université de Sherbrooke

Professeur Langis Gagnon
Évaluateur externe
Centre de Recherche Informatique de Montréal, Université de Montréal

Professeur Shengrui Wang
Prśident rapporteur
Département d'informatique, Université de Sherbrooke

Sommaire

Malignant brain tumors are the second leading cause of cancer related deaths in children under 20 [1]. There are nearly 700,000 people in the U.S. living with a brain tumor and 17,000 people are likely to loose their lives due to primary malignant and central nervous system brain tumor every year. To identify whether a patient is diagnosed with brain tumor in a non-invasive way, an MRI scan of the brain is acquired followed by a manual examination of the scan by an expert who looks for lesions (i.e. cluster of cells which deviate from healthy tissue). For treatment purposes, the tumor and its sub-regions are outlined in a procedure known as brain tumor segmentation . Although brain tumor segmentation is primarily done manually, it is very time consuming and the segmentation is subject to variations both between observers and within the same observer [138]. To address these issues, a number of *automatic* and *semi-automatic* methods have been proposed over the years to help physicians in the decision making process.

Methods based on machine learning have been subjects of great interest in brain tumor segmentation. With the advent of *deep learning* methods and their success in many computer vision applications such as image classification, these methods have also started to gain popularity in medical image analysis.

In this thesis, we explore different machine learning and deep learning methods applied to brain tumor segmentation.

Les tumeurs malignes au cerveau sont la deuxième cause principale de décès chez les enfants de moins de 20 ans [2]. Il y a près de 700 000 personnes aux États-Unis vivant avec une tumeur au cerveau, et 17 000 personnes sont chaque année à risque de perdre leur

1. www.abta.org
2. www.abta.org

vie suite à une tumeur maligne primaire dans le système nerveu central. Pour identifier de façon non-invasive si un patient est atteint d'une tumeur au cerveau, une image IRM du cerveau est acquise et analysée à la main par un expert pour trouver des lésions (c.-à-d. un groupement de cellules qui diffère du tissu sain).

Une tumeur et ses régions doivent être détectées à l'aide d'une segmentation pour aider son traitement. La segmentation de tumeur cérébrale et principalement faite à la main, c'est une procédure qui demande beaucoup de temps et les variations intra et inter expert pour un même cas varient beaucoup [138]. Pour répondre à ces problèmes, il existe beaucoup de méthodes *automatique* et *semi-automatique* qui ont été proposés ces dernières années pour aider les praticiens à prendre des décisions.

Les méthodes basées sur l'apprentissage automatique ont suscité un fort intérêt dans le domaine de la segmentation des tumeurs cérébrales. L'avènement des méthodes de Deep Learning et leurs succès dans maintes applications tels que la classification d'images a contribué à mettre de l'avant le Deep Learning dans l'analyse d'images médicales. Dans cette thèse, nous explorons diverses méthodes d'apprentissage automatique et de Deep Learning appliquées à la segmentation des tumeurs cérébrales.

Mots-clés: brain tumor segmentation, machine learning, deep learning, convolutional neural networks, medical image segmentation, computer aided diagnosis.

Remerciements

Foremost, I would like to express my sincere gratitude to my advisors Dr. Pierre-Marc Jodoin and Dr. Hugo Larochelle for their continuous support of my Ph.D study and research, for their patience, motivation, enthusiasm and immense knowledge. Their guidance helped me through out my research. I could not have imagined having better advisors for my Ph.D study.

Besides my advisors, I would like to thank Dr. Maxime Descoteaux and Jean-Christophe Houde for their insightful discussions and help with medical imaging tools.

My sincere thanks also goes to Dr. Christopher Pal, Dr. Alexandre Le Bouthillier and Dr. Nicolas Chapados for offering me internship opportunities and allowing me to work on diverse exciting projects.

I thank my fellow lab mates and friends in Université de Sherbrooke for the stimulating discussions, for the sleepless nights we were working together, and for all the fun we have had in the last four years.

Last but not least, I would like to thank my family for supporting me throughout my life. Without their love and encouragement, I would not have finished this thesis.

Table des matières

Table des figures

Liste des tableaux

Introduction

In terms of artificial intelligence, brain tumor segmentation is an interesting challenge that humans can learn to do efficiently, however, designing models with similar precision appears to be very challenging. This is due to the fact that humans use high-level features to localize and identify tumors [119]. This suggests that machine learning methods, in particular deep learning, can have a major impact in this application. In this work, we aim to provide beneficial tools using machine learning for brain tumor segmentation. A short introduction to machine learning methods used in this thesis is presented in Chapter 1. Magnetic resonance imaging (MRI) is briefly presented in Chapter 2 and some prior work on brain tumor segmentation is discussed in Chapter 3. More detailed discussions on challenges facing machine learning methods for brain tumor segmentation are discussed in Chapter 4.

In an effort to alleviate the need for excessive pre-processing steps, we present a semi-automatic method which is both fast and accurate while requiring little user interaction. This method is discussed in Chapter 5. While having a semi-automatic tool reduces the segmentation time compared to manual segmentation, the segmentation is still vulnerable to Inter-observer and intra-observer variability (i.e. mistakes made by the expert). Taking advantage of high-level features learned by deep learning, we present a fully automatic method for brain tumor segmentation which greatly reduces segmentation time and achieves high accuracy. This model is discussed in Chapter 6.

While having an accurate automated model solves a lot of issues, as in all machine learning methods, it requires a fixed number of input modalities. In an effort to relax that constraint, in Chapter 7, we present a hetero modality image segmentation model which is flexible to the input modalities it receives.

i

2

Chapter 1

Machine Learning

Machine learning is a domain in computer science which deals with the development of models that can learn from data. This is achieved by introducing examples of the data to the model through a *training* procedure. For the purpose of this thesis, we only consider a specific branch of machine learning, namely *supervised learning*. In this context, a *training example* is a pair of input observation $\mathbf{x}_i \in \Re^d$ and its corresponding target y_i. The set containing the training examples is known as the *training set*. The training examples are introduced to the model and the objective is for the model to extract patterns which describe the relationship between the training examples and their corresponding targets. With this training procedure, we expect the model to make a reasonable prediction ($\hat{y}_t$) given a previously unseen *test* example $\mathbf{x}_t$. A collection of test examples comprise a *test set*. The model's ability to make correct predictions on the test set is known as *generalization*. In practice (especially for small training sets), the variability of input data (e.g. variation in scale, rotation, illumination, etc.) is very large and examples in the training set do not represent the entire data distribution, which in turn makes the generalization suffer. Thus, it is common to map the input data to a representation space in the hope that pattern recognition would be easier. This practice is called *feature extraction* and can be done as a pre-processing step prior to learning (if we already know which features to extract) or using *deep learning* technology where the model learns the features it needs for the task at hand.

In this thesis we only focus on *classification* applications of machine learning. In classification problems, there exists a finite number (C) of individual classes and the goal is to learn a *classification function* which assigns input examples to different individual categories. In this setting y_i is the class label $l \in \{0, ..C-1\}$ to which the input example $\mathbf{x}_i$ is assigned to.

In machine learning, the classification function is estimated based on the training data. The *bias* of the estimated classification function is defined as the difference between the average prediction of the model and the true solution.The *variance* of a classifier is defined as the variability of a model prediction for a given data point. For good generalization, we expect the model to have small variance and small bias. If the model has a high bias then the classifier even fails to classify the training data. This is known as *underfitting*. If the model has high variance that means it is not robust to new examples and it has *overfitted* to the training examples. Generally high variance is the result of having too much *capacity*. The capacity of the model is a measure of complexity and flexibility and is in direct correlation with the number of free parameters in the model.

Model formulation can be divided into two groups based on model parametrization, namely; *non-parametric models* and *parametric models*. Non-parametric methods cover techniques that do not rely on data belonging to any particular distribution and thus, the number of parameters (i.e. capacity) is not a priori fixed as in parametric models. The computations required to obtain the model function f depend on the size of the training set and usually increase as the size of the training set increases. In parametric methods however, the classifier is a function of a fixed size parameter vector $\mathbf{w} \in \Re^d$. A typical example is a binary linear classifier which takes the following form :

$$a(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T\mathbf{x} + b, \tag{1.1}$$

$$f(\mathbf{x}; \mathbf{w}, b) = g(a(\mathbf{x}; \mathbf{w}, b)) \tag{1.2}$$

where b is a bias term and g is a non-linearity which either assigns a class label (such as the sign function) or assigns a class probability (such as the sigmoid function). In the remaining of this thesis for notation simplicity, $f(\mathbf{x}; \mathbf{w}, b)$ is referred to as $f(\mathbf{x})$. To optimize such models, a *loss function* (error function) over the parameter vector is defined which determines the amount of error the model makes when being presented

with training examples. The model updates its parameters in a way to reduce the loss. Thus, the training objective is to minimize the loss function:

$$\arg\min_{\mathbf{w},b} \frac{1}{N} \sum_i J(f(\mathbf{x}; \mathbf{w}, b_i), y_i), \tag{1.3}$$

where $\mathbf{w}$ is the set of all parameters.

The optimization problem in Equation 1.3 can be solved by *gradient descent*, where the model follows the negative direction of gradients in parameter space to find local minima. This is done by first computing the gradient of the loss function with respect to every parameter

$$\Delta = -\nabla_{\mathbf{w}} J(f(\mathbf{x}_i), y_i) \tag{1.4}$$

and updating the parameters as

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Delta, \tag{1.5}$$

where α is known as the *learning rate* and determines the step size between two updates (see Figure 1.2). The bias is updated in similar fashion. One issue with gradient descent is choosing the learning rate. If we use a fixed learning rate but set it too low, the optimization will be very slow (see Figure 1.3 (a)), but if we set it too high, the model might never converge (see Figure 1.3 (b)). A common practice would be to start from an initial learning rate and reduce the learning rate by some factor every few iterations of the model.

To reduce the zig-zags in Figure 1.2, a momentum term can be added to Equation 1.5. This can be written as

$$\mathbf{v} \leftarrow \mu\mathbf{v} + \alpha\Delta \tag{1.6}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}. \tag{1.7}$$

By introducing a velocity vector $\mathbf{v}$ the model has a memory of the previous update direction. The negative direction of the gradient is added to this velocity vector. If the gradient is in the opposite direction of the previous update, the velocity vector will prevent the parameters to wonder off severely which reduces the zig-zag effect. On the other hand, if the model starts to plateau, the velocity vector improves the optimization

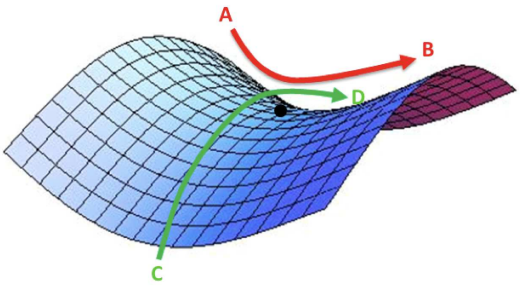


Figure 1.1 – A saddle point over a 2 dimensional error surface. We would like to increase velocity on $C \rightarrow D$ direction and decrease it on $A \rightarrow B$ direction. Figure from [18].

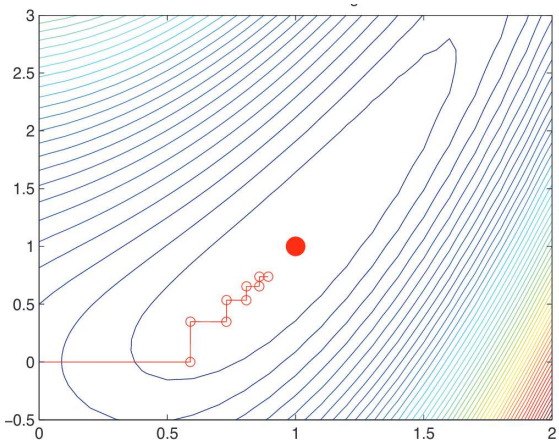


Figure 1.2 – Contour graph visualization of gradient descent on two dimensions of a parameter vector. Figure from [108].

by pushing the model to make larger update steps allowing the learning to proceed. Using momentum, every dimension in the parameter space will have its own velocity value. This can be advantageous in saddle points where a local maximum meets a local minimum and we want the learning rate to increase in some directions and decrease in others (See Figure 1.1).

One way to use the gradient descent algorithm is to update parameters at every training example. This is referred to as *stochastic gradient descent*. Since the gradients are estimated by only one training example, stochastic gradient descent often leads to noisy gradients. A common practice is to use the average gradients of a batch of training examples. This is referred to as *mini-batch gradient descent*. The number of training

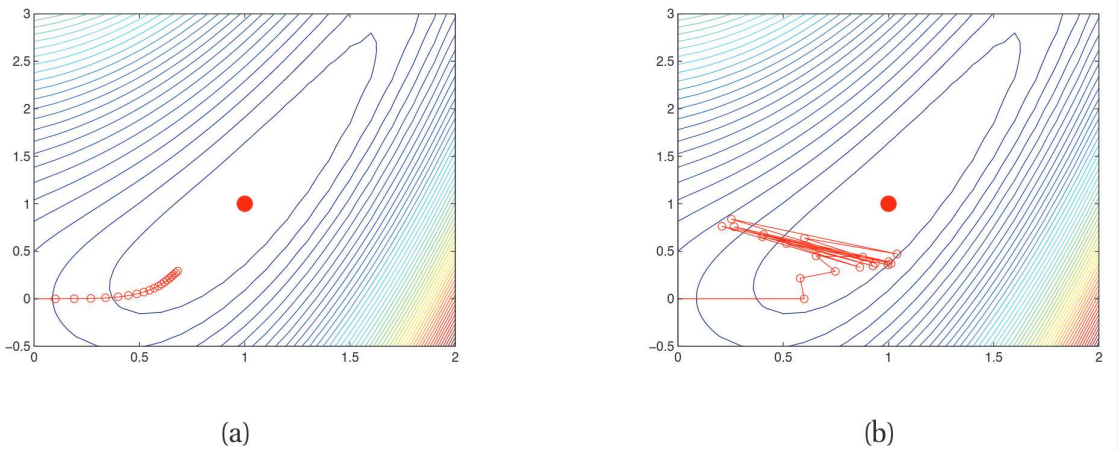


Figure 1.3 – Effects of the learning rate value on gradient descent optimization. Left: very small learning rate results in very slow convergence. Right: very large learning rate results in divergence. Figure from [108].

examples in one mini-batch can be different depending on the type of data. The number of iterations that takes for the model to go through the entire dataset is called an *epoch*.

If the model is not complex enough (e.g too few parameters) with respect to the size of the training set, it will not have enough capacity to extract discriminative patterns from the training set and the model would underfit. On the other hand, if the model is too complex with respect to the size of the training set, the model would have enough capacity to memorize every training example. This can cause the model to overfit to the training data resulting in very small training loss but very bad generalization. Finding the correct number of free parameters depends on the size of the training set and the complexity of the problem, making it necessary to be tuned case by case. Variables such as α or the number of free parameters which can change depending on the dataset, are referred to as *hyper-parameters* and often need to be tuned. This process is known as *model selection* and is performed as follows. First, a small set of training examples is selected to form the *validation set*. Then, a grid of all possible combinations of hyper-parameter values is formed and the model goes through them sequentially or randomly. The hyper-parameter combination which achieves the best accuracy on the validation set, is selected.

As discussed above, minimizing the training loss does not necessarily lead to good

generalization of the test set. The model can overfit only by training too much, thus it is important to stop the training at the correct time. For that purpose, the performance of the model on the validation set is measured after every epoch. We stop training when the accuracy of the model on the validation set starts to drop. This practice is referred to as *early stopping*. It is important to note that no training is performed on the validation set (i.e. the parameters of the model are not updated when the validation set is used.)

In what follows, some of the more relevant machine learning algorithms are described.

1.1 kNN

The k-Nearest Neighbor (kNN) algorithm is a non-parametric method often used for classification. There is no training phase associated to kNN and the training data is simply stored in memory to be used directly at test time. Given a test example, we calculate its distance to every training example and the k closest ones are chosen. To determine the class label of the test example $\mathbf{x}_t$, a vote is taken among the labels of the closest neighbors.

$$\hat{y}_t = \arg\max_c \frac{1}{k} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{N}} \delta(y_i, c), \tag{1.8}$$

where $\mathcal{N}$ is the set containing the k nearest neighbors of $\mathbf{x}_t$, $\delta(a, b)$ is equal to 1 if $a = b$ and 0 otherwise. k (i.e. the number of closest neighbors) is a hyper-parameter and needs to be tuned. If k is too small, the model may have high variance and is prone to overfitting. On the other hand if k is too large the model might underfit. Because the amount of computations increases with the size of the training set, kNN is a poor choice for very large datasets.

1.2 SVM

The support vector machine (SVM) [35] is a linear max-margin binary classifier. It tries to find a linear hyperplane to maximize the margin between the two classes [1]. The SVM

1. SVM can be generalized to more than two classes. The N-class case will be discussed later in this chapter

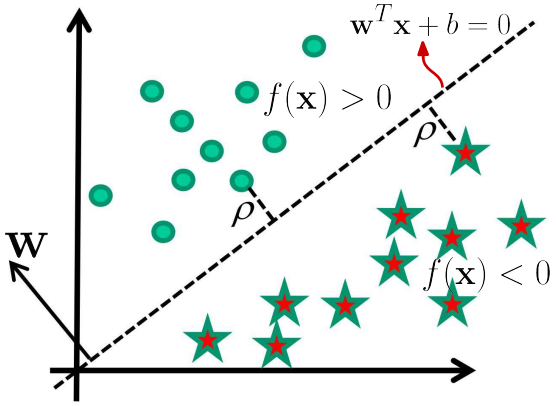


Figure 1.4 – Linear SVM visualization (Figure from [77]).

classifier is a linear classifier where :

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{1.9}$$

and we would like $f(\mathbf{x})$ to be such that :

$$f(\mathbf{x}_i) = \begin{cases} \geq 0 \text{ if } y_i = +1 \\ < 0 \text{ if } y_i = -1 \end{cases},$$

i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification. The SVM tries to satisfy a max-margin property giving the model an advantage of being robust (i.e. low variance). The SVM maximizes the margin ρ by trying to minimize the distance between training points $\mathbf{x}_i$ and the hyperplane of $\mathbf{w}$ (see Figure 1.4). This is formulated by the following equation.

$$\rho = \min_{i=1,..N} |\frac{f(\mathbf{x}_i)}{\|\mathbf{w}\|}|, \tag{1.10}$$

where $|\frac{f(\mathbf{x}_i)}{\|\mathbf{w}\|}|$ can be shown to be the distance between the hyperplane and point $\mathbf{x}_i$. The objective is thus to find the parameter vector bfw that maximizes the margin ρ.

$$\max_{\mathbf{w} \in \Re^d} \rho$$

subject to

$$\rho = \min_{i=1,..N} |\frac{f(\mathbf{x}_i)}{\|\mathbf{w}\|}| \text{ and } y_i f(\mathbf{x}_i) \geq 1 \; \forall i \; . \tag{1.11}$$

One can prove that this optimization criteria can be rewritten as [88] :

$$\min_{\mathbf{w} \in \Re^d} \|\mathbf{w}\|^2 \tag{1.12}$$

subject to

$$y_i f(\mathbf{x}_i) \geq 1 \; \forall i. \tag{1.13}$$

Equation 1.13 assumes that the data is linearly separable, which means that it does not allow for outliers. To make the model flexible with respect to outliers, the notion of *soft-margin* is introduced which allows examples to be wrongly classified at the expense of a penalty cost. The soft-margin criteria leads to the following formula :

$$\min_{\mathbf{w} \in \Re^d, \; \xi_i \in \Re^+} \|\mathbf{w}\|^2 + \frac{C}{N} \sum \xi_i \tag{1.14}$$

subject to

$$y_i(f(\mathbf{x}_i)) \geq 1 - \xi_i \; \forall i. \tag{1.15}$$

In Equation 1.14, the model allows $\mathbf{x}_i$ to violate the original constrains in Equation 1.12 by ξ_i. If $0 < \xi_i < 1$, $\mathbf{x}_i$ violates the margin but is still classified correctly. If however, $\xi_i > 1$, $\mathbf{x}_i$ would be classified incorrectly. This flexibility comes at a cost of adding ξ_i to the loss function. C is a hyper-parameter which controls the trade off between correctness and robustness. Small C allows constraints to be easily ignored while large C makes the constraints hard to ignore.

One can prove that Equation 1.14 is equivalent to [88]:

$$\min_{\mathbf{w} \in \Re^d} \frac{1}{N} \sum \max(0, 1 - y_i(f(\mathbf{x}_i))) + \|\mathbf{w}\|^2, \tag{1.16}$$

where $\max(0, 1 - y_i(f(\mathbf{x}_i)))$ is known as the *hinge loss* [128]. Equation 1.16 is referred to as the *primal problem* and can be solved by quadratic programming. Since hinge loss is piece wise linear, a variant of gradient descent algorithm known as sub-gradient

descent can also be used for optimization [88]. In the primal problem the classifier is a function of parameter vector $\mathbf{w}$ and b, as in Equation 1.1. Alternatively, based on the representer theorem [139], the SVM can be formulated to learn a linear classifier of the form

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i \, y_i(\mathbf{x_i}^T \mathbf{x}), \tag{1.17}$$

by solving an optimization problem over α_i. This is known as the *dual problem*

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i \frac{1}{2} \sum_{ij} \alpha_j \alpha_k y_j y_k (\mathbf{x}_i^T \mathbf{x}_j) \tag{1.18}$$

subject to

$$0 \leq \alpha_i \leq C \; \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

where α_is are known as Lagrange multipliers and C is a regularization term which bounds the possible size of the Lagrange multipliers. At a first glance, the dual problem seems similar to kNN, where at test time we need to have access to the entire training set. However, a lot of the Lagrange multipliers will be very close to zero in the final (learnt) solution. The $\mathbf{x}_i$ with non-zero α_i will be the *support vectors*. In other words, the support vectors of an SVM are training examples for which the coefficient α is not zero.

A linear SVM refers to a linear classifier and is very effective when the data is linearly separable. For the cases where the data is not linearly separable, there are two solutions. The first approach is to project the data in a feature space $\phi(\cdot)$ (typically of higher dimension) where it can be linearly separated. In this case, Equation 1.18 can be formulated as:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i \; \frac{1}{2} \; \sum_{ij} \alpha_j \alpha_k y_i y_j (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)) \tag{1.19}$$

subject to

$$0 \leq \alpha_i \leq C \; \forall i, \text{ and } \sum_i \alpha_i y_i = 0.$$

Since $\phi(\mathbf{x})$ appears in pairs, we can replace the dot product between the two feature

vectors by a kernel function where:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \tag{1.20}$$

This is known as the kernel trick. Kernel trick is used in the SVM dual problem formulation. An advantage of the kernel trick is that the optimization problem is independent of the dimensionality of the parameter vector $\mathbf{w}$, which can be very beneficial in very high dimensional spaces. A choice for the kernel that often proves successful is the radial basis function (RBF) kernel:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(\text{-}\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel_2^2). \tag{1.21}$$

where γ is a hyper-parameter [88] that can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The resulting classifier effectively takes the form of a template matcher, that compares a given input with all training examples, each voting for their class with a weight related to their similarity with the input (as modeled by the kernel). In this sense, it is similar to the kNN classifier, though the former often outperforms the later in practice. In the SVM formulation, the primal problem where the optimization is done with respect to the parameter vector $\mathbf{w}$ is regarded as a parametric model, while the dual problem where the optimization is independent of $\mathbf{w}$ but rather depends on the training data, is regarded as a non-parametric model.

It is possible to generalize the 2-class SVM to support multiple classes using the *one versus all* approach [125]. In this approach, C different SVMs are trained (one for every class). As shown in Figure 1.5, $f_c(\mathbf{x})$ for $c \in \{1, \cdots, C\}$ separates class c from other classes. At test time, the classifier which achieves the maximum score, defines the class label for the query example.

1.3 Artificial Neural Networks

An artificial neural network is a parametric model which is inspired from the human nervous system. In the following, a progression time line of neural networks is pre-

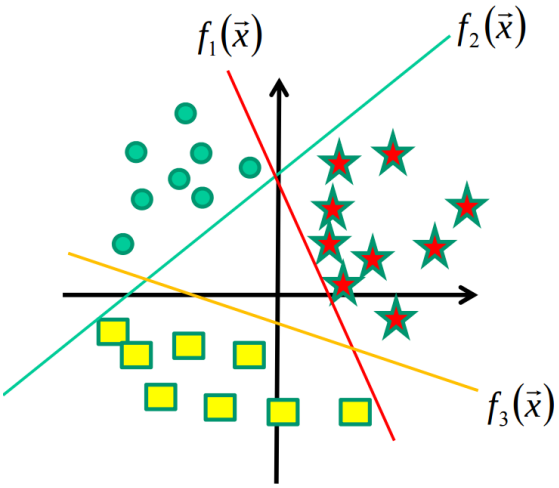


Figure 1.5 – One versus all SVM. For every class c, a classifier is trained to separate class c from other classes. Figure from [77].

sented.

1.3.1 Perceptron

The Perceptron [129] is a linear binary classifier. The first implementation on customized hardware is known to be the first artificial neural network. The Perceptron model is defined as follows :

$$f(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b) \tag{1.22}$$

where the activation function g is defined as

$$g(n) = \begin{cases} +1 & \text{if } n \text{ is } \geq 0 \\ -1 & \text{other wise .} \end{cases}$$

The loss is defined as a sum over wrongly classified training examples and is referred to

as the *Perceptron criterion*.

$$E_P(\mathbf{w}) = -\sum_{i \in \mathcal{M}} y_i(\mathbf{w}^T \mathbf{x}_i + b), \tag{1.23}$$

where $y_i \in \{+1, -1\}$ and $\mathcal{M}$ is the set of all wrongly classified examples. E_P is a linear function of $\mathbf{w}$ in regions of $\mathbf{w}$ where examples are wrongly classified and E_P is zero if all examples are correctly classified. Therefore, E_P is piecewise linear with respect to $\mathbf{w}$. This allows us to optimize E_P using sub-gradient descent.

Being piecewise linear, $\nabla_{\mathbf{w}} E_P(\mathbf{w}) = \sum_i y_i \mathbf{x}_i$. This means provided that we use sub-gradient descent, at every iteration of the model and for all training examples, if $\mathbf{x}_i$ is correctly classified the parameter vector $\mathbf{w}$ remains unchanged. If however, $\mathbf{x}_i$ is wrongly classified, $\eta y_i \mathbf{x}_i$ will be added to the parameter vector. Having this in mind, if the training data is not linearly separable, the algorithm will not converge. However, if the training data is linearly separable, the algorithm is guaranteed to find a solution. Since the loss function is piece wise linear it might have more than one solution and which one is found will depend on the initialization of parameters and the order which the training data is presented to the model.

The Perceptron does not provide a probabilistic output due to its use of the sign function. If we use an activation function such as the sigmoid, the output of the model would be a value between 0 and 1 which can be interpreted as the probability of $\mathbf{x}$ belonging to class 1. If we show class 0 and 1 as c_0 and c_1, then :

$$p(y = c_1|\mathbf{x}) = \text{sigm}(\mathbf{w}^T \mathbf{x} + b) = f(\mathbf{x}) \tag{1.24}$$

and since the conditional distribution $y|\mathbf{x}$ is a Bernoulli distribution, then :

$$p(y = c_0|\mathbf{x}) = 1 - \text{sigm}(\mathbf{w}^T \mathbf{x} + b) = 1 - f(\mathbf{x}), \tag{1.25}$$

where the sigmoid function is defined as:

$$\text{sigm}(x) = \frac{1}{1 + e^{-x}}.$$

Equations 1.24 and 1.25 can be written more compactly as :

$$p(y|\mathbf{x}) = f(\mathbf{x})^y(1 - f(x))^{1-y}, \tag{1.26}$$

where it is desirable to maximize this likelihood. An equivalent minimization problem would be to minimize the *cross entropy* which is defined as the negative log of the likelihood:

$$l(f(\mathbf{x}), y) = -y \log f(\mathbf{x}) - (1 - y) \log(1 - f(\mathbf{x})). \tag{1.27}$$

The sigmoid function can also be interpreted as computing the class posterior probabilities $p(c|\mathbf{x})$ through the Bayes theorem :

$$p(y = c_1|\mathbf{x}) = \frac{p(\mathbf{x}|c_1)p(c_1)}{p(\mathbf{x}|c_1)p(c_1) + p(\mathbf{x}|c_0)p(c_0)} = \frac{1}{1 + e^{-\alpha}} = \text{sigm}(\alpha), \tag{1.28}$$

where $\alpha = \sum_i \log \frac{p(x_i|c_1)p(c_1)}{p(x_i|c_0)p(c_0)} = \sum_i w_i x_i + b$ is a weighted sum of the input. Therefore, the posterior probabilities is equivalent to Equation 1.24.

Figure 1.6 shows the architecture of a single neuron which is considered a building block of any neural network [2]. Variables in this figure are computed as follows:

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^T \mathbf{x} \tag{1.29}$$

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i \, x_i), \tag{1.30}$$

where $\mathbf{w}$ is the weight vector containing connection weights w_i, b is the bias and $g(\cdot)$ is the activation function, $a(\mathbf{x})$ is known as the *pre-activation* and $h(\mathbf{x})$ is the output of the neuron. The following are the most common activation functions :

$$\text{Sigmoid:} \quad \text{sigm}(x) = \frac{1}{1 + e^{-x}}$$

2. Note that if $g(\cdot) = \text{sign}(\cdot)$, the architecture Figure1.6 would present a Perceptron.

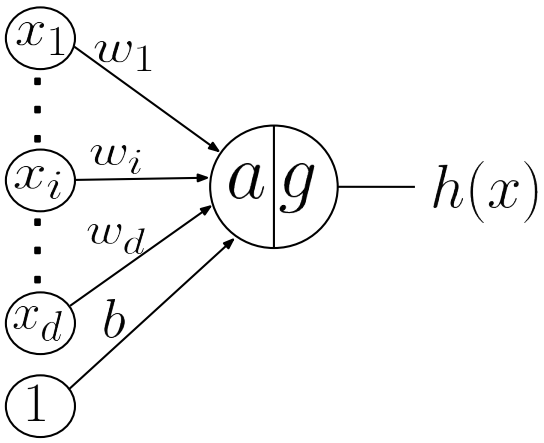


Figure 1.6 – An artificial neuron. The dot product of a d dimensional input vector and a parameter vector of the same dimensions is added with the bias and passed trough a non-linearity g to obtain $h(x)$.

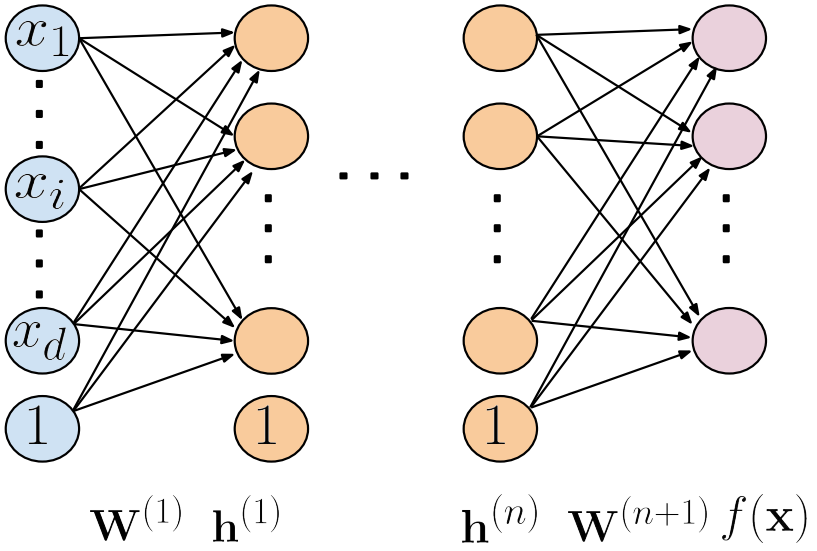


Figure 1.7 – Architecture for multi-layer Perceptron. Every layer is a function of the previous layer, making deep architectures feasible.

$$\text{Hyperbolic tangent:} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\text{Rectified linear unit:} \quad \text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$

A single neuron with a sigmoid activation function can be interpreted as a binary classifier which estimates $p(y = c_1|\mathbf{x})$ (i.e. the probability of $\mathbf{x}$ belonging to class 1). However, the capacity of a single neuron is limited to linear decision boundaries thus, making it a a linear classifier. One way to get around this difficulty is to use a network (or collection) of neurons to make an intermediate (*hidden*) representation of the input which can be linearly separated. This idea gives rise to multi-layered neural networks (see Figure 1.7). In the following, the equations for 2 layer neural networks with sigmoid hidden layer is laid out.

$$\mathbf{a}^{(1)}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x},$$

where

$$a(\mathbf{x})_i^{(1)} = b_i^{(1)} + \sum_j w_{i,j}^{(1)} x_j$$

$$\mathbf{h}^{(1)}(\mathbf{x}) = \mathrm{g}(\mathbf{a}^{(1)}(\mathbf{x})) \tag{1.31}$$

$$f(\mathbf{x}) = \mathrm{sigm}(b^{(2)} + \mathbf{w}^{(2)^T}\mathbf{h}^{(1)}(\mathbf{x})). \tag{1.32}$$

In a neural network the first layer is referred to as the input layer, the last layer is referred to as the output layer and all other intermediate layers are known as *hidden layers*. Neurons in a hidden layer are referred to as *hidden units*.

Features extracted from a single layer neural network, are low level features [177]. To obtain higher level representations of the input vector, more layers can be added. At every layer, representations (i.e. features) from the previous layer are combined with a set of weights to encode a more abstract representation. Provided that the network is deep enough, the representations from the final hidden layer are regarded as high level features extracted from the input. With n hidden layers, the *forward* pass through the

network comprises of the following equations:

$$f(\mathbf{x}) = \text{sigm}(\mathbf{a}^{(n+1)}(\mathbf{x})), \tag{1.33}$$

$$\mathbf{a}^{(n+1)}(\mathbf{x}) = b^{(n+1)} + \mathbf{w}^{(n+1)^T}\mathbf{h}^{(n)}(\mathbf{x}) \tag{1.34}$$

$$\mathbf{h}^{(n)}(\mathbf{x}) = \text{g}(\mathbf{a}^{(n)}(\mathbf{x})) \tag{1.35}$$

$$\mathbf{a}^{(n)}(\mathbf{x}) = \mathbf{b}^{(n)} + \mathbf{W}^{(n)}\mathbf{h}^{(n-1)}(\mathbf{x}) \tag{1.36}$$

where $f(\mathbf{x})$ is the output of the model. In case of mutli-class, its a vector of the size of the number of classes and its noted by $\mathbf{f}(\mathbf{x})$.

Using Equation 1.33, classifier $f(\mathbf{x})$ is a binary classifier. In the case of multiple classes, the output layer contains as many neurons as there are classes and for its activation function, *softmax* is used. The softmax function is defined as follows:

$$p(y = c|\mathbf{x}) = f(\mathbf{x})_c = \frac{e^{(b_c^{(n+1)} + \mathbf{w}_c^{(n+1)}\mathbf{h^{(n)}(x)})}}{\sum_{j=1}^{C} e^{(b_j^{(n+1)} + \mathbf{w}_j^{(n+1)}\mathbf{h^{(n)}(x)})}}, \text{for } c = 1, \cdots, C \tag{1.37}$$

If C is the number of classes, the output of the softmax is a vector of size C and can be interpreted as the probability of input vector $\mathbf{x}$ belonging to each class. The operation in Equation 1.37 is applied on all elements of the pre-activation output which results in $\mathbf{f}(\mathbf{x})$. Using softmax, the output of the model is :

$$\mathbf{f}(\mathbf{x}) = \text{softmax}(\mathbf{a}^{(n+1)}(\mathbf{x})). \tag{1.38}$$

To account for multiple classes, the loss function in Equation 1.27 can be expanded to C classes as follows:

$$l(f(\mathbf{x}), y) = -\sum_c \mathbf{1}_{(y=c)} \log f(\mathbf{x})_c = -\log f(\mathbf{x})_y. \tag{1.39}$$

In Equation 1.39, $c \in \{1, \cdots, C\}$ and the sum is over all possible labels that y can take.

The parameters of a neural network can be updated with a gradient descent algorithm. The gradients for model parameters are computed through the *backpropagation algorithm* [134]. At the heart of the backpropagation algorithm, lies the chain rule. According to the chain rule:

$$\text{if } y = f(u), \text{and } u = g(x) \text{ then } \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}. \tag{1.40}$$

As seen from Equation 1.35, every layer in a neural network is a function of its previous layer. For backpropagation, first the gradient of the loss with respect to the output layer is computed (i.e. $\nabla_{\mathbf{f(x)}} l$, where l is the loss function defined in Equation 1.39). This gradient is propagated through the rest of the network through the chain rule. The backpropagation equations are described in what follows:

As a first step, the partial derivative of the loss with respect to the output is computed as

$$\frac{\partial}{\partial f(\mathbf{x})_c} l(y, f(\mathbf{x})) = \frac{-1_{(y=c)}}{f(\mathbf{x})_y}$$

the gradient which contains the partial derivatives is written as: as

$$\nabla_{\mathbf{f(x)}} l(y, f(\mathbf{x})) = \frac{-\mathbf{e}(y)}{f(\mathbf{x})_y},$$

where $\mathbf{e}(y)$ is a onehot vector containing zero elements at every location other than the y^{th} element which is set to 1.

Using the chain rule we can then compute the partial derivatives of the output before the activation as

$$\frac{\partial}{\partial a^{(n+1)}(\mathbf{x})_c} l(y, f(\mathbf{x})) = -(1_{(y=c)} - f(\mathbf{x})_c)$$

and it's gradient as :

$$\nabla_{\mathbf{a}^{(n+1)}(\mathbf{x})} l(y, f(\mathbf{x})) = -(\mathbf{e}(y) - \mathbf{f}(\mathbf{x})).$$

In a similar way, using the chain rule, the partial derivatives for the variables in the k^{th} layer of the neural network can be derived as :

$$\frac{\partial}{\partial h^{(k)}(\mathbf{x})_j} l(y, f(\mathbf{x})) = \sum_i \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k+1)}(\mathbf{x})_i} \frac{\partial a^{(k+1)}(\mathbf{x})_i}{\partial h^{(k)}(\mathbf{x})_j} = \sum_i \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k+1)}(\mathbf{x})_i} W_{ij}^{(k+1)}$$

$$\frac{\partial}{\partial a^{(k)}(\mathbf{x})_j} l(y, f(\mathbf{x})) = \frac{\partial l(y, f(\mathbf{x}))}{\partial h^{(k)}(\mathbf{x})_j} \frac{\partial h^{(k)}(\mathbf{x})_j}{\partial a^{(k)}(\mathbf{x})_j} = \frac{\partial l(y, f(\mathbf{x}))}{\partial h^{(k)}(\mathbf{x})_j} g'(a^{(k)}(\mathbf{x})_j)$$

$$\frac{\partial}{\partial W_{ij}^{(k)}(\mathbf{x})_j} l(y, f(\mathbf{x})) = \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k)}(\mathbf{x})_i} \frac{\partial a^{(k)}(\mathbf{x})_i}{\partial W_{ij}^{(k)}} = \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k)}(\mathbf{x})_i} h_j^{k-1}(\mathbf{x})$$

$$\frac{\partial}{\partial b_i^{(k)}} l(y, f(\mathbf{x})) = \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k)}(\mathbf{x})_i} \frac{\partial a^{(k)}(\mathbf{x})_i}{\partial b_i^{(k)}} = \frac{\partial l(y, f(\mathbf{x}))}{\partial a^{(k)}(\mathbf{x})_i}.$$

The partial derivatives can be generalized to vectors of gradients. From here we can propagate the gradients layer by layer until we reach the first hidden layer, by computing for k from $n+1$ to 1 :

$$\nabla_{\mathbf{W}^{(k)}} l(y, f(\mathbf{x})) = \nabla_{\mathbf{a}^{(k)}(\mathbf{x})} l(y, f(\mathbf{x})) \mathbf{h}^{(k-1)}(\mathbf{x})^T$$

$$\nabla_{\mathbf{b}^{(k)}} l(y, f(\mathbf{x})) = \nabla_{\mathbf{a}^{(k)}(\mathbf{x})} l(y, f(\mathbf{x}))$$

$$\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} l(y, f(\mathbf{x})) = \mathbf{W}^{(k)^T} \nabla_{\mathbf{a}^{(k)}(\mathbf{x})} l(y, f(\mathbf{x}))$$

$$\nabla_{\mathbf{a}^{(k-1)}(\mathbf{x})} l(y, f(\mathbf{x})) = (\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} l(y, f(\mathbf{x}))) \odot [..., g'(a^{(k-1)}(\mathbf{x})_j), ...],$$

where $\odot$ denotes element wise multipication.

1.4 Convolutional neural networks

Convolutional neural networks (CNN) are a type of NN adopted for spatially or temporally ordered input. The main building block used to construct a CNN architecture is the *convolutional layer*. As in a regular NN, several convolutional layers can be stacked on

top of each other forming a hierarchy of features. Each layer can be understood as extracting features from its preceding layer in the hierarchy. A single convolutional layer takes as input a stack of input planes and produces as output some number of output planes or *feature maps*. Each feature map can be thought of as a topologically arranged map of responses of a particular spatially local non-linear feature extractor (the parameters of which are learned), applied identically to each spatial neighborhood of the input planes in a sliding window fashion. In the case of a first convolutional layer, the individual input planes correspond to different input channels. In the case of MRI, it can be different image modalities and in the case of color images it can be different color channels. In subsequent layers, the input planes typically consist of the feature maps of the previous layer. Computing a feature map in a convolutional layer (see Figure 1.8) consists of the following three steps:

1. *Convolution of kernels (filters):* Each feature map $\mathbf{O}_s$ is associated with one kernel (or several, in the case of Maxout [3]). The feature map $\mathbf{O}_s$ is computed as follows:

$$\mathbf{O}_s = b_s + \sum_r \mathbf{W}_{sr} * \mathbf{X}_r \tag{1.41}$$

where $\mathbf{X}_r$ is the r^{th} input channel, $\mathbf{W}_{sr}$ is the sub-kernel for that channel, $*$ is the convolution operation and b_s is a bias term [4]. In other words, the affine operation being performed for each feature map is the *sum* of the application of R different 2-dimensional $N \times N$ convolution filters (one per input channel/modality), plus a bias term which is added pixel-wise to each resulting spatial position. The convolutional operation of image $\mathbf{X}$ and kernel $\mathbf{W}$ is computed as:

$$\mathbf{C}_{ij} = (\mathbf{W} * \mathbf{X})_{ij} = \sum_m \sum_n \mathbf{X}_{i+m,j+n} \mathbf{W}_{-m,-n}. \tag{1.42}$$

In the above equation, the region in matrix $\mathbf{X}$ which is used in computation of $\mathbf{C}_{ij}$ is referred to as the *local receptive field* for $\mathbf{C}_{ij}$ and so $\mathbf{C}_{ij}$ is only connected to its receptive field, rather than the whole image as it was the case with MLPs. This

3. Maxout will be discussed later in this chapter.
4. Since the convolutional layer is associated to R input channels, $\mathbf{X}$ contains $M \times M \times R$ gray-scale values and thus each kernel $\mathbf{W}_s$ contains $N \times N \times R$ weights. Accordingly, the number of parameters in a convolutional block, consisting of S feature maps is equal to $R \times M \times M \times S$.

greatly reduces the number of parameters of the model. This receptive field is slided across the entire image. For each receptive field, there is a different hidden neuron (i.e. $\mathbf{O}_{s,ij}$). However, the weights to compute every hidden neuron is shared. This further reduces the parameters of the model by a factor of the number of neurons in that feature map. Intuitively, the reason for sharing parameters is that each kernel can be thought of as a feature detector that tries to identify that particular feature at different spatial positions in the image. Also, by sharing parameters, we can greatly reduce the parameters of the model and reduce risk of overfitting.

Whereas traditional image feature extraction methods rely on a fixed recipe (sometimes taking the form of convolutions with a linear filter bank), the key to the success of convolutional neural networks is their ability to learn the weights and biases of individual feature maps, giving rise to data-driven, customized, task-specific dense feature extractors. These parameters are learned via stochastic gradient descent on a surrogate loss function, with gradients computed efficiently via the backpropagation algorithm.

Special attention must be paid to the treatment of border pixels by the convolution operation. One option is to employ the so-called *valid* mode convolution, meaning that the filter response is not computed for pixel positions that are less than $\lfloor N/2 \rfloor$ pixels away from the image border. An $M \times M$ input convolved with an $N \times N$ filter patch, will result in a $Q \times Q$ output, where $Q = M - N + 1$. In Figure 1.8, $M = 7$, $N = 3$ and thus $Q = 5$. Note that the size (spatial width and height) of the kernels are hyper-parameters that must be specified by the user. One can apply the convolutions in *same* mode to preserve the input size. In this mode, zero padding is applied around the input prior to the convolution operation.

2. *Non-linear activation function:* To obtain features that are non-linear transformations of the input, an element-wise non-linearity is applied to the result of the kernel convolution. There are multiple choices for this non-linearity, such as the sigmoid, hyperbolic tangent and rectified linear functions [74], [52] or maxout [53].

   Maxout features are associated with multiple kernels $\mathbf{W}_s$. This implies each Maxout map $\mathbf{Z}_s$ is associated with K feature maps : $\{\mathbf{O}_{Ks}, \mathbf{O}_{Ks+1}, ..., \mathbf{O}_{Ks+K-1}\}$.

Maxout features correspond to taking the max over the feature maps $\mathbf{O}$, individually for each spatial position:

$$Z_{s,i,j} = \max \{O_{Ks,i,j}, O_{Ks+1,i,j}, ..., O_{Ks+K-1,i,j}\} \tag{1.43}$$

where i, j are spatial positions. Maxout features are thus equivalent to using a convex activation function, but whose shape is adaptive and depends on the values taken by the kernels. ReLU function can be considered a special form of Maxout where the max operation is taken over every feature map and a zero matrix of the same size for each spatial position (i.e. $\max(\mathbf{O_s}, \mathbf{0})$).

$$Z_{s,i,j} = \max \{O_{s,i,j}, 0_{i,j}\}. \tag{1.44}$$

Note that in Figure 1.8, the ReLU activation function is used.

3. *Max pooling:* This operation consists of taking the maximum feature (neuron) value over sub-windows within each feature map. This can be formalized as follows:

$$H_{s,i,j} = \max_p Z_{s,Si+p,Sj+p}, \tag{1.45}$$

where p determines the max pooling window size and S is the stride value which corresponds to the horizontal and vertical increments at which pooling sub-windows are positioned. Depending on the stride value, the sub-windows can be overlapping or not (Figure 1.8 shows an overlapping configuration). The max-pooling operation shrinks the size of the feature map. This is controlled by the pooling size p and the stride hyper-parameter. Let $Q \times Q$ be the shape of the feature map before max-pooling. The output of the max-pooling operation would be of size $D \times D$, where $D = (Q-p)/S+1$ [5]. In Figure 1.8, since $Q = 5, p = 2, S = 1$, the max-pooling operation results into a $D = 4$ output feature map. The motivation for this operation is to introduce invariance to local translations. This subsampling procedure has been found beneficial in other applications [86].

5. Note that values p and S should be chosen in a way that the pooling window fits the feature map (i.e. D should be an integer). Alternatively, we can zero pad the feature map Q accordingly.

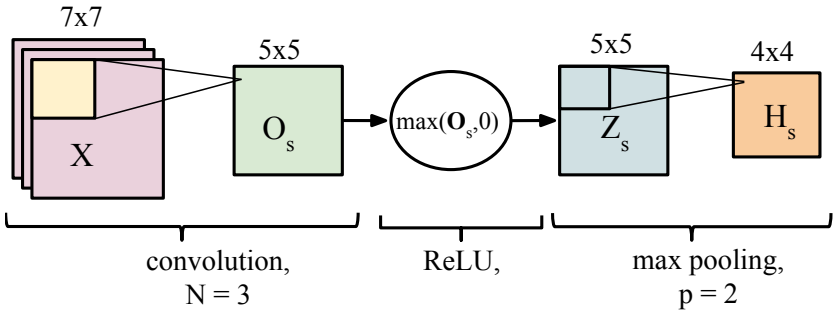


Figure 1.8 – A single convolution layer block showing computations for a single feature map. The input patch (here 7×7), is convolved with a series of kernels (here 3×3) followed by ReLU and max-pooling.

1.5 Regularization

Regularization refers to a technique used in an attempt to alleviate the overfitting problem in statistical models. As mentioned previously, when the model is too complex (i.e. has too much capacity) with respect to the size of the training data, it becomes prone to overfitting. In this section we describe common techniques to deal with overfitting.

1.5.1 L2 and L1 regularization

In L2 and L1 regularization, the weights are penalized by adding a regularization function $R(w)$ to the loss, as seen in Equation 1.46.

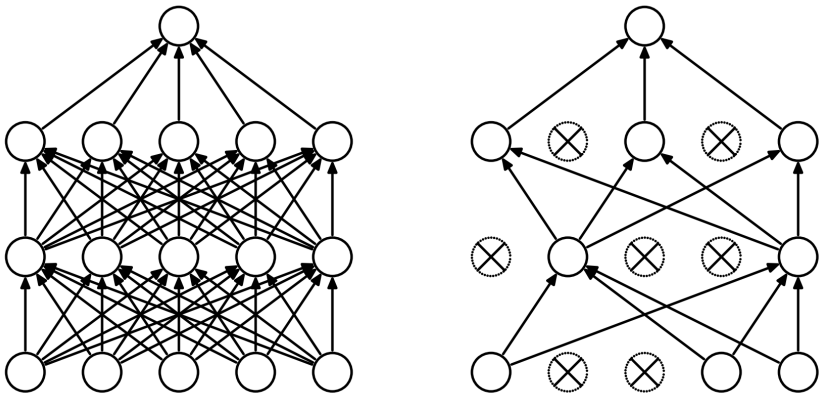
$$\arg\min_{\mathbf{w}} \frac{1}{N} \sum_{t} J(f(\mathbf{x}_i; \mathbf{w}), y_i) + R(w) \tag{1.46}$$

The general intuition is to prevent the model to have large weights in the hope of achieving smooth classification boundaries. For the regularization function $R(w)$ we can use the L2 loss which can be thought of as having a Gaussian prior over the weights as shown in Equation 1.47.

$$R(w) = \| \mathbf{w} \|^2, \tag{1.47}$$

L2 loss encourages the weights to have small values.

The L1 regularization, which can be interpreted as a Laplacian prior over the weights



(a) Standard Neural Net (b) After applying dropout.

Figure 1.9 – Dropout. Each neuron is masked with a probability of p. Figure from [148].

as shown in Equation 1.48, encourages sparsity in the weights. The Laplacian density function puts more mass at 0 and in the tails compared to the Normal distribution. This shows the greater tendency of L1 regularization to produce weights that are large or exactly 0 [157].

$$R(w) = \sum_i |\mathbf{w}|_i. \tag{1.48}$$

1.5.2 Dropout

Dropout has proven to be an effective regularization technique [147]. During training, the activation of every neuron is kept with a probability p or set to zero otherwise and only the parameters of the kept neurons are updated (see Figure 1.9). This can be interpreted as sampling from the full model different sub-models. At test time, all the weights are used and so the neurons see all their inputs. It is desirable that the outputs of neurons at test time be identical to their expected outputs at training time. To achieve this goal, the weights are scaled by p. This can be interpreted as averaging the sampled sub-models. Dropout can be thought of ensembling many *thinned* sub-models which can lead to avoiding overfitting and better generalization

Chapter 2

Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI), is an imaging technique used to investigate the anatomy of the body. MRI is based on the principles of Nuclear Magnetic Resonance (NMR). NMR is the study of the behaviour of atomic nuclei once in a magnetic field, and the frequencies they come into resonance with an electromagnetic field. Hydrogen nuclei also known as protons, have magnetic properties due to their spin motion. Each proton acts like a rotating magnet, which produces a magnetic field represented as a vector. In a normal environment, the protons spin in random directions, thus the direction of the magnetic vectors are randomly distributed. This results into the sum of all spins being zero which is also known as a null net magnetization.

When an external magnetic field (B_0) is applied, nuclear spins would either align in parallel or anti-parallel with the field. Since there are more spins aligned parallel to B_0, the net magnetization vector is in the direction of B_0. The spins wobble about B_0 with an angular frequency ω_0 defined as :

$$\omega_0 = \gamma B_0, \tag{2.1}$$

where γ is called the gyromagnetic ratio and it is a particle-specific constant incorporating size, mass, and spin. This wobbling effect is called precession. The magnetic vector of each spinning proton can be broken down into two parts: a longitude Z com-

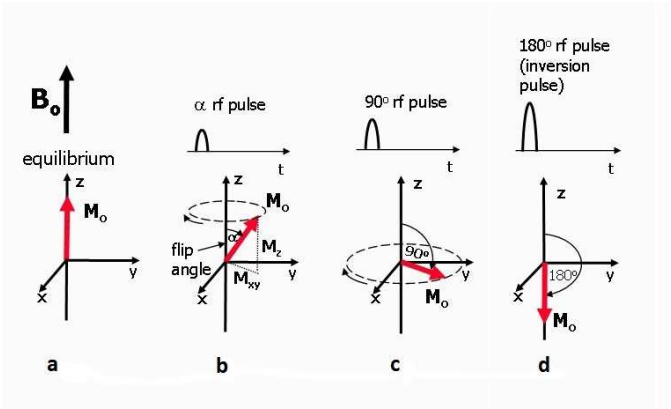


Figure 2.1 – Net magnetization. a) At equilibrium. b) When rf pulse is applied. c) At 90 rf pulse. d) At 180 rf pulse. Figure from [124].

ponent (M_z) and a transverse component (M_{xy}) as shown in Figure 2.1. Precession corresponds to the rotation of the transverse component about the longitudinal axis.

The sum of the longitudinal component of all spins is in the B_0 direction. This is the equilibrium (low energy) state for longitudinal components. The sum of all transverse components is null which means they are out of phase. This is the equilibrium state for transverse components.

By applying an electromagnetic field with a frequency equal to the frequency of the precession, it is possible to change the net magnetization from equilibrium state. This is called excitation. Magnetic resonance corresponds to the energy interaction between spins and electromagnetic radio frequency(RF). This would cause the net magnetization to rotate about B_0. As shown in Equation 2.2, the rotation angle α depends on the duration of the applied electromagnetic field (τ), and also its magnitude (B_1).

$$\alpha = 2\pi\tau\gamma B_1. \tag{2.2}$$

A $90°$ pulse rotates the magnetization vector $90°$ down on the xy plane. $180°$ pulse, rotates the magnetization vector $180°$ down along the $-z$ axis (see Figure 2.1). This electromagnetic RF pulse can be generated by running a current in a coil in the direction of the x axis. When the pulse is stopped, the system returns to equilibrium. This process is known as relaxation. During relaxation, the proton releases the electromag-

netic energy which induces an electromagnetic signal in the coil. Relaxation combines two different mechanisms:

— Longitudinal relaxation which corresponds to the recovery of M_z from 0 to its original value at the equilibrium.

— Transverse relaxation which corresponds to the decay of transverse magnetization.

The longitudinal relaxation is characterized by time $T1$ and transverse relaxation time is characterized by time $T2$, where $T2 \leq T1$. Both $T1$ and $T2$ follow an exponential curve (see Figure 2.2).

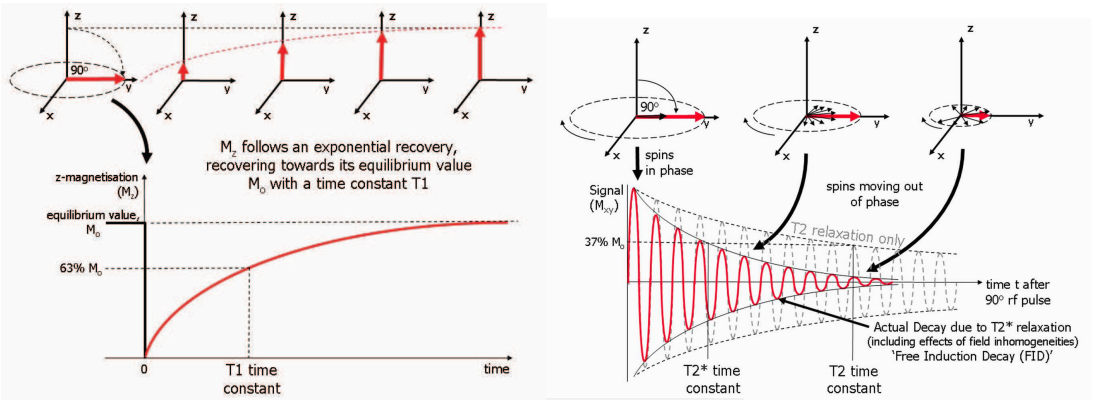


Figure 2.2 – Left, $T1$ relaxation time. Right, $T2$ and $T2^*$ relaxation time. Figure from [124].

In an inhomogeneous magnetic field where the distribution of the magnetic field is not uniform (i.e. the magnetic field is stronger in some locations and weaker elsewhere), protons spin with different frequencies. This will cause much faster transverse magnetization decay (dephasing). In this case, the 90° relaxation time is indicated as Free Induction Decay (FID) and characterized by $T2^*$. The $T2^*$ relaxation time is very short and therefore results in a very noisy Fourier transform. This is due to a Fourier transform property that a thinner signal in time domain would have a wider spectrum in Fourier domain and since the area under spectrum is constant, it would result in a lower magnitude of the spectrum and thus a noisy effect (see Figure 2.3). The area under the spectrum corresponds to the magnitude of the transverse magnetization at time zero of the decay which is same for $T2$ and $T2^*$.

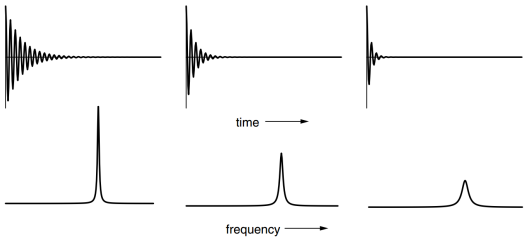


Figure 2.3 – Fourier transform property. The faster the decay in time domain, the noisier the signal in Fourier domain [106]. Figure from [124].

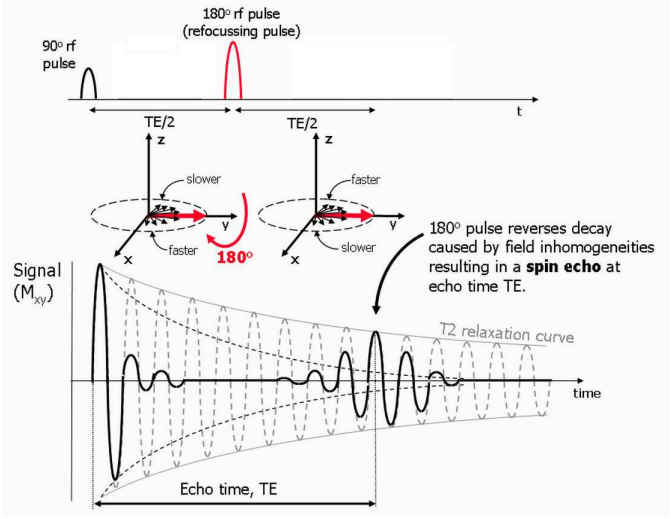


Figure 2.4 – Generating an echo by applying a $90°$ pulse followed by a $180°$ pulse. Figure from [124].

It is possible to use the disadvantage of inhomogeneous magnetic fields (i.e. that protons spin with different frequencies) in the transverse magnetic vectors to our advantage by creating an echo. This is done by applying a $90°$ pulse followed by a $180°$ pulse (see Figure 2.4). When the system is excited by a $90°$ pulse, the net magnetization deviates from the equilibrium state. Because the magnetic field is inhomogeneous, transverse magnetization vectors would spin with different speeds, where the vectors with high speed spin in front of vectors with lower speed. By applying a $180°$ pulse the fast spinning vectors go in the back and slow spinning vectors would come in front. At this point, the transverse magnetizing vectors would start to rephase and reach their maximum rephase at Echo Time (TE).

However, at TE, the signal is not as high as the initial transverse magnetization intensity. This process can be repeated many but limited number of times as the transverse magnetization intensity decreases each time. The signal envelope joining maximums of echos is known as the $T2$ decay curve (T). The MR signal sampling is after the echo time TE. The time between the 90° pulse and 180° pulse is $TE/2$. The time between two 90° excitation pulses is called repetition time (TR).

Each tissue has a specific proton density, $T1$ and $T2$ times. By varying TE and TR, it is possible to affect Nuclear Magnetic Resonance (NMR) signals (see Figure 2.2 and Table 2.1). Let A and B be two tissue types with different $T1$ and T2. If TR is too long, the net magnetization of both tissue types would have reached their equilibrium state by the time of the next excitation and thus both tissue would have the same transverse magnetization intensity after the next excitation. On the other hand, if TR is short and $T1$ relaxation time of tissue A is greater than that of tissue B, then M_Z of A would have recovered less than M_Z of B after the next excitation and therefore tissue A and B would have different contrasts. This is an effect related to T1. If TR is long, there would be no difference for A and B from $T1$ relaxation. Now lets consider the difference in $T2$ time for A and B. $T2$ is related to the transverse magnetization decay. If we apply a short TE time, that is if we apply the 180° pulse just after the 90° pulse, almost no decay has appeared and the transverse magnetization vector for both tissues would be almost the same size. In this case, no difference in $T2$ relaxation can be observed for both tissues. If TE is long enough, tissue A and tissue B would have different magnitudes of transverse magnetization vector (i.e. a phase difference) before the 180° pulse is applied. Thus the $T2$ curve would be different for these tissues. T1-weighted, T2-weighted and proton density-weighted (PD) signals are achieved by varying TR and TE signals.

The proton density-weighted signal depends primarily on the density of protons (see Table 2.1). Proton density contrast is a quantitative summary of the number of protons per unit of volume. The higher the number of protons in a unit of tissue, the greater the transverse magnetization, and the brighter the signal on the proton density contrast image.

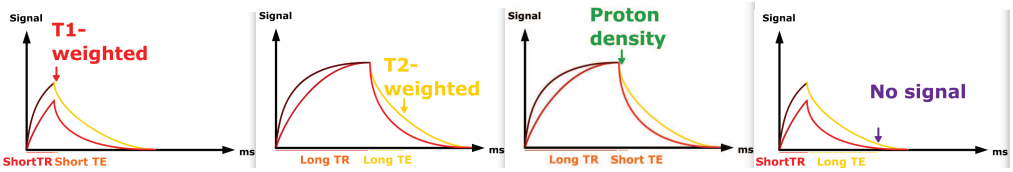
The brightness of tissue is known as signal intensity (SI) and can be computed as:

$$SI = K\rho(1 - e^{-\frac{(TR-TE)}{T1}} e^{\frac{-TE}{T2}}), \tag{2.3}$$

PD	T1-weighted	T2-weighted
Long TR (2000 ms)	Short TR (200-500 ms)	Long TR (2500 ms)
Short TE(15-30 ms)	Short TE (15-30 ms)	Long TE (100-200 ms)

Table 2.1 – Effect of TE and TR on NMR signal

where K is a proportionality constant which depends on the sensitivity of the signal detection circuitry on the scanner, ρ is the proton density contrast, TR the repetition time, TE the echo time, $T1$ and $T2$ are the relaxation times. The values of $T1$, $T2$, and ρ are specific to a tissue or pathology.



(a) T1-weighted signal (b) T2-weighted signal (c) PD signal (d) No signal

Figure 2.5 – Effect of TE and TR on NMR signal. Figure from [124].

Spatial encoding

For a 3D MR image, we need to incorporate spatial information in the NMR signals. The first step is to determine the slice plane. This is done by applying a magnetic field gradient perpendicular to the desired slice plane, which is added to B_0. In this case, protons on each slice will spin with a unique frequency and so the resonance frequency varies along the z axis.

Now by applying an RF wave with a frequency equal to the resonance frequency of the desired slice, only protons on that slice would be excited. The thickness of the slice can be controlled by the bandwidth of the RF wave. The wider the bandwidth, the higher the number of excited protons and therefore, the thicker the slice.

For the second step of spatial encoding, a phase encoding gradient is applied in the vertical direction. The phase encoding does not affect the frequency of the spins, but rather the phase. As a result, the protons on each row would have the same phase and the phase varies slightly along the columns. The greater the phase difference, the thinner

and clearer the rows would be. On that account, many different acquisitions are made with different phase encodings and then multiplied to have better effect.

The third and final step in spatial encoding is to apply a frequency encoding gradient on the horizontal axis. By doing this, protons on each column spin with the same frequency, while the resonance frequency along the horizontal axis would vary slightly. This gradient is applied at the same time when the signal is being measured.

In summary, to incorporate spatial relations in the MR signal, three different magnetic field gradients are applied in three steps. Using the Fourier transform, it is possible to analyze the MR signal. To do this, the signal is quantized (digitized) and is written into a data matrix called K-space which is in Fourier domain. The inverse Fourier transform of the K-space would comprise one slice of the MR image. By changing the magnetic field gradients we can fill in the K-space data matrix elements one by one. This process is done for all slices along the z axis.

Fluid-attenuated inversion recovery (FLAIR)
FLAIR is a sequence that produces a strong T2-weighted image but with a suppressed cerebrospinal fluid (CSF) signal. This is done by choosing a very long TE and TR signals. FLAIR helps to distinguish between CSF and lesions that appear similar in T2.

T1-weighted contrast enhanced (T1C)
To improve the contrast of MR images, MRI contrast agents are used. Gadolinium is the most common compound used for this purpose. Once injected in the blood, the molecule of the compound gather in the tumor area and reduce the $T1$, $T2$ relaxation times of the protons in their vicinity.

Chapter 3

Brain Tumor Segmentation

3.1 Anatomy of brain tumors

Tumors are mass of cells that have grown and multiplied uncontrollably. Brain tumors are serious and life threatening. One can classify brain tumors in many different ways either based on the place of origin, the infiltration degree or their location in the brain and many other ways. In this chapter, we address some of these classifications.

3.1.1 Classification by place of origin

Brain tumors either start in the brain which are referred to as primary brain tumors, or are spread into the brain via tumorous cells from a cancer else where in the body, which are referred to as secondary or metastasis.

Primary

Primary brain tumors originate in the brain and do not spread outside of the central nervous system. Depending on the type of the affected cells, primary tumors can be divided into two major subsections: glioma and non-glioma.

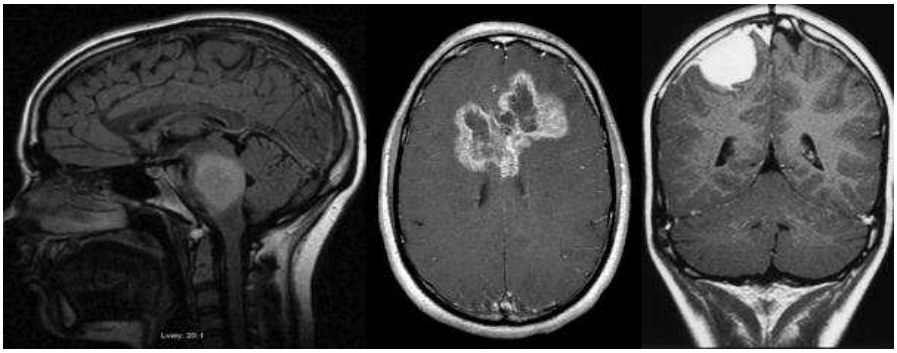


Figure 3.1 – Various types of brain tumors. From left to right images show samples of brain stem glioma, multi-form glioblastomas and meningioma.

Glioma

Glioma is a type of tumor that arises from glial cells. Glial cells are non-neuronal cells that provide supportive role within the brain by nourishing, protecting and supporting neurons. Glioma can be catogarized as:

— *Multiform glioblastoma (GBM)* –Most invasive of gliomas tumors. Grows rapidly. May be composed of several types of cells. It can evolve from other types of brain tumors. It is common in men and women between 50 to 70 years of age [28]. It can spread to other parts of the brain. Multiform glioblastoma often has a ring enhancement around the necrosis, visible in T1C. See Figure 3.1.

— *Brainstem glioma* –Located in the basin of the brain, they typically spread throughout the nervous system. They range from low to high grade and mostly appear in children 3 to 10 years of age [97]. See Figure 3.1.

— *Ependymal* –A type of glioma that the tumor originates in the cells that line the central canal of the spinal cord. They can be supratentorial (cerebral hemisphers) or infratentorial (back of the brain). Their peak occurrence is at age 5 and 35.

— *Oligodendro gliomas* –A type of glioma which most frequently appears in the frontal or temporal lobes. This type of glioma comprises 12% of the infiltrating (invasive) gliomas. While it can accour in children, it is more common in men and women of age 20 to 40 years [97]. The cause of oligodendro is genetics.

— *Astrocytomas* –A type of glioma which originates in star-shaped glial cells in the cerebrum. They can have any of the 4 tumor grades. Tumor grades are explained later in this chapter.

Non-glioma

These are tumors which arise from cells in the brain that are not glial. They include:

— *Medulloblastoma (MDL)* –It originates in the cerebellum and spreads. Most common in men (62% of the reported cases were male) and children before age of 5 [144]. They make up about 2% of all brain tumors.

— *Meningioma* –It is the most common primary brain tumor. Meningioma originates in meninge (skull area). They are benign in nature and have a slow growth rate. See Figure 3.1.

— *Pituitary adenomas* –Located in the pituitary gland, they are generally non-cancerous. 65% of these types of tumors are benign, 35% invasive and only 0.1% cancerous. They comprise about 14% of all brain tumors [85].

— *Cerebellopontine angle syndrome (CPA)*-It is located in the cerebellopontine angle which is the anatomic space between the cerebellum and the pons. They account for 5-10% of intracranial tumors and are mostly benign [146].

Secondary (metastatic)

A metastatic brain tumor is a cancer that started in another part of the body and spread to the brain. Many tumor or cancer types can spread to the brain. Most common are breast cancer, kidney cancer, lung cancer and bladder cancer.

3.1.2 Classification by terms of aggressiveness

Depending on whether or not they can spread by metastasis, tumors are classified as being either benign or malignant [97].

Benign tumors

Benign tumors are the type of tumors which lack the ability to metastasize. Therefore, benign tumors are non-cancerous. Benign tumors have slower growing rate than malignant tumors. They have distinct borders. Although most benign tumors are not life-threatening, many types of benign tumors can become malignant.

Malignant tumors

Malignant tumors are capable of spreading by metastasis. Generally the term cancerous tumors refers to malignant tumors [97]. The characteristics of malignant tumors include:

— They possess rapid growth.
— They are invasive to neighboring tissues inside the nervous system.
— They lack distinct borders.
— They are life-threatening and have deep roots in the brain.

In Figure 3.1, the image to the right shows a patient with a bening meningioma tumor while the image in the middle shows a patient with a malignant glioblastoma multiform tumor.

3.1.3 Classification by grade

Physicians usually classify brain tumors by group, which is based on the shape and the behaviour of tumor cells. Over time, a low grade tumor can evolve into a high grade tumor.

— *Grade I* –Tumor cells are benign, look like a normal brain tissue and grow slow.
— *Grade II* –Tumor cells are malignant. They are more differentiable from normal tissue than grade I tumors.
— *Grade III* –Tumor cells are malignant and look very different from normal cells and they actively grow.
— *Grade IV* –Malignant and tend to grow quickly.

3.1.4 Classification by location in brain

Brain tumors can be classified into two groups based on their position in relation to the tentorium.

— *infra tentorium* –Tumors which arise below the tentorium are called infra tentorium. These tumors exist in the cerebellum part of the brain. The cerebellum controls functions such as balance, heart function, breathing, consciousness and involuntary muscle movements.

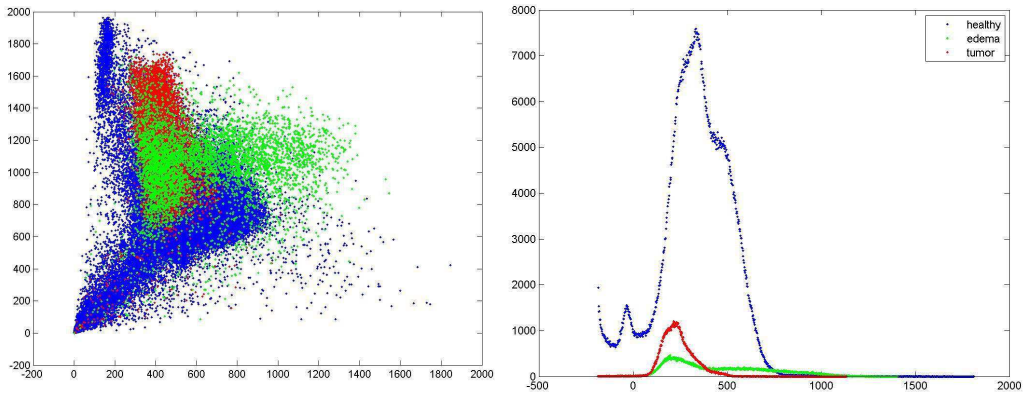


Figure 3.2 – Image intensity overlap of tumor and edema with healthy tissue. The left figure shows scatter plot of voxels on T2 (y axis) and T1C (x axis), while the figure to the right shows the histogram of healthy, edema and tumor on T1C. In both figures, the healthy class is shown in blue, edema in green and tumor in red.

— *supratentorium* –Tumors arising above the tentorium are called supratentorium. These tumors exist in the part of the brain called cerebrum. The cerebrum is responsible for functions such as movement, learning, problem solving, reasoning, personality.

An example of various type of tumors are shown in Figure 3.1. In this work we focus on segmentation of glioblastomas since they are the most challenging form of tumors in terms of segmentation. For example while meningioma are primary located in the skull area and are visible as a white blob and with will defined borders, glioblastomas can appear any where in the brain and their image intensity overlaps with that of healthy tissue (see Figure 3.2).

3.2 Brain Tumor Segmentation

A standard way to diagnose a brain tumor is by using MRI. Brain tumor segmentation is necessary for monitoring the tumor growth or shrinkage, tumor volume measurement, surgical and radiotherapy planning as well as estimating the extent of resection. For these applications, not only the tumor needs to be outlined but also the surrounding tissue. Currently, segmentation is done manually which is time consuming and tedious. The second problem with manual segmentation is that the segmentation is subject to

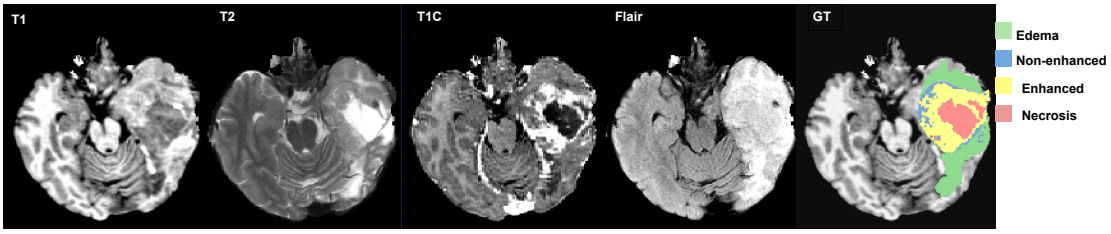


Figure 3.3 – MRI modalities and tumor sub-regions.

variation, between observers and also within the same observer. The objective of this work is to develop semi-automatic and automatic methods for brain tumor segmentation which can be used in a clinical facility.

The MRI modalities used for brain tumor segmentation are T1-weighted (also referred to as T1), T2-weighted (also referred to as T2), T1-weighted contrast-enhanced (gadolinium-DTPA) which we refer to as T1C and T2-weighted FLAIR (referred to as FLAIR). T1 is the most commonly used modality for structural analysis and distinguishing healthy tissues. In T1C the borders of the tumor are enhanced. This modality is most useful for distinguishing the active part of the tumor from the necrotic parts. In T2, the edema region appears bright. Using FLAIR we can distinguish between the edema (i.e. the swelling caused by the tumor) and the cerebral spinal fluid (CSF). This is possible because CSF appears dark in FLAIR.

Using the above mentioned inputs, the objective is to segment the tumor and its sub-regions. As shown in Figure 3.3, the sub-regions of a tumor are as follows:

— *Necrosis*–The dead part of the tumor.
— *Edema*–Swelling caused by the tumor. As the tumor grows, it can block the cerebrospinal fluid from going out of the brain. New blood vessels growing in and near the tumor can also lead to swelling.
— *Active-enhanced*–Refers to the part of the tumor which is enhanced in T1C modality.
— *Non-enhanced*–Refers to the part of the tumor which is not enhanced in T1C modality.

3.2.1 Challenges in brain tumor segmentation

This section highlights some challenges associated with a brain tumor segmentation via MRI modalities. These problems are associated to the data acquisition procedure and the nature of brain tumors:

— *Local noise*–It is a white noise introduced while measuring the signal for every pixel. It can be modeled to some extent for each pixel, by a Rician distribution independent from tissue type.
— *Intensity variation*–It is associated to intensity inhomogenity of homogeneous tissues as well as spatial intensity variations along each dimension.
— *Intensity non-standardization*–As mentioned before, the intensity of MR images depend on parameters which are in turn affected by the hardware specifications of the MRI machine.
— *Inconsistency in brain tumor shape or intensity*–Brain tumors can appear anywhere in the brain and have any shape and intensity. That makes it hard to apply a shape prior or a statistical model of the tumor with a small variance. Also, tumor (and or edema) can have intensity overlap with healthy tissue in other parts of the brain (See Figure 3.2).
— *Lack of labeled data*–Lack of labeled data makes methods based on machine learning prone to over fitting. A more thorough discussion on this matter will be presented in Chapter 4.

3.3 Previous work

Brain tumor segmentation methods can be divided in two great families : interactive (or semi-automatic) methods and automatic methods. Table 3.1 provides an overall summary of the methods described in this chapter. A more thorough overview of the related work is presented in Chapter 4.

3.3.1 Semi-automatic methods

Interactive methods or semi-automatic methods are those relying on user intervention. Many of these methods rely on active deformable models (*e.g.* snakes) or classification methods [12].

Deformable models

For these methods, the user initializes a contour around the region of interest, i.e. the tumor. The active contour then converges slowly to its closest optimal configuration. It is assumed that the global minimum energy is achieved when the contour reaches the borders of the tumor. Jiang et al. [76] uses a level set method to perform tumor segmentation. Wang et al. [172] proposed the fluid vector flow active contour model that improves its capture range in MR images. Efforts have been made to initialize the contour automatically and therefore eliminate the need for human interaction.

One problem with deformable models is that they are highly dependent on the image gradients and if the tumor region does not have well-defined borders, they are likely to fail. Also, strong gradients from surrounding objects may attract the active contour in the wrong direction. Moreover, it is not trivial to integrate multiple MRI modalities into these algorithms. Also, since snakes and level set are fundamentally 2-class segmentation methods, it is non trivial to make segmentation of $N > 2$ classes as is often required for tumor segmentation.

Classification

For these methods, the user labels some pixels as to which class (i.e. healthy, edema, active tumor, necrosis, etc) they belong to. These labeled voxels act as training data to train a classifier to predict the class for other voxels.

Vinitski et al. [170] proposed a semi-automatic method using T1,T2 and PD features, where the user labels some voxels. These labeled voxels are used as training data for the kNN to perform tissue segmentation.Vaidyanathan et al. [164] compared this method

with semi-supervised fuzzy c-means (SFCM). Kaus et al. [79] incorporated spatial information by registering the modalities to the segmented atlas and adding extra features (one for every class in the atlas) to the feature space. Zhang et al. [179] proposed a semi-automatic method where the user selects voxels only from the tumor and based on that, a one-class support vector machine is trained to segment abnormalities in the brain. Morphological operations are then used to remove false positives. Havaei et al. [63] proposed to use spatial features (i.e. positions of voxels in the xyz coordinate space) as additional features to the image intensity values. In their method they used SVM for classification and conditional random fields (CRF) was used as post processing.

3.3.2 Automatic methods

Deformable models

Automatic deformable models are those for which the algorithm tries to initialize the contour automatically and therefore eliminate the need for human interaction. Ho et al. [67] used the difference between pre and post contrast T1 as features to a Gaussian mixture model (GMM) in order to compute a probability map for the tumor, which was used to initialize the active contour. Rexilius et al. [122] initialized the segmentation by a tumor probability map based on global cross subject intensity variability, which is achieved by histogram matching. Prastawa et al. [118] used voxel registration to an atlas as a way to get a probability map for abnormalities. The snake method is initialized using this probability map. Khotanlou et al. [82] initialized their deformable model by taking advantage of the symmetrical property of the brain. They used histogram analysis of the difference image from the left and right hemispheres of the brain, to locate the tumor.

The energy minimization in deformable models is based on two main terms. The data term and the smoothness term. Much research has been done in exploring different data terms. The most common data term uses a Gaussian probability density function [175][133]. Ho et al. [67] used the class conditional density achieved by fitting a Gaussian on the histogram of the $T1 - T1C$ image. Cobzas et al. [29] showed that the probability density function (PDF) estimated discriminatively using logistic regression,

is better than the Gaussian distribution. They used intensity, texture and atlas based features to train logistic regression model in a high dimensional feature space. On a follow-up study, Popuri et al. [116] proposed a method based on Parzen windows to estimate the PDF. Automatic deformable models share most drawbacks of their interactive version.

Machine Learning methods

Automatic methods are often based on machine learning classification and clustering techniques [12]. This is mostly due to the fact that different MRI modalities can be handled in a multidimensional feature space. The choice of features can play a crucial role in the ability of the method to generalize well. Textures are sometimes extracted to provide extra dimensions to the feature space [160, 46]. After constructing a feature space by integrating different intensity and texture features, a machine learning classifier is trained so it can decide to which class a voxel belongs to. Classification in general, calls for supervised learning for which training data is needed to train a classifier based on which new observations of data can be labeled. Clustering on the other hand, works in an unsupervised way where observations are grouped based on similarity, or certain knowledge that we have from the data.

Classification methods

Jensen and Schmainda [75] combined morphological, diffusion weighted and perfusion weighted features to train a two hidden layer neural network across patient brains. Other methods have used random forests for classification. Reza [123] used T1, T2 and FLAIR along with other intensity and texture features to trained a random forest classifier. Festa [46] used series of intensity, texture and neighborhood information features. A total of 300 features were computed. A decision forest comprising of 50 trees was trained in this feature space. Tustison [159] constructed a large feature space using first order neighborhood statistical images, probability maps achieved from Gaussian Mixture Models (GMM) and template differences to train a random forest. Subbanna and Arbel [152] registered MRI modalities to a segmented brain atlas. By superimposing tumor ground truths, they created a multi-class train set. From each MR image, Gabor

filter features are extracted. They train a Bayesian classifier on the train set where each class is modeled as a GMM. MRF is applied as post processing.

Clark et al. [27] take advantage of the prior knowledge we have from properties of tumors and MRI modalities. In his knowledge-based technique, MRI modalities are processed in 2D where MRI slices in the axial view are grouped into healthy and unhealthy. This is done via fuzzy c-means algorithm and using symmetrical properties and expected intensity values of different regions in the brain. Based on the knowledge of tissue intensities in different modalities, thresholding is applied on the abnormal slides. This process is refined using density screening.

Constantin et al. [30] used spectroscopy data to coarsely detect the tumor location in the brain. Having found the rough location of the tumor, the FLAIR modality is thresholded to finely detect the tumor area. Having separated the healthy voxels from tumor affected voxles, the healthy part is further segmented into white matter, gray matter and CSF by fitting a GMM to the healthy voxels. Expectation maximization (EM) was used to find the parameters of the GMM. The PDFs were then used in an MRF model to perform segmentation.

As mentioned previously, features play an important role when it comes to methods based on classification and clustering. Schmidt et al. [137] compared the combination of different feature sets such as binary mask, left to right symmetry and probability after alignment. With the use of Gaussian filtering, multi-scale features were extracted. Linear SVM was then used for classification on high dimensional feature space, followed by median filtering for post processing. Simonetti et al. [141] explored ways to reduce the dimensionality of the feature space. They compared PCA, ICA, quantification and LC model. Nearest neighbor with respect to Mahalanobis distance was used to perform classification. Luts et al. [98] compared different feature selection methods such as Fisher discriminant, Kruskal wallis, relief-f and ARD for LS-SVM . The results were compared with LDA on a high-dimensional feature space.

The advantage of using machine learning classification methods is that it is possible to integrate many features, even if they are redundant. The drawback is that these methods can be vulnerable to overfitting, which is likely when having small datasets, especially when the distribution of the data is very variable due to the images being acquired by dif-

ferent MRI machines. Also, many machine learning methods require high-dimensional feature maps.

Clustering methods

Prastawa et al. [119] presented an unsupervised method where the query brain is registered to the segmented atlas and so the probability density function for each class is calculated. By computing $T1C - T1$, an initial estimation for the PDF of the tumor is made. This PDF is used as initialization for EM to calculate the parameters of GMM. Capelle et al. [20] used EM to compute the class conditional density function as the data term for MRF to perform multi-class label segmentation. Saha et al. [135] localized the tumor with a bounding box. First, a bounding box potential is calculated for every slice using a change detection method which uses the symmetrical property of the brain as the reference. The bounding boxes are clustered into healthy and non healthy using mean-shift. Archip et al. [5] used normalized cuts on T1 to perform clustering. To reduce the memory cost, they divide the image into supervoxels, where all voxels in each supervoxel are supposed to belong to the same class. However, after the method is launched, the user has to choose the class containing the tumor.

Deep learning methods

Recently, much research has been focused on applying deeplearning methods to brain tumor segmentation. Most of these methods are based on convolutional neural networks and provide promising results over publicly available datasets. A thorough description of these methods are provided in Chapter 4.

3.4 BRATS datasets

BRATS is a brain tumor segmentation challenge which is held annually in conjunction with the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Each challenge is associated with a dataset which are considered benchmarks for brain tumor segmentation methods. While until 2013, the top performing methods were based on decision trees, since 2014 the winning methods

Authors	Description	Training	Features	FA/SA
Havaei et al. [63]	kNN,SVM	Yes	T1C,T2,Flair,x,y,z	SA
Tustison et al. [159]	RF	Yes	T1,T1C,T2,Flair,atlas,geometry,T1C-t1 based	FA
Meier et al. [103]	RF+CRF	Yes	T1,T1C,T2,Flair	FA
Reza and Iftekharuddin [123]	RF	Yes	T1,T1C,T2,Flair,intensity difference, texture	FA
Zhao et al. [180]	Hsit. matching	Yes	T1C,T2,Flair	FA
Cordier[171]	Patch similarity	Yes	T1,T1C,T2,Flair	FA
Festa et al. [46]	RF	Yes	T1,T1C,T2,Flair,intensity difference,texture,	FA
Schmidt et al. [137]	LSVM	Yes	T1C,T2,B,atlas,A,symmetry	FA
Mangin et al. [100]	Hist.analysis+morph	No	T1	FA
Constantin et al. [30]	Thresh+Morph. op.	No	Flair,T1,T1C,T2,MRS	FA
Vinitski et al. [170]	kNN	Yes	PD,T2,T1,T1C	SA
Vaidyanathan et al. [164]	kNN,SFCM	Yes	PD,T2,T1,T1C	SA
Prastawa et al. [119]	GMM	No	(T1C-T1),T1,T2,atlas	FA
Archip et al. [5]	NCuts	No	T1	SA
Clark et al. [27]	fuzzy cmeans,Hist.thresh.	Yes	T1,T2,PD	FA
Goyal et al. [57]	Hist.analysis	No	T2,T1,T1C,symmetry	FA
Subbanna and Arbel [152]	Baissian classif.+MRF	Yes	T1,T2,Flair,T1C,GaborFeatures	FA
Khotanlou et al. [82]	Hist.analysis +deformable models	No	T1	FA
Corso et al. [33]	GMM+graph hierarchy	Yes	T1,T1C,T2,Flair	FA
Kaus et al. [80]	KNN	No	T1,T1C,T2,atlas	SA
Su et al. [151]	fuzzy clustering+svm activ.learning	Yes	T1,T1C,T2,Flair	FA
Ho et al. [67]	Hist.analysis+levelsets	No	(T1C-T1),T2	FA
Capelle et al. [20]	GMM+MRF	No	T1,T1C,T2	FA
Zhang et al. [179]	oneclass-SVM+morph.op.	Yes	T1,T1C	SA
Prastawa et al. [118]	parzen window+levelsets	No	T1,T2,atlas	FA
Jiang et al. [76]	levelsets	No	TIC	SA
Cobzas et al. [29]	Logistic.regress.+levelsets	Yes	T1,T1c,T2,atlas	SA
Luts et al. [98]	LS-KSVM,LDA	Yes	MRSI,T1,T2,PD,T1C	FA
Nie [109]	GMM+MRF	No	T1,Flair,T2	FA
Simonetti et al. [141]	PCA,ICA	Yes	H-MRSI,T1,T2,PD,T1C	FA
Saha [135]	Change detection+meanshift	No	T1C,T2,symmetry	FA
Lee et al. [92]	SVM+CRF	Yes	T1,T2,T1C	FA
Popuri et al. [116]	levelsets	Yes	T1,T2,T1C,atlas,texture,symmetry	FA

Table 3.1 – Summary of some methods on brain tumor segmentation. Columns from left to right represent name of the author, description of the method, training if applicable and the type of features used. Methods using deep learning are not discussed in this table.

in BRATS challenges have consistently used convolutional neural networks. Detailed discussions on BRATS datasets, the evaluation metrics and the top performing methods are presented in Chapter 4.

Chapter 4

Deep learning in brain pathology segmentation

Résumé

In this chapter we review in more detail, the challenges facing machine learning methods when applied to medical image segmentation and specifically to brain focal pathology segmentation. We describe the solutions that different methods in this field take to address these challenges. We provide a detailed overview of deep learning methods applied to brain tumor and lesion segmentation while addressing their pros and cons.

Commentaires

This article is to appear as a book chapter in Springer LNCS volume on Machine Learning For Health Informatics [63]. The article was mostly written and organized by the Ph.D. candidate.

Deep learning trends for focal brain pathology segmentation in MRI

Mohammad Havaei
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
seyed.mohammad.havaei@usherbrooke.ca

Nicolas Guizard
Imagia Inc., Canada
nicolas.guizard@imagia.com

Hugo Larochelle
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
hugo.larochelle@usherbrooke.ca

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Keywords: Brain tumor segmentation, Brain lesion segmentation, Deep learning, Convolutional neural network

Abstract

Segmentation of focal (localized) brain pathologies such as brain tumors and brain lesions caused by multiple sclerosis and ischemic strokes are necessary for medical diagnosis, surgical planning and disease development as well as other applications such as tractography. Over the years, attempts have been made to automate this process for both clinical and research reasons. In this regard, machine learning methods have long been a focus of attention. Over the past two years, the medical imaging field has seen a rise in the use of a particular branch of machine learning commonly known as deep learning. In the non-medical computer vision world, deep learning based methods have obtained state-of-the-art results on many datasets. Recent studies in computer aided diagnostics have shown deep learning methods (and especially convolutional neural networks - CNN) to yield promising results. In this chapter, we

provide a survey of CNN methods applied to medical imaging with a focus on brain pathology segmentation. In particular, we discuss their characteristic peculiarities and their specific configuration and adjustments that are best suited to segment medical images. We also underline the intrinsic differences deep learning methods have with other machine learning methods.

4.1 Introduction

Focal pathology detection of the central nerveous system (CNS), such as lesion, tumor and hemorrhage is primordial for accurate diagnosis, treatment and for future prognosis. The location of this focal pathology in the CNS, determines the related symptoms but clinical examination might not be sufficient to clearly identify the underlying pathology. Ultrasound, computer tomography and conventional MRI acquisition protocols are standard image modalities used clinically. The qualitative MRI modalities T1 weighted (T1), T2 weighted (T2), Proton density weighted (PDW), T2-weighted FLAIR (FLAIR) and contrast-enhanced T1 (T1C), diffusion weighted MRI and functional MRI are sensitive to the inflammatory and demyelinating changes directly associated with the underlying pathology. As such, MRI is often used to detect, monitor, identify and quantify the progression of the diseases.

For instance, in multiple sclerosis (MS), T2 lesions are mainly visible in white matter (WM), but can be found also in gray matter (GM). MS lesions are more frequently located in the peri-ventricular or sub-cortical region of the brain. They vary in size, location and volume, but are usually elongated along small vessels. These lesions are highly heterogeneous and include different underlying processes: focal breakdown of the blood-brain barrier, inflammation, destruction of the myelin sheath (demyelination), astrocytic gliosis, partial preservation of axons and remyelination. Similarly, in Alzheimer's disease (AD), white matter hyperintensity (WMH), which are presumed to be from vascular origin, are also visible in FLAIR images and are believed to be a biomarker of the disease. Similar to vascular hemorrhages, ischemic arterial or venous strokes can be detected with MRI. MRI is also used for brain tumor segmentation which is necessary for monitoring the tumor growth or shrinkage, for tumor volume measurement and also for surgical and radiotherapy planning. For glioblastoma segmentation, different MRI modalities highlight different tumor sub-regions . For example, T1 is the most commonly used modality for structural analysis and distinguishing healthy tissues. In T1C, the borders of the glioblastoma are enhanced. This modality is most useful for distinguishing the active part of the glioblastoma from the necrotic parts. In T2, the edema region appears bright and using FLAIR, we can distinguish between the edema and CSF. This is possible because CSF appears dark in FLAIR.

The sub-regions of a glioblastoma are as follows:

— *Necrosis*–The dead part of the tumor.
— *Edema*–The swelling caused by the tumor. As the tumor grows, it can block the cerebrospinal fluid from going out of the brain. New blood vessels growing in and near the tumor can also lead to swelling.
— *Active-enhanced*–Refers to the part of the tumor which is enhanced in T1C modality.
— *Non-enhanced*–Refers to the part of the tumor which is not enhanced in T1C modality.

There are many challenges associated with the segmentation of a brain pathology. The main challenges come from the data acquisition procedure (MRI in our case) as well as from the nature of the pathology. Those challenges can be summarized as follows:

— Certainly, the most glaring issue with MR images comes from the non-standard intensity range obtained from different scanners. Either because of the various magnet strengths (typically 1.5, 3 or 7 Tesla) or because of different acquisition protocols, the intensity values of a brain MRI, is often very different from one hospital to another, even for the same patient.
— There are no reliable shape or intensity priors for brain tumors/lesions. Brain pathology can appear anywhere in the brain, they can have any shape (often with fuzzy borders) and come with a wide range of intensities. Furthermore, the intensity range of such pathology may overlap with that of healthy tissue making computer aided diagnosis (CAD) complicated.
— MR images come with a non negligible amount of white Rician noise introduced during the acquisition procedure.
— Homogeneous tissues (typically the gray and the white matter) often suffer from spatial intensity variations along each dimension. This is caused by a so-called bias field effect. The MRI bias is a smooth low-frequency signal that affects the image intensities. This problem calls for a bias field correction pre-processing step which typically increases intensity values at the periphery of the brain.
— MR images may have non-isotopic resolution, leading to low resolution images, typically along the coronal and the saggital views.
— The presence of a large tumor or lesion in the brain, may warp the overall struc-

ture of the brain, thus making some procedures impossible to perform. For example, large tumors may affect the overall symmetry of the brain, making left-right symmetry features impossible to compute. Also, brains with large tumors can hardly be registered onto a healthy brain template.

Methods relying on machine learning also have their own challenges when processing brain images. To count a few:

— Supervised methods require a lot of labeled data in order to generalize well to unseen examples. As opposed to non-medical computer vision applications, acquiring medical data is time consuming, often expensive and requires the non-trivial approval of an ethical committee as well as the collaboration of non-research affiliated staff. Furthermore, the accurate ground truth labeling of 3D MR images is time consuming and expensive, as it has to be done by highly trained personnel (typically neurologists). As such, publicly-available medical datasets are rare and often made of a limited number of subjects. One consequence of not having enough labeled data is that the models trained on such datasets are prone to overfitting and perform poorly on new subjects.

— In supervised learning, we typically estimate by maximum likelihood and thus assume that the examples are identically distributed. Unfortunately, the intensity variation from one MRI machine to another, often violates that assumption. Large variations in the data distribution can be leveraged by having a sufficiently large training dataset, which is almost never the case with medical images.

— Classic machine learning methods rely on computing high dimensional feature vectors, which can make them computationally inefficient both memory-wise and processing-wise.

— Generally in brain tumor/lesion segmentation, ground truth is heavily imbalanced since regions of interest are very small compared to the whole brain. This is very unfortunate for many machine learning methods such as neural networks which work best when classes have similar size.

— Because of the variability of the data, there is no standard pre-processing procedure.

Most brain lesion segmentation methods use hand-designed features [44, 104]. These methods implement a classical machine learning pipeline according to which features

are first extracted and then given to a classifier whose training procedure does not affect the nature of those features.

An alternative would be to *learn* such a hierarchy of increasingly complicated features (i.e. low, mid and high level features). Deep neural networks (DNNs) have been shown to be successful in learning task-specific feature hierarchies [15]. Importantly, a key advantage of DNNs is that they allow to learn MRI brain-pathology-specific features that combine information from across different MRI modalities. Also, convolutions are very efficient and can make predictions very fast. We investigate several choices for training Convolutional Neural Networks (CNNs) for this problem and report on their advantages, disadvantages and performance. Although CNNs first appeared over two decades ago [90], they have recently become a mainstay for the computer vision community due to their record-shattering performance in the ImageNet Large-Scale Visual Recognition Challenge [86]. While CNNs have also been successfully applied to segmentation problems [3, 96, 61], most of the previous work have focused on non-medical tasks and many involve architectures that are not well suited to medical imagery or brain tumor segmentation in particular.

Over the past two years, we have seen an increasing use of deep learning in health care and more specifically in medical imaging segmentation. This increase can be seen in recent Brain Tumor Segmentation challenges (BRATS) which is held in conjunction with Medical Image Computing and Computer Assisted Intervention (MICCAI). While in 2012 and 2013 none of the competing methods used DNNs, in 2014, 2 of the 15 methods and in 2015, 7 of the 13 methods taking part in the challenge were using DNNs. In this work, we explore a number of approaches based on deep neural network architectures applied to brain pathology segmentation.

4.2 Glossary

Cerebral spinal fluid (CSF) : a clear, colorless liquid located in the middle of the brain.

Central nervous system (CNS) : part of the nervous system consisting of the brain and the spinal cord.

Diffusion weighted image (DWI) : MR imaging technique, measuring the diffusion of water molecules within tissue voxels. DWI is often used to visualize hyperintensities.

Deep Neural Network (DNN) : an artificial intelligence system inspired from human nervous system, where through a hierarchy of layers, the model learns a hierarchy of low to high end features.

Convolutional Neural Network (CNN) : a type of DNN adopted for imagery input. The number of parameters in a CNN is significantly less than that of a DNN due to a parameter sharing architecture made feasible by convolutional operations.

FLAIR image : an MRI pulse sequence that suppresses fluid (mainly cerebrospinal fluid (CSF)) while enhancing edema.

Gray matter (GM) : a large region located on the surface of the brain consisting mainly of nerve cell bodies and branching dendrites.

High-grade glioma : malignant brain tumors of types 3 and 4.

Low-grade glioma : slow growing brain tumors of types 1 and 2.

Multiple sclerosis (MS) : a disease of the central nervous system attacking the myelin, the insulating sheath surrounding the nerves.

Overfitting : in machine learning the *overfitting* phenomenon occurs when the model is too complex relative to the number of observations. Overfitting reduces the ability of the model to generalize to unseen examples.

Proton density weighted (PDW) image : an MR image sequence used to measure the density of protons; an intermediate sequence sharing some features of both T1 and T2. In current practices, PDW is mostly replaced by FLAIR.

T1-weighted image : one of the basic MRI pulse sequences showing the difference in the T1 relaxation times of tissues [47].

T1 Contrast-enhanced image : a T1 sequence, acquired after a gadolinium injection. Gadolinium changes the signal intensities by shortening the T1 time in its surroundings. Blood vessels and pathologies with high vascularity appear bright in T1 weighted post gadolinium images.

T2-weighted image : one of the basic MRI pulse sequences. The sequence highlights differences in the T2 relaxation time of various tissue[48].

White matter hyperintensity : changes in the cerebral white matter in aged individuals or patients suffering from a brain pathology [120].

4.3 Datasets

In this section, we describe some of the most widely-used public datasets for brain tumor/lesion segmentation.

BRATS benchmark The Multimodal BRain Tumor image Segmentation (BRATS), is a challenge held annually in conjunction with the MICCAI conference since 2012. The BRATS 2012 training data consist of 10 low- and 20 high-grade glioma MR images whose voxels have been manually segmented with three labels (*healthy*, *edema* and *core*). The challenge data consist of 11 high- and 5 low-grade glioma subjects and no ground truth is provided for this dataset. Having only two basic tumor classes is insufficient due to the fact that the *core* label contains structures which vary in different

modalities. For this reason, the BRATS 2013 dataset contains the same training data but was manually labeled into 5 classes; *healthy*, *necrosis*, *edema non-enhanced* and *enhanced tumor*. There are also two test sets available for BRATS 2013 which do not come with ground truth; the *leaderboard* dataset which contains the BRATS 2012 challenge dataset with additional 10 high-grade glioma patients and the BRATS 2013 *challenge* dataset which contains 10 high-grade glioma patients. The above mentioned datasets are available for download through the challenge website [171].

For BRATS 2015, the size of the dataset was increased extensively [1]. BRATS 2015 contains 220 subjects with high-grade and 54 subjects with low grade gliomas for training and 53 subjects with mixed high and low grade gliomas for testing. Similar to BRATS 2013, each brain from the training data, comes with a 5 class segmentation ground truth. BRATS 2015 also contains the training data of BRATS 2013. The ground truth for the rest of the training subjects are generated automatically with the integration of the top performing methods in BRATS 2013 and BRATS 2012. Although some of the automatically generated ground truths have been refined manually by a user, some challenge participants have decided to remove subjects with heavily corrupted ground truths from their training data [64, 161, 83]. This dataset can be downloaded through the challenge website [171].

All BRATS datasets, share four MRI modalities namely; T1, T1C, T2, FLAIR. Image modalities for each subject are co-registered to T1C. Also, all images are skull stripped.

Quantitative evaluation of the model's performance on the test set is achieved by uploading the segmentation results to the online BRATS evaluation system [171]. The online system provides the quantitative results as follows: The tumor structures are grouped in 3 different tumor regions. This is mainly due to practical clinical applications. As described by Menze et al. (2014) [104], tumor regions are defined as:

1. The *complete* tumor region (including all four tumor structures).
2. The *core* tumor region (including all tumor structures exept “edema").
3. The *enhancing* tumor region (including the “enhanced tumor" structure).

1. Note that the BRATS organizers released a dataset in 2014 which was later removed from the web. This version of the dataset is no longer available.

For each tumor region, Dice, Sensitivity, Specificity, Kappa as well as the Hausdorff distance are reported. The online evaluation system provides a ranking for every method submitted for evaluation. This includes methods from the 2013 BRATS challenge published in [104] as well as anonymized unpublished methods for which no reference is available.

ISLES benchmark Ischemic Stroke Lesion Segmentation (ISLES) challenge started in 2015 and is held in conjunction with the Brain Lesion workshop as part of MICCAI. ISLES has two categories with individual datasets; sub-acute ischemic stroke lesion segmentation (SISS) and acute stroke outcome/penumbra estimation (SPES) datasets [40]. Similar to BRATS, an online evaluation system is available to evaluate the segmentation outputs of the test subjects.

SISS contains 28 subjects with four modalities, namely: FLAIR, DWI, T2 TSE (Turbo Spin Echo), and T1 TFE (Turbo Field Echo). The challenge dataset consists of 36 subjects. The evaluation measures used for the ranking are the Dice coefficients, the average symmetric surface distance, and the Hausdorff distance.

SPES dataset contains 30 subjects with 7 modalities namely: CBF (Cerebral blood flow), CBV (cerebral blood volume), DWI, T1C, T2, Tmax and TTP (time to peak). The challenge dataset contains 20 subjects. Both datasets provide pixel level ground truth of the abnormal areas (2 class segmentation). The metrics used to gauge performances are the Dice score, the Hausdorff distance, the recall and precision as well as the average symmetric surface distance (ASSD).

MSGC benchmark The MSGC dataset which was introduced at MICCAI 2008 [150], provides 20 training MR cases with manual ground truth MS lesion segmentation and 23 testing cases from the Boston Children's Hospital (CHB) and the University of North Carolina (UNC). For each subject, T1, T2 and FLAIR are provided which are co-registered. While lesions masks for the 23 testing cases are not available for download, an automated system is available to evaluate the output of a given segmentation

algorithm. The MSGC benchmark provides different metric results normalized between 0 and 100, where 100 is a perfect score and 90 is the typical score of an independent rater [150]. The different metrics (volume difference "VolD", surface distance "SurfD", true positive rate "TPR" and false positive rate "FPR") are measured by comparing the model output segmentation to the manual segmentation of two experts at CHB and UNC.

4.4 State-of-the-art

In this section, we present a brief overview of some methods used to segment brain lesions and brain tumors from MR images.

4.4.1 Pre deep learning era

These methods can be grouped in two major categories: *semi-automatic* and *automatic* methods. Semi-automatic (or interactive) methods are those relying on user intervention. Many of these methods rely on active deformable models (*e.g.* snakes) where the user initializes the tumor contour [76, 172]. Other semi-automatic methods use classification which the input to the model is given through regions of interest drawn from inside and outside of the tumor [79, 179, 63, 65, 12]. Semi-automatic methods are appealing in medical imaging applications since the datasets are generally very small [69, 51]. Automatic methods on the other hand are those for which no user interaction is made. These methods can be divided into two groups; The first group of methods are based on *anomaly* detection, where the model estimates intensity similarities between the query subject and an atlas. By doing so, brain regions which deviate from healthy tissue are detected. These techniques have shown good results in structural segmentation when using non-linear registration [58, 122, 118, 82].

The second group of methods are *machine learning methods*, where a discriminative model is trained using *pre-defined* features of the input modalities. After integrating different intensity and texture features, a classifier is trained to decide to which class each voxel belongs to. Random forests have been particularly popular. Reza et al. [123] used a mixture of intensity and texture features to train a random forest for voxelwise

classification. One problem with this approach is that the model should be trained in a high-dimensional feature space. For example, Festa et al. [46] used a feature space of 300 dimensions and the trained random forest comprised of 50 trees. To train more descriptive classifiers, some methods have taken the approach of adding classes to the ground truth [11, 181]. Tustison et al. [160] does this by using Gaussian Mixture Models (GMMs) to get voxelwise tissue probabilities for WM, GM, CSF, edema, non-enhancing tumor, enhancing tumor, necrosis. The GMM is initialized with prior cluster centers learnt from the training data. The voxelwise probabilities are used as input features to a random forest. The intuition behind increasing the number of classes is that the distribution of the healthy class is likely to have different modes for WM, GM and the CSF and so the classifier would be more confidant if it tries to classify them as separate classes. Markov random fields (MRF) as well as conditional random fields (CRF) are sometime used to regularize the predictions [103, 66, 94, 160]. Usually, the pairwise weights in these models are either fixed [66] or determined by the input data. They work best in the case of weak classifiers such as k-nearest neighbor (kNN) or decision trees and become less beneficial when using stronger classifiers such as convolutional neural networks [132].

Deformable models can also be used as post-processing, where an automatic method is used to initialize the counter as opposed to user interaction in semi-automatic methods [67, 122, 118, 82].

4.4.2 Deep learning based methods

As mentioned before, classical machine learning methods in both automatic and semi-automatic approaches use pre-defined (or hand-crafted) features which might or might not be useful in the training objective. Oppose to that, deep learning methods *learn* features specific to the task at hand. Moreover, these features are learnt in a hierarchy of increasing feature complexity, which results in more robust features.

Recently, deep neural networks have proven to be very promising for medical image segmentation. In the past two years, we have seen an increase in use of neural networks applied to brain tumor and lesion segmentations. Notable mentions are the MICCAI

brain tumor segmentation challenges (BRATS) in 2014 and 2015 and the ISLES challenge in 2015 where the top performing methods were taking use of convolutional neural networks [44, 45].

In spite of the fact that CNNs were originally developed for image classification, it is possible to use them in a segmentation framework. A simple approach is to train the model in a *patch wise* fashion as in [25], where for every training (or testing) pixel i, a patch $\mathbf{x}_i$ of size $n \times n$ around i is extracted, and the goal is to identify class label of the center pixel.

Although MRI segmentation is a 3D problem, most methods take a 2D approach by processing the MRI slice by slice. For these methods, training is mostly done patch wise on the axial slices. Zikic et al. [183] use a 3 layer model with 2 convolutional layers and one dense layer. The input size of the model is 19×19, however, since the inputs have been downsampled by a factor of 2, the effective receptive field size is 38×38. *Max pooling* with a stride of 3 is used at the first convolutional layer. During test time, downsampled patches of 19×19 are presented to the model in sliding window fashion to cover the entire MRI volume. The resulting segmentation map is upsampled by a factor of two in order to have the same size as the input.

The TwoPathCNN by Havaei et al. [66] consists of two pathways: a *local pathway* which concentrates on the pixel neighborhood information and a *global pathway* which captures more the global context of the slice. Their local path consists on 2 convolutional layers with kernel sizes of 7×7 and 5×5 respectively, while the global path consists of one convolutional layer with 11×11 kernel size. In their architecture, they use *Maxout* [53] as activation function for intermediate layers. Training patch size is set to 33×33, however during test time, the model is able to process a complete slice making the overall prediction time drop to a couple of seconds. This is achieved by implementing a convolutional equivalent of the dense layers. To preserve pixel density in the segmentation map, they use a stride of 1 in all max pooling and convolutional layers. [2] This architecture is shown in Figure. 4.1.

2. Using stride of n means that every n pixels will be mapped to 1 pixel in the label map (assuming the model has one layer). This causes the model to loose pixel level accuracy if full image prediction is to be used at test time. One way to deal with this issue is presented by Pinheiro et al. [113]. Alternatively, we can use a stride of 1 every where in the model.

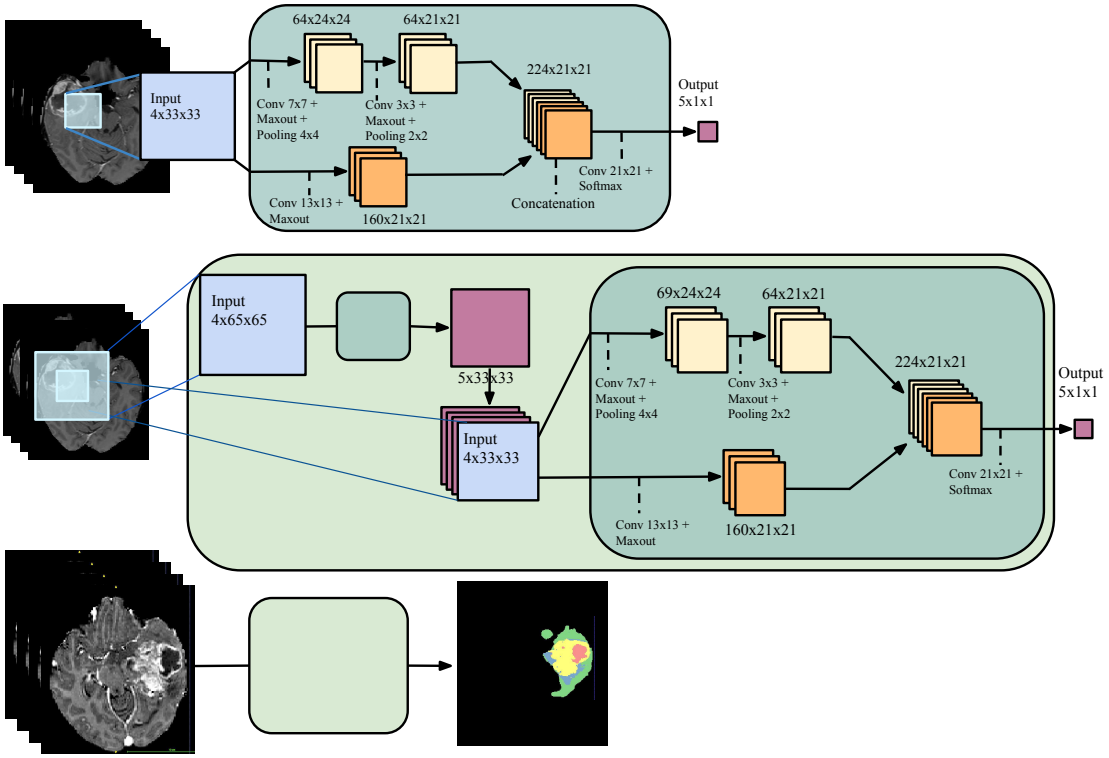


Figure 4.1 – The proposed architecture by Havaei et al. [66]. First row: TWOPATHCNN. The input patch goes through two convolutional networks each comprising of a local and a global path. The feature maps in the local and global paths are shown in yellow and orange respectively. Second row: INPUTCASCADECNN. The class probabilities generated by TWOPATHCNN are concatenated to the input of a second CNN model. Third row: Full image prediction using INPUTCASCADECNN.

Havaei et al. [66] also introduce a cascaded method where the class probabilities from a base model are concatenated with the input image modalities to train a secondary model similar in architecture to that of the base model. In their experiments, this approach refined the probability maps produced by the base model and brought them among the top 4 teams in BRATS 2015 [64].

Pereira et al. [112] also use a CNN with patch wise training and small kernel sizes (i.e. 3×3) as suggested by [142]. This allowed them to have a deeper architecture while maintaining the same receptive field as shallow networks with larger kernels. They train separate models for HG and LG tumors. For the HG model, their architecture consists of 8 convolutional layers and 3 dense layers, while the LG model is a bit shallower,

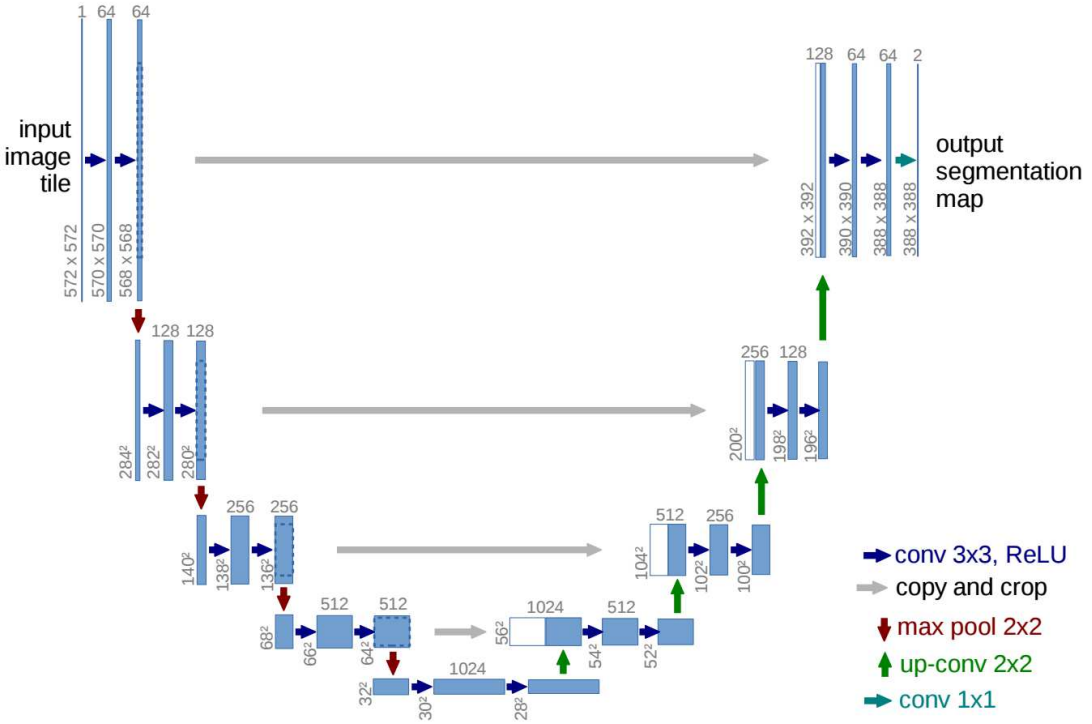


Figure 4.2 – U-Net: The proposed architecture by Ronneberger et al. [127].

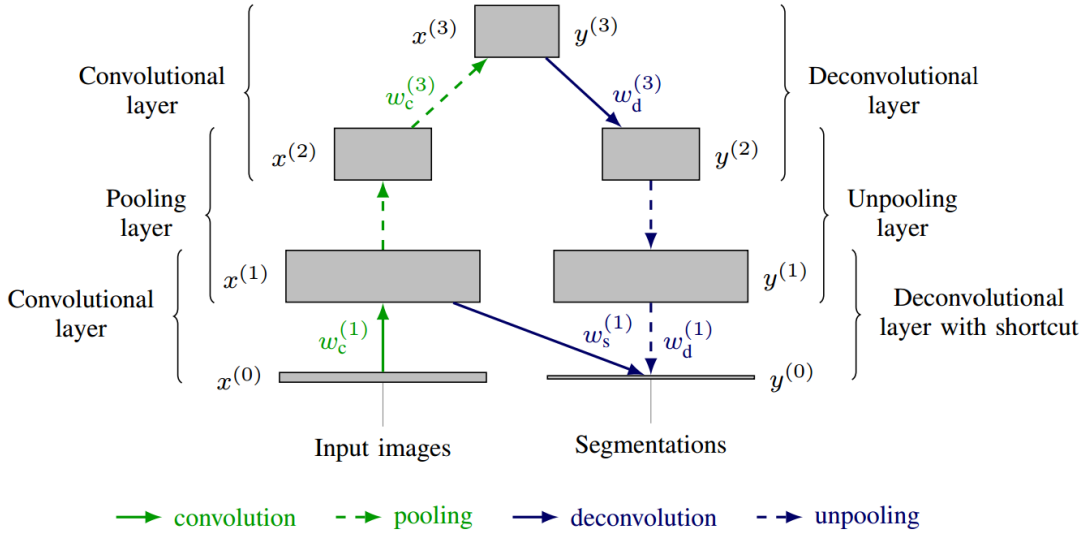


Figure 4.3 – CEN-s: The proposed architecture by Brosch et al. [17].

containing 4 convolutional layers and 3 dense layers. They use max pooling with a stride of 2 and dropout is used only on the dense layers. Leaky rectified linear units (LRLU) [99] are used for the activation function of all intermediate layers. This method achieved good results in the BRATS 2015 challenge, ranking them among the top 4 winners. The authors also find *data augmentation* by rotation to be useful. That said, the method comes with a major inconvenience, which is for the user to manually decide the type of the tumor (LG or HG) to process.

Dvorak et al. [39] applied the idea of *local structure prediction* [37] for brain tumor segmentation, where a dictionary of label patches is constructed by clustering the label patches into n groups. The model is trained to assign an input patch to one of the n groups. The goal is to force the model to take into account labels of the neighboring pixels in addition to the center pixel.

The methods discussed above treat every MRI modality as a channel in the CNN. Rao et al. [121] proposed instead to treat these modalities as inputs to separate convolutional streams. In this way, they train 4 separate CNN models each on a different modality. After training, these models are used as feature extractors where features from the last pooling layer of all 4 models are concatenated to train a random forest classifier. The CNNs share the same architecture of 2 convolutional layers of kernel size 5×5 followed by 2 dense layers. Every CNN takes as input 3 patches of size 32×32, extracted from 3 dimensions (i.e. axial, sagittal, coronal) around the center pixel.

Segmentation problems in MRI are often 3D problems. However, employing CNNs on 3D data remains an open problem. This is due to the fact that MRI volumes are often anisotropic (especially for the FLAIR modality) and the volume resolution is not consistent across subjects. A solution is to pre-process the subjects to be isotropic [104, 58]. However, these methods only interpolate the data and the result ends up being severely blurry when the data is highly anisotropic. One way to incorporate information from 3D surroundings is to train on orthogonal patches extracted from axial, sagittal and coronal views. The objective would then be to predict the class label for the intersecting pixel. This is referred to as 2.5D in the literature [121, 140]. Havaei et al. [66] experimented with training on 2.5D patches. However, they argued that since BRATS 2013 train and test data have different voxel resolutions, the model did not generalize better than when

only training on patches from the axial view. Vaidya et al. [163] and Urban et al. [161] used 3D convolutions for brain lesion and tumor segmentation. Using 3D convolution implies that the input to the model has an additional depth dimension. Although this has the advantage of using the 3D context in the MRI, if the gap between slices across subjects varies a lot, the learnt features would not be robust. In a similar line of thought, Klein et al. [84] also used 3D kernels for their convolutional layers, but with a different architecture. Their architecture consists of 4 convolutional layers with large kernel sizes on the first few layers (i.e. $12 \times 12 \times 12$, $7 \times 7 \times 7$, $5 \times 5 \times 5$, $3 \times 3 \times 3$) with input patch size of $41 \times 41 \times 41$. The convolutional layers are followed by 2 dense layers.

Kamnitsas et al. [78] used a combination of the methods above [161, 66, 112], applied to lesion segmentation. In their 11 layer fully convolutional network which consisted of 2 pathways similar to [66], they used 3D convolutions with small kernel sizes of $3 \times 3 \times 3$. Using this model, they ranked among the winners of the ISLES 2015 challenge.

Stollenga et al. [149] used a long short-term memories (LSTM) network applied to 2.5D patches for brain segmentation.

As opposed to methods which use deep learning in a CNN framework, Vaidhya et al. [162] used a multi-layer perceptron consisting of 4 dense layers. All feature layers (i.e. the first 3) were pre-trained using denoising auto-encoder as in [169]. The input consists of 3D patches of size $9 \times 9 \times 9$. Training is performed on a resampled version of the BRATS dataset, which balances the number of class patches. However, similar to [66], fine-tuning is done on the original dataset with imbalanced classes to reflect the real distribution of label classes.

Inspired by [102], Brosch et al. [16] presented the convolutional encoder networks (CEN) for MS lesion segmentation. The model consists of 2 parts; the encoder part which decreases the resolution of the feature maps and the up sampling part (also known as the decoder part) which increases the resolution of the feature maps and performs pixel level classification [3]. The encoder consists of 2, 3D convolutional layers in *valid mode* [4] with kernel size $9 \times 9 \times 9$ in both layers, followed by an ReLU activation function. The up sampling part of the model consists of convolutions in *full mode* [5] which results

3. In the literature this way of up sampling is some times wrongly referred to as *deconvolution*.
4. Valid mode is when kernel and input have complete overlap.
5. Full mode is when minimum overlap is a sufficient condition for applying convolution.

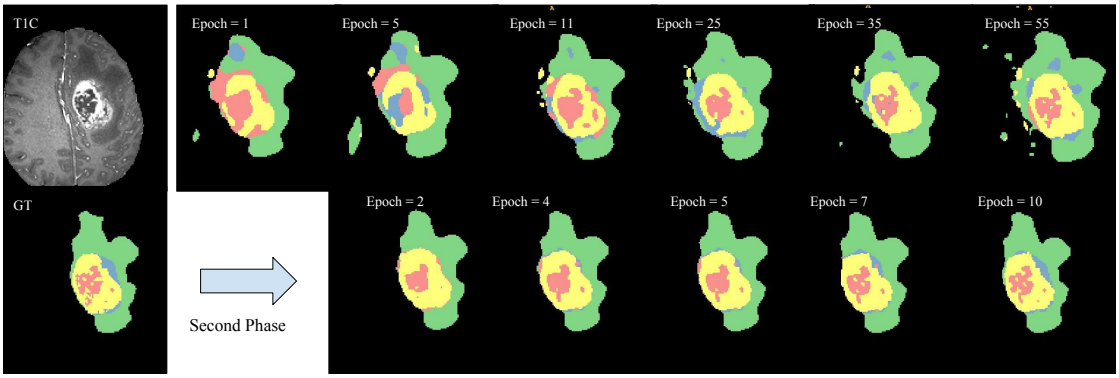


Figure 4.4 – Effect of second phase training proposed by [66]. The figure shows how the second phase regularizes the predictions and removes false positives.

in up sampling the model. Balancing label classes is done by introducing weights per class in the loss function. They improved on this method in [17] by introducing CEN-s, where they combine feature maps from the first hidden layer to the last hidden layer. As shown in Figure. 4.3 and Figure 4.2, this model is very similar to the U-Net by Ronneberger et al. [127] with a difference in the way the up sampling step is applied. While U-Net uses interpolation for up sampling, CEN-s uses convolutions and the transformation weights are learnt during training. Also U-Net is deeper with 11 layers, while CEN-s contains only 4 layers.

Combining feature maps from shallow layers to higher layers (also referred to as *skip* or *shortcut* connections) are popular in semantic segmentation [96, 62].

4.5 Open Problems

4.5.1 Preparing the dataset

Preparing the dataset in a proper way can play a key role in learning. In this chapter, we discuss important aspects of dataset preparation for medical imaging.

Pre-processing

As mentioned before, the grayscale distribution of MR images depends on the acquisition protocol and the hardware. This makes learning difficult since we expect to have the same data distribution from one subject to another. Therefore, pre-processing to bring all subjects to similar distributions is an important step. Also, it is desirable that all input modalities have the same intensity range, so one modality does not have prior advantage over others in deciding the output of the model. Among the many pre-processing approaches reported in the literature, the following are the most popular:

— Applying the N4/N3 bias field correction [160, 66, 56, 183, 87, 58, 39]. Kleesiek et al. [83] and Urban et al. [161] did not apply bias field correction, instead, they performed intensity normalization with mean CSF value, which they claim to be more robust and effective.

— Truncating the 1% or 0.1% quantiles of the histogram to remove outliers from all modalities [160, 66, 162].

— Histogram normalization, which is mostly done by matching the histogram of every modality to their corresponding template histogram. [8, 112, 162, 58].

— Normalizing modalities [66, 39] or the selected training patches [112] to have zero mean and unit variance.

Shuffling

Introducing the data to the model in a sequential order results in biasing the gradients and can lead to poor convergence. By sequential order, we mean training first on data (i.e. patches or slices) extracted from a subject, then training on data extracted from another subject, and so on until the end of the training set. Depending on the dataset, MRI subjects can be very different in terms of noise and even intensity distribution. Therefore, it is important to shuffle the entire dataset so the model does not overfit to the current training subject and forget its previous findings. It is desirable that the distribution from which we introduce training examples to the model does not change significantly. An advantage of patch wise training over full image training is that patch wise training allows us to fully shuffle the dataset. This means, in patch wise training, every mini batch contains patches from different slices of different subjects while in full

image training, there is no shuffling at pixel level.

Balancing the dataset

A dataset is imbalanced when class labels are not approximately equally represented. Unfortunately, brain imaging data are rarely balanced due to the small size of the lesion compared to the rest of the brain. For example, the volume of a stroke is rarely more than 1% of the entire brain and a tumor (even large glioblastomas) never occupy more than 4% of the brain. Training a deep network with imbalanced data often leads to very low, true positive rate since the system gets biased towards the one class that is over represented.

Ideally, we want to learn features invariant to the class distribution. This can be done through balancing the classes in the dataset. One approach is to take samples from the training set so we get an equal number of samples for every class. Another approach is to weight the loss for the training examples from different classes based on the frequency of appearance of every class in the training data [127] [16]. Sampling from the training set can be done randomly [132, 131, 130], or follow an importance sampling criterion to help the model learn features we care about (for example border between classes). In Havaei et al.'s [66] patch wise training method, the importance sampling is done by computing the class entropy for every pixel in the ground truth and giving training priority to patches with higher entropy. In other words, patches with higher entropy, contain more classes which makes them good candidates to learn the border regions from.

Training on a balanced dataset makes the model believe all classes are equiprobable and thus may cause some false positives. In order to compensate for this, one can account for the imbalanced nature of the data with a second training phase, during which, only the classification layer is trained and other feature layers are fixed. This allows to regularize the model and remove some false positives. The effect of the second phase training is presented in Fig 4.4. Ronneberger et al. [127] took a different approach which is best suited for full image training. In their approach, they compute the distance of every pixel to class borders and, based on that, a weight is assigned to every pixel. A weight map is created for every training image and is used in the loss function to weight every

sample differently.

Pereira et al. [112] balance classes mainly by data augmentation. In their case, data augmentation can be either a transformation applied on a patch or simply using patches from similar datasets. For example, using patches from brains with high-grade glioma when training a low-grade glioma model.

4.5.2 Global information

Adding context information has always been a subject of interest in medical image analysis [2, 32, 34]. Since anatomical regions in closeup view can appear similar and borders may be diffused in some parts due to lack of contrast or other artifacts, additional context is needed to localize a region of interest.

In a CNN, it is possible to encode more contextual information by increasing the portion of the input image that each neuron sees (directly or indirectly). Although it is possible to increase the receptive filed of a neuron on the input image through series of convolutional and pooling layers of stride 1, using strides greater than 1 is computationally more efficient and results in more robust features. By doing so, the model looses precision of spatial information which is needed for segmentation purposes. To take advantage of both worlds (i.e. having spatial precision while learning robust features through pooling layers) encoder-decoder type architectures can be used. Ronneberger et al. [127] and Brosch et al. [17] learn a global understanding of the input by down sampling the image (through series of convolutional and pooling layers) to smaller size feature maps. These feature maps are later up sampled in the decoder section of the model and combined with feature maps of lower layers that preserve the spatial information (see Figure 4.2 and Figure 4.3).

Havaei et al. [66] take a different approach where feature maps from 2 convolutional streams (using the same input) are concatenated before going through the classification layer. This two pathway approach, allows the model to learn simultaneously local and global contextual features (see Figure 4.1).

4.5.3 Structured prediction

Although CNNs provide powerful tools for segmentation, they do not model spatial dependencies in the segmentation space directly. To address this issue, many methods have been proposed to take the information of the neighboring pixels in the label image into account. These methods can be divided into two main categories. The first category are methods which consider the information of the neighboring labels in an *implicit* way, while providing no specific pairwise term in the loss function. An example of such an approach is provided by Havaei et al. [66] which refine predictions made by a first CNN model by providing the posterior probabilities over classes as extra inputs to a second CNN model. Roth et al. [132] also use a cascaded architecture to concatenate the probabilities of their first convolutional model with features extracted from multiple scales in a *zoom out* fashion [107]. The second category of methods are ones that *explicitly* define a pairwise term in the loss function which is usually referred to as Conditional Random Field (CRF) in the literature. Although it is possible to train the CNN and CRF end to end, usually for simplicity, the CRF is trained or applied as a post processing secondary model to smooth the predicted labels. The weights for the pairwise terms in the CRF can be fixed [63], determined by the input image [63] or learned from the training data [132]. In their work, Roth et al. [132] trained an additional CNN model between pairs of neighboring pixels.

Post-processing methods based on *connected components* have also proved to be effective to remove small false positive blobs [162, 66, 112]. In [132], the authors also try 3D isotropic Gaussian smoothing to propagate 2D predictions to 3D and according to them, Gaussian smoothing was more beneficial than using a CRF.

4.5.4 Training on small or incomplete datasets

Deep neural networks generalize better on new data if a large training set is available. This is due to the large number of parameters present in these models. However, constructing a medical imaging dataset is an expensive and tedious task which causes datasets to be small and models trained on these datasets prone to overfitting. Even the largest datasets in this field does not exceed a few hundred subjects. This is much

smaller than datasets like ImageNet, which contains millions of images.

Another problem arises from incomplete datasets. Medical imaging datasets are often multi-modal with images from MRI acquisitions (T1, T2, PD, DWI, etc.) [104, 95]. However, not all modalities are always available for every subject. How to effectively use the incomplete data rather than simply discarding them is an open question. Another scenario is how to generalize on subjects with missing modalities. In this section we review several effective approaches to train on small and/or incomplete datasets

Data augmentation

Increasing the size of the dataset by data augmentation is commonly employed in machine learning to enrich a dataset and reduce overfitting [86]. Flipping the image, applying small rotations and warping the image are common practices for this purpose [86, 26, 127]. Roth et al. [132] and Ronneberger et al. [127] use non-rigid deformation transformations to increase the size of their datasets and report it to be a key element in achieving good results. The type of data augmentation technique depends on the anatomy of the data and the model being used. For example, Pereira et al. [112] only tested with rotation for data augmentation because the label of the patch is determined by the center pixel and so warping or applying translations might change the position of the center pixel. They used angles multiple of 90° and managed to increase the size of the dataset 4 times. They found data augmentation to be very effective in their experiments.

Transfer learning

Deep learning has made significant breakthroughs in computer vision tasks due to training on very large datasets such as ImageNet. ImageNet contains more than 1.2 million training examples on over 1000 classes. To improve generalization on smaller datasets, it is common to first train a *base* model on a large dataset such as ImageNet and then fine-tune the learnt features on a second *target* model which is often much smaller in size. Yosinski et al. [176] showed that the transferability of the features depends on how general those features are. The transferability gap increases as the distance between

the tasks increase and also as the features become less general in higher levels. However, initializing weights from a pre-trained model (preferably on a large dataset), is still better than initializing weights randomly.

Transfer learning can take different forms. One way is to generate features from the base model and then use those features to train a classifier such as SVM or logistic regression [9, 166, 6]. Bar et al. [9] used an ImageNet pre-trained base model to extract features. These features are concatenated with other hand-crafted features before being introduced to an SVM classifier. Van et al. [166] used *overfeat* pre-trained weights to generate features for lung tumor detection. To address the overfeats 3 input channels, 3 2d patches are extracted from axial, saggital and coronal views. SVM is used as classifier.

Although this way of transfer learning has proved to be somewhat successful, the degree of its usefulness depends on how similar the source and target datasets are. If not very similar, a better alternative is to fine-tune the features on the target dataset [22, 21, 49, 101]. Gao et al. [49] used this fine-tuning scheme to detect lung disease in CT images. To account for the 3 color channels of the base model which has been pre-trained on ImageNet, 3 attenuation scales with respect to lung abnormality patterns are captured by rescaling the original 1-channel CT image. Carneiro et al. [21] uses this method to reach state-of-the-art results on the InBreast dataset. Shin et al. [140] reported experimental results in 3 transfer learning scenarios for Lymph node detection. 1) No transfer learning 2) transfering the weights from a base model and only training the classification layer (i.e. weights from other layers are frozen), 3) transfering the weights from a base model and fine-tuning all layers. According to their experiments, the best performance was achieved in the 3rd scenario where the weights of the target model are initialized from the weights of a previously trained base model and then all layers are fine-tuned on the Lymph node dataset. Also, scenario 1 achieved the worst performance. This is expected since the two datasets are very different and the features learnt by a model trained on ImageNet are not general enough to be used as is on a medical imaging dataset. Tajbakhsh et al. [156] conducted a similar study on transferring pre-trained weights from *AlexNet* trained on ImageNet to 4 medical imaging datasets. Based on their findings, initializing the weights to a pre-trained model and fine-tuning all layers should be preferred to training from scratch, regardless of the size of

the dataset. However, if the target dataset is smaller we should be expecting a better gain in performance compared to when the target dataset is sufficiently large. They also observed that transfer learning increases the convergence speed on the target model. Also, since the natural scene image datasets such as ImageNet are very different from medical imaging datasets, we are better off fine-tuning all the layers of the model as opposed to only fine-tuning the last few layers. Van et al. [166] also came to a similar conclusion.

Another approach to transfer learning is to initialize the model to weights which have been pre-trained separately in an unsupervised way using models such as *Autoencoders* or *RBMs* [89]. This allows the weights of the target model to be initialized in a better *basin of attraction* [41]. In their lung segmentation problem where they had access to a large un-annotated dataset and a smaller annotated dataset, Schlegl et al. [136] used convolutional restricted boltzmann machine to pre-train a CNN model in an unsupervised fashion. A shallow model is used as it helps the unsupervised model to learn more general features and less domain specific features.

Missing modalities

Different modalities in MRI need to be acquired separately and it often happens that different subjects are missing some modalities. The most common practice is to prepare the dataset using modalities that exist for most subjects. This leads to either discarding some subjects from the dataset or discarding some modalities which are not present in all subjects. Another approach is to impute the missing modalities by zero or the mean value of the missing modality. Li et al. [95] used a 3 dimensional CNN architecture to predict a PET modality given a set of MRI modalities. Van et al. [167] proposed to synthesize one missing modality by sampling from the hidden layer representations of a Restricted Boltzmann Machine (RBM). They perform their experiments on BRATS 2013 using a patch wise training approach. For every training patch, they train the RBM with every modality to learn the joint probability distribution of the four modalities. At test time, when only one of the modalities is missing, they can estimate the missing modality by sampling from the hidden representation vector.

4.6 Future Outlook

Although deep learning methods have proven to have potential in medical image analysis applications, their performance depends highly on the quality of the pre-processing and/or the post processing. These methods tend to perform poorly when input data do not follow a common distribution which is often the case. Learning robust representations which are invariant to the noise introduced by the acquisition is needed. Unsupervised learning or weakly supervised learning might hold the key to this problem. Also methods based on domain adaptation might help us learn representations which can better generalize across datasets.

Chapter 5

Within-Brain Classification for Brain Tumor Segmentation

Résumé

As discussed in previous chapters, the image distribution from one patient to another, can vary significantly. Machine learning methods which use data across patients rely on pre-processing methods to bring data distributions close together. In this work, we propose an alternative approach which requires minimum pre-processing. In this approach, the training and generalization is done *within* a single brain. While requiring minimum user interaction, we increase generalization accuracy. Taking into consideration the physical characteristics of the tumor that the tumor cells are localized, we propose to use the spatial feature coordinates as extensions to image intensity features.

Commentaires

This article was published in international journal of computer assisted radiology and surgery in 2015.
The initial crude idea was proposed by the Ph.D candidate's supervisors which was refined and extended by the Ph.D. candidate. He was involved in the

development of the method from the beginning. The entire MATLAB code and part of the python code associated to this project was developed by the Ph.D. candidate. He carried out experiments, results submissions and wrote most parts of the paper.

Within-Brain Classification for Brain Tumor Segmentation

Mohammad Havaei
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
seyed.mohammad.havaei@usherbrooke.ca

Hugo Larochelle
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
hugo.larochelle@usherbrooke.ca

Philippe Poulin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
Philippe.Poulin2@usherbrooke.ca

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Keywords: Brain tumor segmentation machine learning

Abstract

Purpose: In this paper, we investigate a framework for interactive brain tumor segmentation which, at its core, treats the problem of interactive brain tumor segmentation as a machine learning problem.

Methods: This method has an advantage over typical machine learning methods for this task where generalization is made across brains. The problem with these methods is that they need to deal with intensity bias correction and other MRI-specific noise. In this paper, we avoid these issues by approaching the problem as one of *within brain generalization*. Specifically, we propose a semi-automatic method that segments a brain tumor by training and generalizing within that brain only, based on some minimum user interaction.

Conclusion: We investigate how adding spatial feature coordinates (i.e. i, j, k) to the intensity features can significantly improve the performance of different classification methods such as SVM, kNN and random forests. This

would only be possible within an interactive framework. We also investigate the use of a more appropriate kernel and the adaptation of hyper-parameters specifically for each brain.
Results: As a result of these experiments, we obtain an interactive method whose results reported on the MICCAI-BRATS 2013 dataset are the second most accurate compared to published methods, while using significantly less memory and processing power than most state-of-the-art methods.

5.1 Introduction

Brain tumor segmentation is primarily used for diagnosis, patient monitoring, treatment planning, neurosurgery planning and radiotherapy planning. The task of brain tumor segmentation is to locate the tumor and delineate different sub-regions of the tumor, namely edema, non-enhanced, and enhanced regions (see Fig. 1). A standard way to diagnose a brain tumor is by using magnetic resonance imaging (MRI), for which many different modalities can be used. The most frequent MRI modalities used for brain tumor segmentation are Flair, T1-weighted (also referred to as T1), T2- weighted (also referred to as T2) and T1-weighted contrast-enhanced (gadolinium-DTPA) which we refer to as T1C. These different modalities are often used jointly as they provide complementary information for locating tumors.

Unfortunately, tumors (especially glioblastomas and metastases) can appear almost anywhere in the brain. They have no prior shape, and often have poorly defined edges. Also, they visually present themselves in grayscales that are present in healthy tissues as well. As a consequence, brain tumor segmentation in practice is still done manually. Manual segmentation is not only time consuming and tedious, it is also subject to variations between observers and also within the same observer [137].

Many methods have been proposed to facilitate the tumor segmentation process. Among them, *automatic* methods, which rely on machine learning, are very popular and in some cases very efficient [12]. These methods are trained on a number of subjects and generalize on data which might be gathered from different MRI scanners. Because there is no intensity standardization among MRI scanners, this makes generalization difficult for automatic methods. In an attempt to overcome these difficulties, a lot of prepossessing steps are made which can be time consuming. Also, to improve generalization, these methods often compute high dimensional feature vectors [137] which add to the processing time and take up a

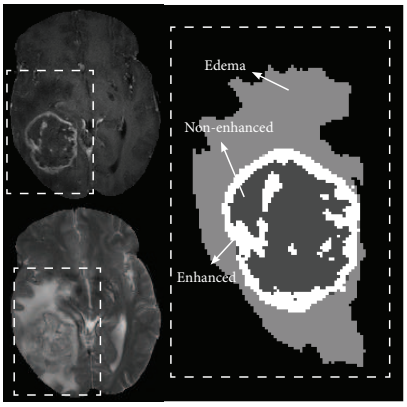


Figure 5.1 – **Left:** T1C and T2 modality. **Right:** groundtruth tumor segmentation.

lot of memory.

In this paper, we consider the specific problem of segmenting an imaged brain into 4 classes: edema, non-enhancing tumor, enhancing tumor and healthy tissue (see Fig. 5.1). Note that the non-enhancing tumor sometimes includes necrotic tissue. Our approach is halfway between automatic and semi-automatic methods. While machine learning methods train on a pre-selected set of brains and then generalize to testing brains, our method implements a “single brain" supervised learning method. The user roughly selects brain voxels associated to each class and then these voxels are used as training data. The method then generalizes by labeling non-selected voxels.

The main characteristics of our method are as follows:

— Since it treats each brain as a separate dataset, it is immune to the multi-MRI disadvantages mentioned above.
— Although it uses only 6 simple features, it produces highly accurate results.
— The segmentation process for a $240 \times 240 \times 168$ brain takes approximately 10 seconds for our fastest method which is much faster than most state-of-the-art methods which can take up to 100 minutes.
— The method is extremely memory efficient (50 Mb vs. $>$2 Gb for other methods)

In this paper we first evaluate this framework on variations of three popular machine learning methods namely; k nearest neighbor classifier (kNN), support vector machines (SVM), random forests and boosted decision trees. Having confirmed that SVMs give superior results, we propose better distance metrics to be used by SVM classifier in the context of this approach. We also investigate the importance of performing hyper-parameter selection individually for each brain, as opposed to using generic hyper-parameters for every brain. Thanks to this investigation, we were able to significantly improve the resulting brain segmentation system and achieve a competitive performance compared to the methods submitted to the brain tumor segmentation challenge online evaluation benchmark [104].

5.2 Related Work

Brain tumor segmentation methods can be divided into *automatic* methods and *semi-automatic* (interactive) methods. Semi-automatic methods are those relying on user interaction. Most of these methods use either deformable models or classification methods to perform segmentation (see Bauer et al. [12] for a survey).

For automatic methods, machine learning classification techniques are a tool of choice for designing such systems, as they can easily integrate different MRI modalities as well as other features. After integrating different intensity and texture features, these methods decide to which class each voxel belongs to.

For instance, Festa et al. [104] used a series of intensity and texture based features to make a feature space of over 300 dimensions, on which a random forest classifier was trained. Tustison et al. and Reza et al. also used random forests [104]. Tustison et al. constructed a multi-dimensional feature space by incorporating first order neighborhood statistical images, GMM and Markov Random Field (MRF) posteriors, and template differences. [93] performed binary segmentation (tumor vs. non-tumor) using T1, T2, T1C in an SVM framework followed by a variation of conditional random fields to account for neighborhood relationships. [10] used a kernel SVM for multiclass segmentation of brain tumors, where a CRF is used to regularize the results.

Schmidt et al. [137] compared the combination of many different feature sets, such as binary mask, average intensity, left to right symmetry. Luts et al. [98] also compared different feature selection methods such as Fisher discriminant analysis, Kruskal wallis, relief-f and ARD for LS-SVM.

Because automatic methods train on multiple brains, these methods are vulnerable to the variations in the MRI data. These variations come from the fact that MR images are generated by different machines and each have their own unique noise and intensity level. To overcome this difficulty, most of these methods rely on a large number of features, which requires a lot of memory and computation time.

As for semi-automatic methods, deformable models are often employed. These algorithms are usually initialized by a user drawing a contour around the tumor. Following an energy minimization criterion, the contour shrinks down towards the borders of the

tumor [76, 172]. Hamamci et al. [60] used a so-called CA-based method on T1 weighted images to produce a probability map for the tumor, based on seeds provided by the user. This probability map is later used in a level set framework. Later, they extend their method to accept multi-modal MRI inputs namely T1C and Flair. For a two class segmentation (tumor, edema) this method takes 1 minute for user interaction and 10-20 minutes for segmentation depending on the size of the tumor [59]. There exists a line of research focusing on how to efficiently initialize the active contour and thus remove user interaction. In this context, the location of the tumor is roughly determined by some other method and deformable models are used as post-processing for refinement. Ho et al. [67] use the difference between T1 and T1C together with a Gaussian mixture model (GMM) to get a probability map of the tumor, which is used in a level-set model to initialize the contour. Prastawa et al. [118] used voxel registration with an atlas as a way to get a probability map for abnormalities. An active contour is then initialized using this probability map and iterates until the change in posterior probability is below a certain threshold.

Although deformable models have been popular in medical image analysis, they have some significant disadvantages. Because these methods rely on image gradients, they are likely to fail when the object of interest does not have well defined borders. The contour may get attracted by strong gradients from surrounding objects. Incorporating different features into the model is also non-trivial. Finally, without a GPU implementation, these methods can be extremely slow.

There has been research on ensembling results from multiple methods applied to brain tumor segmentation. Huo et al. [72] used three segmentation methods: fuzzy connectedness, GrowCut and voxel classification using SVM to generate candidate segmentations for each voxel. Confidance-based averaging (CMA) was used to make the ensemble.

Although our approach is a semi-automatic method, it shares with automatic methods the use of a machine learning classification algorithm, ran on a feature representation of voxels and improved by a spatial dependency model. The main difference is that generalization is performed *within* each brain, based on the training data provided by the user's interaction. This simplified generalization problem allows us to use a very simple feature space, yielding an interactive segmentation method that is fast and ef-

fective. [164] used a similar, semi-automatic, kNN classification method, applied to proton density, T1 and T2 modalities. [19] also proposed a semi-automatic segmentation method that uses instead Quadratic Discriminative Aanalysis to perform multi-class segmentation. However, they did not use the $\langle i, j, k \rangle$ voxel positions as features (see Section 5.3.2) nor did they deal with label spatial dependency modeling (see Section 5.3.4), which we found to play a crucial role in obtaining competitive performances.

5.3 Investigating Within-Brain Generalization

Within-brain generalization treats the segmentation of each brain as its own machine learning experiment, in which a classifier is trained (on user-labeled voxels) and used to generalize to new observations (voxels not labeled by the user).

This approach is motivated by the observation that, with current computers and for relatively small data sets with small feature spaces, a machine learning experiment (including hyper-parameter selection) can actually be performed within a very short delay, even for more sophisticated algorithms that require more than simply storing the data (as in kNN). Moreover, segmenting only within a given brain removes the challenging problem of generalizing across brain imaging acquisition conditions.

In what follows, we describe the details of our approach and enumerate the different variations we explored in this direction.

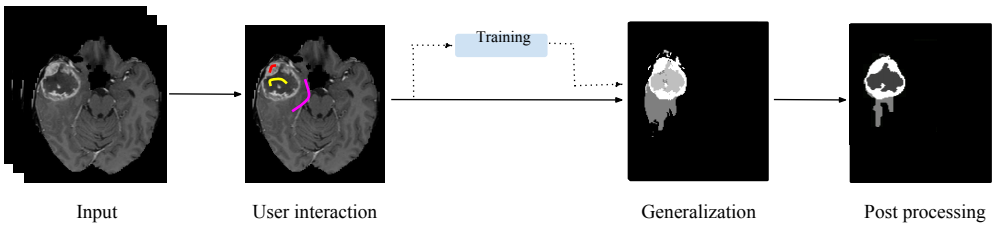


Figure 5.2 – Our method in a nutshell. The segmentation is performed on the entire brain based on data provided by user interaction.

Figure 5.2 shows our method in a nutshell. We explain these steps in Section 5.3.

5.3.1 Feature representation and manual selection

The first step of our method is to collect voxel label data for a given brain image to segment. This is done by the user who roughly selects a subset of voxels associated with each class, through a graphical interface. The number of strokes required for obtaining the training data depends on the number of tumors in a given brain. However, usually one or two strokes per-class is enough. The user interaction step takes 1 minute on average and up to 2 minutes for complicated tumors or noisy MRIs. We will note as M a binary mask such that $M_v \in \{0, 1\}$ indicates whether a voxel v has been manually selected (*i.e.* labeled) or not. T will then be the class-selection mask where $T_v \in \{\text{edema, non-enhancing tumor, enhancing tumor, healthy}\}$ is the class label associated with the voxel v by the user.

We must also decide on a feature representation for the different voxels. Each brain image I is assumed to come with 3 MRI modalities (T1C, T2, Flair), such that I is a tensor where each voxel v in I is a 3D vector containing the grayscale values of the modalities. These modalities are often chosen because of their discriminative power. In fact, while the non-enhanced necrosis vs edema can be distinguished from T1C modality, the non-enhanced active area and the edema can be distinguished with the Flair modality. This is represented by I_v^1, I_v^2, I_v^3. By converting each voxel v to an N-dimensional feature representation F_v, it will be possible to train a classifier to predict the voxel label T_v, for every voxel, from its feature representation. We propose a simple 6 dimensional feature represeentation, which consists of the MRI modality gray scales and the 3d position of voxel v: $F_v = (I_v^1, I_v^2, I_v^3, i, j, k)$. These features are normalized between zero and one.

At this point, from each labeled voxel, we can thus generate a training pair (F_v, T_v) and construct a training set $\mathcal{D}$ that we shall use to classify the non-selected voxels using a classifier.

5.3.2 Voxel classifiers

Having built the training set through manual interaction, the next step is to train a classifier and generalize the segmentation to non-selected voxels. We investigate the use of different machine learning algorithms to produce a classifier. While we could, theoreti-

cally, consider any existing algorithm, it is natural to prefer algorithms that are known to be robust and fairly "black box" in their use. For instance, we do not want the user (typically a doctor or a neuro-scientist) to have to manually tune hyper-parameters for each brain, with trial and error. So we chose algorithms that are known to be easily tuned or for which default values of their hyper-parameters tend to work well. These algorithms have also shown to be successful for automatic brain tumor segmentation [137, 104].

K-Nearest Neighbors (kNN)

To start, k nearest neighbor (kNN), one of the simplest classifiers, is considered. For every voxel v, kNN finds among the training data $\mathcal{D}$, the set of k nearest neighbors ($\mathcal{N}_v$) based on F_v. Let $\mathcal{N}_v = ((F_{v_1}, T_{v_1}), (F_{v_2}, T_{v_2}), ..., (F_{v_k}, T_{v_k}))$ where F_{v_i} is the i^{th} closest training point of F_v. The kNN classification rule assigns a class label to some voxel v following this equation

$$T_v = \arg\max_c \frac{1}{k} \sum_{(F_{v_i}, T_{v_i}) \in \mathcal{N}_v} \delta(T_{v_i}, c) \tag{5.1}$$

where c is a class label and $\delta(a, b)$ returns 1 when $a = b$ and 0 otherwise. Note that this formulation can be seen as using a posterior class probability:

$$p(T_v = c|F_v) = \frac{1}{k} \sum_{(F_{v_i}, T_{v_i}) \in \mathcal{N}_v} \delta(T_{v_i}, c) \tag{5.2}$$

which states that the probability of an observation F_v of being in class c is given by the proportion of nearest neighbors assigned to that class. This probabilistic formulation of the classifier will be reused for the unary terms of a CRF, described in Section 5.3.4.

Support Vector Machine

The support vector machine (SVM) [35] is probably the most frequently used classifier. This is in part due to the existence of many freely available, mature and easy-to-use implementations. In its parametric form, it is a linear classifier that attempts to classify data points by maximizing the margin between the decision boundaries of the different

classes and their closest points.

Of higher interest in our setting is the kernelized version of SVM [88]. A choice for the kernel that often proves successful is the radial basis function (RBF) kernel:

$$\mathcal{K}(F_j, F_v) = \exp(\text{-}\gamma \parallel F_j - F_v \parallel_2^2). \tag{5.3}$$

where γ is a hyper-parameter. Also, a slack variable C is used to relax the constraints in the SVM optimization problem [88]. The resulting classifier effectively takes the form of a template matcher, that compares a given input with all training examples, each voting for their class with a weight related to their similarity with the input (as modeled by the kernel). In this sense, it is similar to the kNN classifier, though the former often outperforms the later in practice.

It is also possible to obtain a posterior class probability $p(T_v = c|F_v)$ from the SVM. This is done by training the parameters of an additional sigmoid function of the form

$$P(T_v = c|F_v) = \frac{1}{1 + \exp\left(Af(F_v, c) + B\right)} \tag{5.4}$$

where $f(F_v, c)$ is the unthresholded output of the SVM and A, B are the parameters to be estimated [114]. Here again, the posterior probability function will be used later on, for the CRF unary term.

Ensemble of Decision Trees

Another popular approach to classification are ensembles of decision trees. Each decision tree is trained by recursively partitioning the feature space, according to some heuristic that favors a good separation of classes. Once a criterion for stopping the tree growth is reached, a conditional class distribution is then computed at each leaf, based on the training data falling into the corresponding partition. Specifically, the class distribution $p(T_v = c|F_v)$ is set as

$$P(T_v = c|F_v) = \frac{N_c}{N} \tag{5.5}$$

where N_c is the relative frequency of examples belonging to class c of the partition in which F_v falls and N is the total number of examples.

The performance of a single decision tree is often disappointing. However, by constructing an ensemble of such trees, a competitive classification performance is achievable. There are different approaches to combining decision trees into an ensemble. The two most popular algorithms for ensembles of decision trees are random forests and Adaboost [108]. We considered these two algorithms for our experiments.

5.3.3 Distance Metric/Kernel

The performances of the SVM classifier often depends on the choice of metric or kernel used to compare data points. Thus, it is generally beneficial to adapt this choice to each individual problem. For example, the conventional RBF kernel puts equal weight to each dimension of the feature space. However, in our within-brain framework, the spatial coordinate features $\langle i, j, k \rangle$ and the modality features actually play different roles. Intuitively, one role of the spatial coordinates is to avoid that a user-labeled voxel starts influencing the prediction made at a voxel far away from it, e.g. to avoid false positives in faraway regions. The modality features, are thus mostly informative within the vicinity of a user-labeled voxel.

Therefore, we might want to weight the modality and spatial features differently, within the RBF kernel of the SVM. To maintain positive-semidefiniteness of the kernel, we simply opt for using two different values of γ for MRI modality intensities and the spatial features:

$$\begin{aligned} \mathcal{K}(F_j, F_v) = \exp(\quad & -\gamma_1 \quad \| \, F_{j,\{1:N\}} - F_{v,\{1:N\}} \, \|_2^2 \\ & -\gamma_2 \quad \| \, F_{j,\{N+1:N+3\}} - F_{v,\{N+1:N+3\}} \, \|_2^2). \end{aligned} \tag{5.6}$$

This kernel is also equivalent to the product of two RBF kernels, each defined on the subspace of modalities and of spatial coordinates, and each having their own hyper-parameters. The hyper-parameters required by this approach are γ_1 and γ_2.

5.3.4 Importance of Within-Brain Hyper-Parameter Selection

When training a classifier, hyper-parameter values must be specified. One approach which is commonly implemented [104] is to choose hyper-parameters by cross-validation in a grid search approach on a subset of brains and fix the selected set of hyper-parameters for the rest of the brains. We hypothesize given the variations in MRI data, using a fixed set of hyper-parameters for generalization is not optimal. An alternative way is to perform hyper-parameter selection individually for each brain, in order to adapt to the specificity of each case. We measure the potential gains of this approach in our experiments when selecting the hyper-parameters for the SVM, namely the slack variable C and the coefficient γ. A detailed discussion of this experiment is presented in section 5.4.2.

Conditional Random Fields (CRF)

As mentioned earlier, segmentation accuracy can easily be improved by leveraging a model of the 3D spatial regularity of labels. One way of enforcing spacial regularity is to define a joint (conditional) distribution over the labels of all voxels in the brain that expresses the expected dependencies between neighboring voxels. Conditional Random Fields (CRF) provide a convenient formalism for that. CRFs model directly the posterior probabilities of the labels given the features $P(T|F)$ directly, alleviating the need to model the distribution over the feature vectors F and allowing us to construct rich conditionals $P(T|F)$.

Formally speaking, we use the following form for $P(T|F)$:

$$P(T|F) = \frac{1}{Z} \prod_{v} \phi(F_v, T_v)\phi(T_v, F_v, T_r, F_r) \quad \text{where } r \in \eta_v \tag{5.7}$$

where Z is a normalization term, ϕ are clique potential functions and η_v is the set of voxels surrounding v.

Segmenting a brain requires that we find the labeling T with highest probability $P(T|F)$. This leads to an optimization problem of the form $T = \arg\max_T \prod_v \phi(F_v, T_v)\phi(T_v, T_r)$

or, equivalently,

$$T = \arg\min_{T \in \mathcal{T}} \sum_{v} \left(V(F_v, T_v) + \sum_{r \in \eta_v} I(T_v, F_v, T_r, F_r) \right). \tag{5.8}$$

where we set the equivalence $V(F_v, T_v) = -\log \phi(F_v, T_v)$ and $I(T_v, F_v, T_r, F_r) = -\log \phi(T_v, F_v, T_r, F_r)$.

In our case, we model the unary terms $V(F_v, T_v)$ by taking the negative log of the posterior distribution

$$V(F_v, T_v) = -\log P(T_v | F_v) \tag{5.9}$$

specified in Eq.(5.2), (5.4) or (5.5). As for the pairwise term, we set it to be

$$I(T_v, F_v, T_r, F_r) = \lambda(1 - \delta(T_v, T_r)) \exp(\frac{-|F_v - F_r|}{\sigma^2}). \tag{5.10}$$

The choice of these unary and pairwise terms allows us to perform the optimization of Equation 5.8 using the graphcut algorithm.

We refer to the segmentation methods using this label dependency model as **kNN-CRF**, **SVM-CRF**, and **DT-CRF**, depending on the unary term used.

5.4 Experiments

5.4.1 Experimental Setup

All our experiments were conducted on real patient data obtained from the brain tumor segmentation challenge dataset (BRATS2013) (Menze et al. [104]) as part of the MICCAI conference. The BRATS2013 dataset is comprised of 3 sub-datasets. The training dataset, which contains 30 patient subjects all with pixel-accurate ground truth (20 high grade and 10 low grade tumors); the test dataset which contains 10 (all high grade tumors) and the leaderboard dataset which contains 25 patient subjects (21 high grade and 4 low grade tumors). There is no ground truth provided for the test and leaderboard datasets. For each subject there exist 4 modalities which are co-aligned together,

namely: T1, T1C, T2 and Flair . In our experiments, we used T1C, T2 and Flair only. We found T1 to be redundant with T1C and using it did not improve the overall performance of the model. For each brain, the user is asked to manually label voxels in only two 2D slices for each class. The choice of slices depend on the size and spread of the tumor. Considering the fact that the user can choose slices from any view (i.e. axial, sagittal and coronal), the tumor coverage is sufficient and the results are not very sensitive to the slices chosen for labeling. On average, only 0.4% of the voxels containing pathology and 0.03% of the voxels corresponding to healthy tissue were manually selected, thus providing minimal labeled data to the algorithm. To make operations faster, we disregard all the voxels outside of the skull and consider them as healthy.

The quantitative results for each method was obtained from the BRATS online evaluation system, which provides Dice, Specificity and Sensitivity as measures of performance. These measures are defined as follows:

$$\begin{aligned} Dice(P,T) &= \frac{|P_1 \wedge T_1|}{(|P_1| + |T_1|)/2}, \\ Sensitivity(P,T) &= \frac{|P_1 \wedge T_1|}{|T_1|}, \\ Specificity(P,T) &= \frac{|P_0 \wedge T_0|}{|T_0|}, \end{aligned}$$

where $\wedge$ is the logical AND operation, P represents the model predictions and T represents the ground truth labels. We also note as T_1 and T_0 the subset of voxels predicted as positives and negatives for the tumor region in question. Similarly for P_1 and P_0 [104].

We report these measures for the test subjects over the three categories considered by the BRATS evaluation (i.e. complete, core, enhanced). The complete category is the union of classes containing un-healthy tissue. i.e. $\{l|l \in [\text{necrosis, edema, enhancing}]\}$), the core category are classes containing tumor core i.e. $\{l|l \in [\text{necrosis, enhancing}]\}$ and the enhancing category is the enhancing tumor class. i.e. $\{l|l \in [\text{enhancing}]\}$. The online evaluation system also provides a ranking for every method submitted for evaluation. This includes methods from the 2013 BRATS challenge published in [104] as well as anonymized unpublished methods for which no reference is available. The methods in each table presented in this section are ordered according to the ranking provided by the

online evaluation system.

Please note that we could not use the BRATS 2014 dataset due problems with both the system performing the evaluation and the quality of the labeled data. For these reasons the old BRATS 2014 dataset has been removed from the official website and, at the time of submitting this manuscript, the BRATS website still showed: “Final data for BRATS 2014 to be released soon” For these reasons, we decided to focus on the BRATS 2013 data. Also, this article does not contain any studies with human participants performed by any of the authors.

5.4.2 Results and Discussion

In this section, we report experimental results obtained with the machine learning methods presented in Section 5.3.2. This includes linear SVM (LSVM), kernel SVM with rbf kernel (KSVM), our proposed product kernel SVM (PKSVM), kNN, decision trees trained with Ada-Boost (ADT), and random forests (RDT). All these methods have been explored with and without the CRF. The CRF parameters α and β were set for each method, by cross-validation on 6 brains on the training set. We also investigate the extent to which adding spatial features $\langle i, j, k \rangle$ helps improving the performance. This is noted by adding a “$*$" next to the method’s name.

KNN

The results for the kNN related experiments are presented in Table 5.1. We first made an experiment without including the $\langle i, j, k \rangle$ position features in the feature vector as presented by [164]. Since his method uses neither the spatial coordinate features nor the CRF regularization, it performs significantly worse than other kNN related experiments. While adding the spatial coordinates to this method improves the result by a significant margin, the best performance is achieved when we use both spatial coordinates and a CRF regularization.

Table 5.1 – Dice, Specificity and Sensitivity measures for kNN methods on BRATS-2013 test set. "$*$" shows the use of spatial features.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
kNN-CRF*	**0.85**	**0.75**	0.60	**0.91**	**0.85**	**0.77**	0.78	0.69	0.56
kNN*	0.81	0.68	**0.65**	0.76	0.62	0.62	**0.90**	**0.84**	**0.73**
kNN-CRF	0.80	0.69	0.55	0.92	0.83	0.75	0.74	0.63	0.48
kNN	0.65	0.52	0.53	0.59	0.49	0.50	0.77	0.68	0.65

Table 5.2 – Dice, Specificity and Sensitivity measures for various SVM methods on the BRATS-2013 test set. "$*$" shows the use of spatial features.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
PKSVM-CRF*	**0.86**	**0.77**	**0.73**	**0.88**	**0.85**	**0.76**	0.78	0.68	0.58
KSVM-CRF*	0.84	0.75	0.70	0.87	0.77	0.72	0.82	0.79	0.71
PKSVM*	0.82	0.71	0.69	0.84	0.73	0.71	0.80	0.76	0.71
KSVM*	0.81	0.68	0.65	0.76	0.62	0.62	**0.90**	**0.84**	**0.73**
KSVM-CRF	0.74	0.67	0.53	0.82	0.82	0.79	0.73	0.61	0.45
LSVM-CRF*	0.79	0.64	0.51	0.86	0.74	0.70	0.74	0.62	0.45
LSVM*	0.69	0.59	0.62	0.65	0.54	0.47	0.84	0.76	0.59
LSVM-CRF	0.72	0.60	0.46	0.77	0.66	0.59	0.72	0.61	0.44
KSVM	0.65	0.50	0.50	0.61	0.49	0.49	0.75	0.63	0.58
LSVM	0.51	0.35	0.45	0.48	0.35	0.43	0.73	0.59	0.59

SVM

The results for the SVM-related experiments are presented in Table 5.2. Results confirm that using spatial coordinate features (shown with "*") and using the CRF model (shown with "-CRF") improve the performance of both a linear SVM (LSVM) and an RBF kernel SVM (KSVM). It is also quite clear from this experiment that the non-linearity of the kernel SVM is crucial, as it significantly outperforms the linear SVM (LSVM).

As for the PKSVM method which stands for the RBF product kernel SVM presented in Section 5.3.3 (c.f. Eq.(5.7)) it clearly improved the Kernel-SVM and Kernel-SVM+CRF results. This underlines the relative importance of the spatial coordinate features $\langle i, j, k \rangle$ versus the input T1, T2 and Flair modalities.

Table 5.3 – Dice, Specificity and Sensitivity measures for ensemble of decision trees with AdaBoost (ADT) and random forests (RDT) on BRATS-2013 test dataset. "$*$" shows the use of spatial features.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
RDT*	0.81	**0.69**	**0.64**	0.83	0.71	0.64	**0.79**	**0.75**	**0.70**
RDT-CRF*	**0.82**	**0.69**	0.51	**0.92**	**0.83**	**0.79**	0.73	0.61	0.50
RDT-CRF	0.80	0.66	0.49	**0.92**	**0.83**	0.78	0.71	0.60	0.40
ADT-CRF*	0.79	0.64	0.51	0.88	0.75	0.71	0.72	0.61	0.45
ADT-CRF	0.78	0.63	0.50	0.87	0.73	0.67	0.72	0.61	0.45
ADT*	0.73	0.57	0.58	0.73	0.60	0.59	0.75	0.64	0.66
RDT	0.67	0.55	0.55	0.66	0.55	0.53	0.72	0.65	0.65
ADT	0.65	0.48	0.54	0.66	0.55	0.53	0.69	0.52	0.62

Decision trees

For these experiments, we fixed the number of decision trees for AdaBoost (ADT) and random forests (RDT) to 100 and the leaf size to 1. For AdaBoost, decision stumps were used. The quantitative results are shown in Table 5.3. While adding spatial features are beneficial for both random forests and AdaBoost, using the CRF model is mostly beneficial except for random forest without spatial coordinates. However, the segmentation systems relying on decision trees tend to be worse than using kNN or SVM methods.

Robustness of hyper-parameter selection

In our method when using the SVM as the classifier, the hyper-parameters (regularization constant C and kernel hyper-parameters γ, γ_1 and γ_2) were always cross-validated for each brain individually, using an automated grid search. For this purpose we create a smaller training and validation set (with proportions of 70% for the training set and 30% for validation set) from the sub-sampled interaction points. The hyper-parameters are selected based on the performance on the validation set. On the other hand, for automatic methods, a fixed set of hyper-parameters is used for generalization. Given the variation of the MRI data and tumor types, we hypothesize that using a fixed set of hyper-parameters will degrade the performance quite significantly.

To evaluate the importance of performing per-brain model selection, we conducted an experiment where we used a fixed configuration of hyper-parameters for all subjects.

For this experiment, we considered our top two segmentation methods, PKSVM-CRF* and KSVM-CRF*. The values of the hyper-parameters were chosen by taking the hyper-parameter value most frequently selected by these methods, across all the brains. The idea was to pick values that are most likely to work well in general. For the KSVM-CRF*, C was set to 1 and γ to 5 and for the PKSVM-CRF*, C was set to 1, γ_1 to 100 and γ_2 to 10.

The results (Table 5.4) show a decrease in performance if fixed hyper-parameters are used for all brains. We also performed this experiment on the BRATS training data (not shown here) and the performance decreased even more. This was not unexpected, since the training data is more varied and actually consists of both high grade tumors and low grade tumors, while the test data only contains high grade tumors.

While it appears the tuning of the SVM's hyper-parameter to each brain is beneficial, we tested the extent to which small changes to the optimal hyper-parameters would affect the performance. This is meant to simulate the fact that cross-validation might not always find the same hyper-parameters between variations on the manually labeled voxels. In order to measure how resilient our method is to slight hyper-parametric shifts, we ran another experiment to measure the sensitivity of our model. We did so by randomly selecting 20 brains from the BRATS training data, trained an SVM whose hyper-parameters have been obtained from cross validation. We then added noise to the hyper-parameters and measured the effect on the resulting segmentation. The noise corresponded to Gaussian noise, whose standard deviation was set to a certain percentage of the hyper-parameters' values. Figure 5.3 shows the resulting Dice measure for different noise level. As one can see, even with a noise level corresponding to a corruption of 25% of the hyper-parameter values, the end result is still close to the one obtained without any noise.

Finally, the importance of optimizing the hyper-parameters was found to be less crucial for the other methods. For kNN, we evaluated the effect of using different values of k, with $k = 3$ consistently producing higher performance. The same type of experiment was performed to measure the effect of using different number of trees and leaf size in ADT and RDT. For these methods, setting the number of decision trees to 100 and leaf size to 1 always worked well.

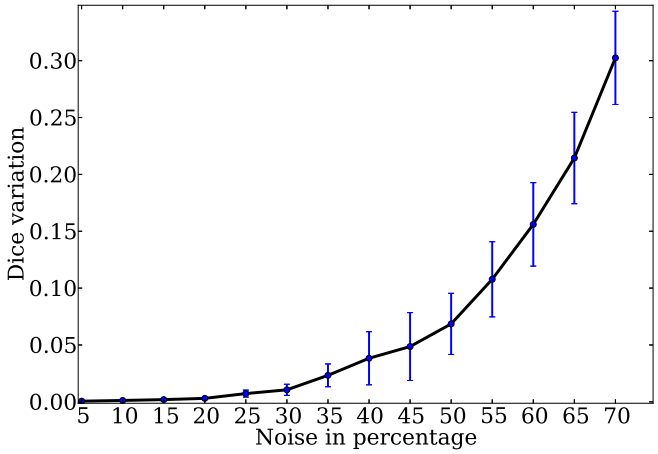


Figure 5.3 – Sensitivity of the model with respect to the gamma hyper parameter.

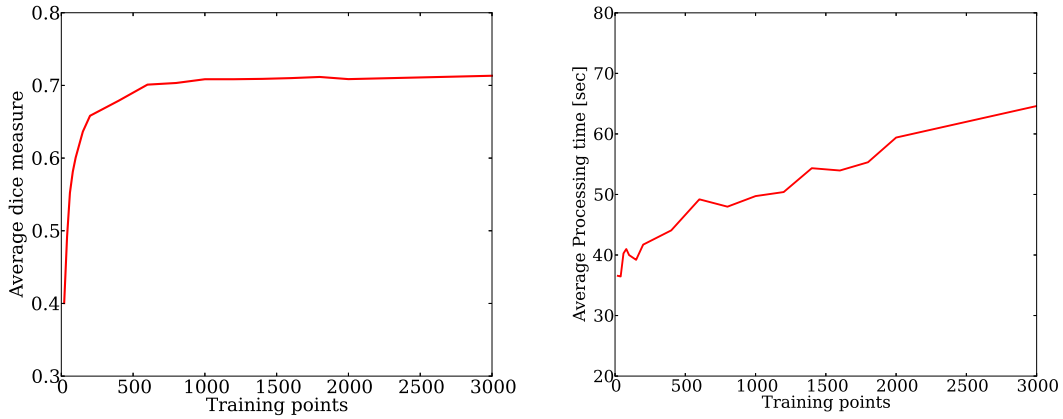
Table 5.4 – The effect of having a fixed selection of hyper-parameters for kernel SVM and product kernel SVM. "$*$" shows the use of spatial features.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
PKSVM-CRF*	**0.86**	**0.77**	**0.73**	0.88	**0.85**	**0.76**	0.78	0.68	0.58
KSVM-CRF*	0.84	0.75	0.70	0.87	0.77	0.72	**0.82**	**0.79**	**0.71**
FixedKSVM-CRF*	0.82	0.69	0.56	**0.93**	0.82	0.78	0.75	0.64	0.49
FixedPSVM-CRF*	0.72	0.56	0.55	0.71	0.62	0.58	0.73	0.65	0.65

Speed-up procedure

Every segmentation method presented in this paper uses manually-selected voxels as their input. However, these selected voxels often carry out similar information. That is especially true for neighboring voxels whose $\langle i, j, k \rangle$ position is almost the same, and whose T1,T2, Flair values are likely to be identical. Thus, in order to speed-up the segmentation procedure, one can randomly down-sample the training data. To have an overall idea to what extent we can down-sample the data without hurting too much the overall precision, we conducted an experiment where we divide the training points into healthy and non-healthy subsets and subsample them separately while trying to keep equal proportions in the un-healthy classes and also balanced proportion for the healthy vs union of un-healthy classes. In other words, the *healthy* class comprises of roughly 50% of the training data while *non-enhanced*, *edema* and *enhanced* classes each take about 16%. The outcome of this process is a smaller training set but with roughly the

same proportion of healthy points and non-healthy points. Figure 5.4 shows the result of this experiment. The curves were obtained by averaging the results of 20 randomly selected brains from BRATS training data. The horizontal axes in Figure 5.4 shows the number of training points in the subsampled training set. As shown in Figure 5.4(a), with maximum number of training points (i.e 3000) we get an average Dice measure of 0.72 and by considering 1000 training points the average Dice measure barely drops to 0.71, while the processing time decreases by 60%. Thus, all experiments submitted to the BRATS website were done with this subsampling measure.



(a) (b)

Figure 5.4 – Sensitivity of the model with respect to the number of training points. (a) shows variation in average Dice measure while (b) shows variation in the average processing time and memory usage.

5.5 Conclusion

5.5.1 Putting it all together

We finally present how our top performing methods compare with other state-of-the-art methods. The BRATS official website provides a ranking system for this purpose. However, because the BRATS organizers have recently made all methods anonymous, a complete comparison is not possible. For that reason, we rank our method based on

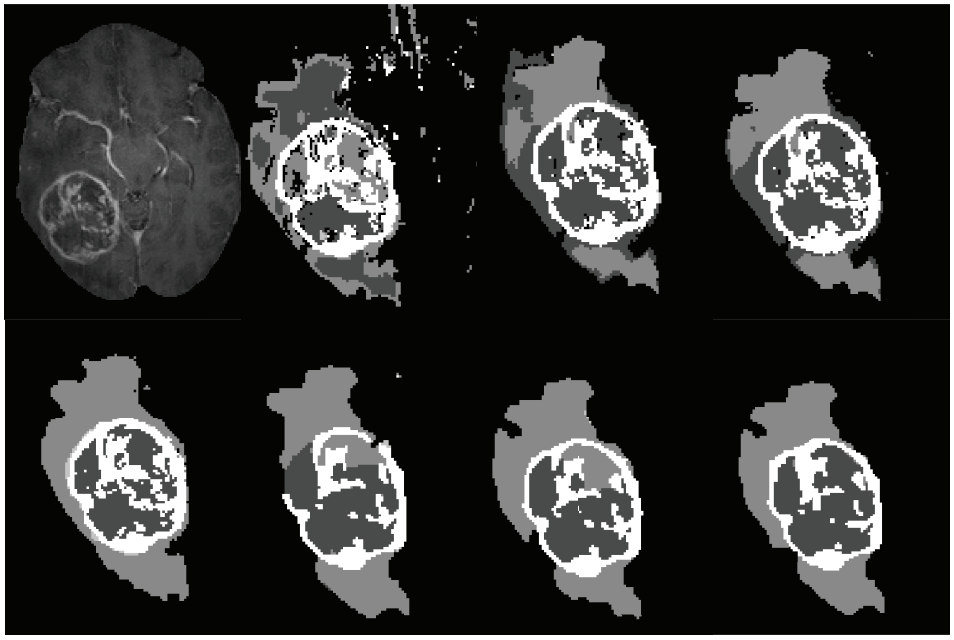


Figure 5.5 – Illustration of brain tumor segmentation maps predicted by different variations of SVM. Top row from left to right : T1C modality, KSVM, KSVM*, PKSVM*. Bottom row from left to right: ground truth, KSVM-CRF, KSVM*-CRF, PKSVM*-CRF.

the MICCAI-BRATS 2013 challenge results for which references to the methods were available. This is shown in table 5.5 [1]. As one can see, PKSVM-CRF* and KSVM-CRF* are ranked second and third respectively, closely behind Tustison et al. and kNN-CRF* is ranked 6^{th} in this table. Using the spatial features $\langle i, j, k \rangle$, and CRF post-processing is vital to produce highly accurate results. Many methods in this table (like that of Tustison et al. Reza et al. and Festa et al.) use random forests with a large number of features. In our case, random forests did not perform as well as the SVM or kNN methods. This might be due to the low dimensionality of our feature space. Recently Subbanna et al. [154] published competitive results on the BRATS 2013 dataset, reporting Dice measures of $0.86, 0.86, 0.77$ for Complete, Core and Enhancing tumor regions. Since they do not report Specificity and Sensitivity measures, a completely fair comparison with that method is not possible. However, as mentioned in [154], their method takes 70 minutes to process a subject, which is significantly slower than our

1. Please note that the results mentioned in Table 5.5 are from methods competing in the BRATS 2013 challenge for which a static table is provided [https://www.virtualskeleton.ch/BRATS/StaticResults2013]. Since then, other methods have been added to the score board but for which no reference is available.

Table 5.5 – Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2013 test set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
Tustison	**0.87**	**0.78**	**0.74**	0.85	0.74	0.69	**0.89**	**0.88**	**0.83**
PKSVM-CRF*	0.86	0.77	0.73	0.88	**0.85**	0.76	0.78	0.68	0.58
KSVM-CRF*	0.84	0.75	0.70	0.87	0.77	0.72	0.82	0.79	0.71
kNN-CRF*	0.85	0.75	0.60	**0.91**	**0.85**	**0.77**	0.78	0.69	0.56
Meier	0.82	0.73	0.69	0.76	0.78	0.71	0.92	0.72	0.73
Reza	0.83	0.72	0.72	0.82	0.81	0.70	0.86	0.69	0.76
Zhao	0.84	0.70	0.65	0.80	0.67	0.65	0.89	0.79	0.70
Cordier	0.84	0.68	0.65	0.88	0.63	0.68	0.81	0.82	0.66
Festa	0.72	0.66	0.67	0.77	0.77	0.70	0.72	0.60	0.70
Doyle	0.71	0.46	0.52	0.66	0.38	0.58	0.87	0.70	0.55

method.

To further validate our model, we present results of our top performing methods on the BRATS 2013 leaderboard and compare it with published methods which reported results on that same dataset. Note that as with BRATS 2013 test set, results from other methods are currently available on the online scoreboard but for which no reference is available. Results of published methods are presented in Table 5.6. As can be seen, our top approaches out perform state-of-the-art methods on this dataset.

Please note that since BRATS2012 dataset is a subset of BRATS2013 leaderboard and that more methods are competing on the BRATS2013 leaderboard, we did not include results for the 2012 dataset.

Figure 5.5 shows a visualisation of segmentation results, for different variations of our SVM method. This illustrates the contribution of adding spatial features, using a CRF and using our improved kernel function, in improving the general performance of the SVM approach.

5.5.2 Processing time and memory usage

A key advantage of our proposed method is in having a very small processing time (1 minute 40 seconds in total which includes the user interaction) and memory usage, while maintaining high accuracy. Due to the low dimensionality of our feature space, it

Table 5.6 – Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2013 leaderboard set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
PKSVM-CRF*	**0.83**	**0.69**	**0.59**	**0.86**	**0.78**	0.55	0.84	0.71	**0.67**
KSVM-CRF*	0.81	0.68	0.56	0.81	0.75	**0.61**	0.83	0.69	0.58
kNN-CRF*	0.79	0.66	0.54	0.77	0.72	0.55	0.85	0.70	0.61
Tustison	0.79	0.65	0.53	0.83	0.70	0.51	0.81	0.73	0.66
Zhao	0.79	0.59	0.47	0.77	0.55	0.50	0.85	**0.77**	0.53
Meier	0.72	0.60	0.53	0.65	0.62	0.48	**0.88**	0.69	0.6
Reza	0.73	0.56	0.51	0.68	0.64	0.48	0.79	0.57	0.63
Cordier	0.75	0.61	0.46	0.79	0.61	0.43	0.78	0.72	0.52

only takes up, on average, 50 MB of RAM to store the feature space of a brain. This is very small compared to state-of-the-art methods, whose memory footprint of the feature space is on the order of GB's. For example, Festa et al. use a feature space of 300 dimensions for their random forest approach which would take up to 2.7GB's. Tustison et al. Reza et al. and Meier et al. also take a similar approach using random forests [104]. These methods rely on a high number of texture features which are computationally time consuming and memory wise expensive.

Apart from the feature space, our proposed methods have different speed and memory footprint. We can make a comparison in accuracy, speed and memory usage as presented in Table 5.7. The processing time was measured on an 8-core processor and includes both training and testing. The time required by graphcut inference is the same for all methods and involves only an additional 8 seconds. As shown in Table 5.7, PKSVM-CRF* has the highest accuracy but requires a higher processing time (35 seconds) and memory usage (7.7 MB), on top of the 50 MB required to store the feature space. On the other hand, KSVM-CRF* and kNN-CRF* are closer to real time implementations with negligeable memory consumption. This allows the expert to interact in real-time with the software. That being said, all methods presented in Table 5.7 are significantly faster than state-of-the-art methods. For example, Tustison's method takes around 30 minutes to process a brain as mentioned in Menze et al. [104].

In this paper we evaluated the capability of *within brain generalization* using a variety of classifiers. We showed that the SVM reached the best performances, thanks in part to a kernel function specifically adapted to our feature space. Most interestingly, we also showed that adopting a fixed hyper-parameter configuration for all brains actually

Table 5.7 – Best performing methods for each machine learning category with average processing time and memory usage.

Method	Dice			Specificity			Sensitivity			Time	Memory
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing		
PKSVM-CRF*	**0.82**	**0.71**	**0.69**	**0.84**	**0.73**	**0.71**	0.80	0.76	0.71	35sec	7.7MB
KSVM-CRF*	0.81	0.68	0.65	0.76	0.62	0.62	**0.90**	**0.84**	**0.73**	10sec	75KB
kNN-CRF*	0.81	0.68	0.65	0.76	0.62	0.62	0.90	0.84	0.73	3sec.	40KB
RDT*	0.81	0.69	0.64	0.83	0.71	0.64	0.79	0.75	0.70	10sec	120KB

decreases the performance of the SVM. A better strategy was to also perform hyper-parameter selection for each brain individually, in order to adapt to the specificities of each brain, further motivating our *within brain generalization* framework.

5.6 Conflict of Interest

The authors declare that they have no conflict of interest.

5.7 Ethical approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

This article does not contain any studies with human participants performed by any of the authors.

Chapter 6

Brain Tumor Segmentation with Deep Neural Networks

Résumé

In this chapter, we present a fully automatic method for brain tumor segmentation based on deep learning. While being very accurate, the method is extremely fast. The motivation for this work comes from recent success of convolutional neural networks in natural image datasets such as ImageNet. While classical machine learning methods rely on high dimensional hand-designed feature vectors, deep learning presents an arena for the model to learn its own features from raw input data. This presents two promising advantages. It removes the need for intermediate methods to extract hand crafted features while learning more robust high level features which eliminate the need for excessive pre-processing steps. We explore different deep learning architectures and training procedures in order to efficiently utilize neural networks for brain tumor segmentation.

Commentaires

This article was submitted to the journal of Medical Image Analysis in 2015 and was accepted for publication in in 2016 [66].

The initial idea was proposed by the professors involved in this project which was refined and extended by the Ph.D. candidate as well as other students involved. The initial python code was developed in University of Montreal (LISA lab) by the Ph.D. candidate and other student co-authors. The project was continued at University of Sherbrooke which with the help of his supervisors, the Ph.D. candidate extended the method, the python code and carried out the experiments. The paper was mostly written by the Ph.D. candidate.

Brain Tumor Segmentation with Deep Neural Networks

Mohammad Havaei
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
seyed.mohammad.havaei@usherbrooke.ca

Axel Davy
École Normale supérieure, Paris, France
axel.davy@ens.fr

David Warde-Farley
Université de Montréal, Montréal, Canada
david.warde-farley

Antoine Biard
École polytechnique, Palaiseau, France
antoine.biard.10@gmail.com

Aaron Courville
Université de Montréal, Montréal, Canada
aaron.courville@umontreal.ca

Yoshua Bengio
Université de Montréal, Montréal, Canada
yoshua.bengio@umontreal.ca

Chris Pal
École Polytechnique de Montréal , Canada
chris.j.pal@gmail.com

Pierre-Marc Jodoin
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
pierre-marc.jodoin@usherbrooke.ca

Hugo Larochelle
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
hugo.larochelle@usherbrooke.ca

Keywords: Brain tumor segmentation, k-nearest neighbour, interactive method, within-brain generalization

Abstract

In this paper, we present a fully automatic brain tumor segmentation method based on Deep Neural Networks (DNNs). The proposed networks are tailored to glioblastomas (both low and high grade) pictured in MR images. By their very nature, these tumors can appear anywhere in the brain and have almost any kind of shape, size, and contrast. These reasons motivate our exploration of a machine learning solution that exploits a flexible, high capacity DNN while being extremely efficient. Here, we give a description of different model choices that we've found to be necessary for obtaining competitive performance. We explore in particular different architectures based on Convolutional Neural Networks (CNN), i.e. DNNs specifically adapted to image data.

We present a novel CNN architecture which differs from those traditionally used in computer vision. Our CNN exploits both local features as well as more global contextual features simultaneously. Also, different from most traditional uses of CNNs, our networks use a final layer that is a convolutional implementation of a fully connected layer which allows a 40 fold speed up. We also describe a 2-phase training procedure that allows us to tackle difficulties related to the imbalance of tumor labels. Finally, we explore a cascade architecture in which the output of a basic CNN is treated as an additional source of information for a subsequent CNN. Results reported on the 2013 BRATS test dataset reveal that our architecture improves over the currently published state-of-the-art while being over 30 times faster.

6.1 Introduction

In the United States alone, it is estimated that 23,000 new cases of brain cancer will be diagnosed in 2015 [1]. While gliomas are the most common brain tumors, they can be less aggressive (i.e. low grade) in a patient with a life expectancy of several years, or more aggressive (i.e. high grade) in a patient with a life expectancy of at most 2 years.

Although surgery is the most common treatment for brain tumors, radiation and chemotherapy may be used to slow the growth of tumors that cannot be physically removed. Magnetic resonance imaging (MRI) provides detailed images of the brain, and is one of the most common tests used to diagnose brain tumors. All the more, brain tumor segmentation from MR images can have great impact for improved diagnostics, growth rate prediction and treatment planning.

While some tumors such as meningiomas can be easily segmented, others like gliomas and glioblastomas are much more difficult to localize. These tumors (together with their surrounding edema) are often diffused, poorly contrasted, and extend tentacle-like structures that make them difficult to segment. Another fundamental difficulty with segmenting brain tumors is that they can appear anywhere in the brain, in almost any shape and size. Furthermore, unlike images derived from X-ray computed tomography (CT) scans, the scale of voxel values in MR images is not standardized. Depending on the type of MR machine used (1.5, 3 or 7 tesla) and the acquisition protocol (field of view value, voxel resolution, gradient strength, b0 value, etc.), the same tumorous cells may end up having drastically different grayscale values when pictured in different hospitals.

Healthy brains are typically made of 3 types of tissues: the white matter, the gray matter, and the cerebrospinal fluid. The goal of brain tumor segmentation is to detect the location and extension of the tumor regions, namely active tumorous tissue (vascularized or not), necrotic tissue, and edema (swelling near the tumor). This is done by identifying abnormal areas when compared to normal tissue. Since glioblastomas are infiltrative tumors, their borders are often fuzzy and hard to distinguish from healthy tissues. As a solution, more than one MRI modality is often employed, e.g. T1 (spin-lattice re-

1. cancer.org

laxation), T1-contrasted (T1C), T2 (spin-spin relaxation), proton density (PD) contrast imaging, diffusion MRI (dMRI), and fluid attenuation inversion recovery (FLAIR) pulse sequences. The contrast between these modalities gives almost a unique signature to each tissue type.

Most automatic brain tumor segmentation methods use hand-designed features [44, 104]. These methods implement a classical machine learning pipeline according to which features are first extracted and then given to a classifier whose training procedure does not affect the nature of those features. An alternative approach for designing task-adapted feature representations is to *learn* a hierarchy of increasingly complex features directly from in-domain data. Deep neural networks have been shown to excel at learning such feature hierarchies [15]. In this work, we apply this approach to learn feature hierarchies adapted specifically to the task of brain tumor segmentation that combine information across MRI modalities.

Specifically, we investigate several choices for training Convolutional Neural Networks (CNNs), which are Deep Neural Networks (DNNs) adapted to image data. We report their advantages, disadvantages and performance using well established metrics. Although CNNs first appeared over two decades ago [90], they have recently become a mainstay of the computer vision community due to their record-shattering performance in the ImageNet Large-Scale Visual Recognition Challenge [86]. While CNNs have also been successfully applied to segmentation problems [3, 96, 61, 24], most of the previous work has focused on non-medical tasks and many involve architectures that are not well suited to medical imagery or brain tumor segmentation in particular. Our preliminary work on using convolutional neural networks for brain tumor segmentation together with two other methods using CNNs was presented in BRATS'14 workshop. However, those results were incomplete and required more investigation (More on this in chapter 6.2).

In this paper, we propose a number of specific CNN architectures for tackling brain tumor segmentation. Our architectures exploit the most recent advances in CNN design and training techniques, such as Maxout [53] hidden units and Dropout [147] regularization. We also investigate several architectures which take into account both the local shape of tumors as well as their context.

One problem with many machine learning methods is that they perform pixel classification without taking into account the local dependencies of labels (i.e. segmentation labels are conditionally independent given the input image). To account for this, one can employ structured output methods such as conditional random fields (CRFs), for which inference can be computationally expensive. Alternatively, one can model label dependencies by considering the pixel-wise probability estimates of an initial CNN as additional input to certain layers of a second DNN, forming a cascaded architecture. Since convolutions are efficient operations, this approach can be significantly faster than implementing a CRF.

We focus our experimental analysis on the fully-annotated MICCAI brain tumor segmentation (BRATS) challenge 2013 dataset [44] using the well defined training and testing splits, thereby allowing us to compare directly and quantitatively to a wide variety of other methods.

Our contributions in this work are four fold:

1. We propose a fully automatic method with results currently ranked second on the BRATS 2013 scoreboard;

2. To segment a brain, our method takes between 25 seconds and 3 minutes, which is one order of magnitude faster than most state-of-the-art methods.

3. Our CNN implements a novel two-pathway architecture that learns about the local details of the brain as well as the larger context. We also propose a two-phase training procedure which we have found is critical to deal with imbalanced label distributions. Details of these contributions are described in Sections 6.3.1 and 6.3.2.

4. We employ a novel cascaded architecture as an efficient and conceptually clean alternative to popular structured output methods. Details on those models are presented in Section 6.3.1.

6.2 Related work

As noted by Menze et al. [104], the number of publications devoted to automated brain tumor segmentation has grown exponentially in the last several decades. This observation not only underlines the need for automatic brain tumor segmentation tools, but also shows that research in that area is still a work in progress.

Brain tumor segmentation methods (especially those devoted to MRI) can be roughly divided in two categories: those based on generative models and those based on discriminative models [104, 12, 4].

Generative models rely heavily on domain-specific prior knowledge about the appearance of both healthy and tumorous tissues. Tissue appearance is challenging to characterize, and existing generative models usually identify a tumor as being a shape or a signal which deviates from a normal (or average) brain [27]. Typically, these methods rely on anatomical models obtained after aligning the 3D MR image on an atlas or a template computed from several healthy brains [38]. A typical generative model of MR brain images can be found in Prastawa et al. [117]. Given the ICBM brain atlas, the method aligns the brain to the atlas and computes posterior probabilities of healthy tissues (white matter, gray matter and cerebrospinal fluid) . Tumorous regions are then found by localizing voxels whose posterior probability is below a certain threshold. A post-processing step is then applied to ensure good spatial regularity. Prastawa et al. [118] also register brain images onto an atlas in order to get a probability map for abnormalities. An active contour is then initialized on this map and iterated until the change in posterior probability is below a certain threshold. Many other active-contour methods along the same lines have been proposed [81, 29, 115], all of which depend on left-right brain symmetry features and/or alignment-based features. Note that since aligning a brain with a large tumor onto a template can be challenging, some methods perform registration and tumor segmentation at the same time [87, 111].

Other approaches for brain tumor segmentation employ discriminative models. Unlike generative modeling approaches, these approaches exploit little prior knowledge on the brain's anatomy and instead rely mostly on the extraction of [a large number of] low level image features, directly modeling the relationship between these features and the

label of a given voxel. These features may be raw input pixels values [63, 60], local histograms [83, 126] texture features such as Gabor filterbanks [153, 154], or alignment-based features such as inter-image gradient, region shape difference, and symmetry analysis [110]. Classical discriminative learning techniques such as SVMs [10, 137, 91] and decision forests [184] have also been used. Results from the 2012, 2013 and 2014 editions of the MICCAI-BRATS Challenge suggest that methods relying on random forests are among the most accurate [104, 56, 83].

One common aspect with discriminative models is their implementation of a conventional machine learning pipeline relying on hand-designed features. For these methods, the classifier is trained to separate healthy from non-heatlthy tissues assuming that the input features have a sufficiently high discriminative power since the behavior the classifier is independent from nature of those features. One difficulty with methods based on hand-designed features is that they often require the computation of a large number of features in order to be accurate when used with many traditional machine learning techniques. This can make them slow to compute and expensive memory-wise. More efficient techniques employ lower numbers of features, using dimensionality reduction or feature selection methods, but the reduction in the number of features is often at the cost of reduced accuracy.

By their nature, many hand-engineered features exploit very generic edge-related information, with no specific adaptation to the domain of brain tumors. Ideally, one would like to have features that are composed and refined into higher-level, task-adapted representations. Recently, preliminary investigations have shown that the use of deep CNNs for brain tumor segmentation makes for a very promising approach (see the BRATS 2014 challenge workshop papers of Davy et al. [36], Zikic et al. [183], Urban et al. [161]). All three methods divide the 3D MR images into 2D [36, 183] or 3D patches [161] and train a CNN to predict its center pixel class. Urban et al. [161] as well as Zikic et al. [183] implemented a fairly common CNN, consisting of a series of convolutional layers, a non-linear activation function between each layer and a softmax output layer. Our work here [2] extends our preliminary results presented in Davy et al.

2. It is important to note that while we did participate in the BRATS 2014 challenge, we could not report complete and fair experiments for it at the time of submitting this manuscript. See Section 6.5 for a discussion on this point.

[36] using a two-pathway architecture, which we use here as a building block.

In computer vision, CNN-based segmentation models have typically been applied to natural scene labeling. For these tasks, the inputs to the model are the RGB channels of a patch from a color image. The work in Pinheiro and Collobert [113] uses a basic CNN to make predictions for each pixel and further improves the predictions by using them as extra information in the input of a second CNN model. Other work [42] involves several distinct CNNs processing the image at different resolutions. The final per-pixel class prediction is made by integrating information learned from all CNNs. To produce a smooth segmentation, these predictions are regularized using a more global superpixel segmentation of the image. Like our work, other recent work has exploited convolution operations in the final layer of a network to extend traditional CNN architectures for semantic scene segmentation [96]. In the medical imaging domain in general there has been comparatively less work using CNNs for segmentation. However, some notable recent work by Huang and Jain [71] has used CNNs to predict the boundaries of neural tissue in electron microscopy images. Here we explore an approach with similarities to the various approaches discussed above, but in the context of brain tumor segmentation.

6.3 Our Convolutional Neural Network Approach

Since the brains in the BRATS dataset lack resolution in the third dimension, we consider performing the segmentation slice by slice from the axial view. Thus, our model processes sequentially each 2D axial image (slice) where each pixel is associated with different image modalities namely; T1, T2, T1C and FLAIR. Like most CNN-based segmentation models [113, 42], our method predicts the class of a pixel by processing the $M \times M$ patch centered on that pixel. The input $\mathbf{X}$ of our CNN model is thus an $M \times M$ 2D patch with several modalities.

The main building block used to construct a CNN architecture is the *convolutional layer*. Several layers can be stacked on top of each other forming a hierarchy of features. Each layer can be understood as extracting features from its preceding layer into the hierarchy to which it is connected. A single convolutional layer takes as input a stack of input planes and produces as output some number of output planes or *feature maps*. Each

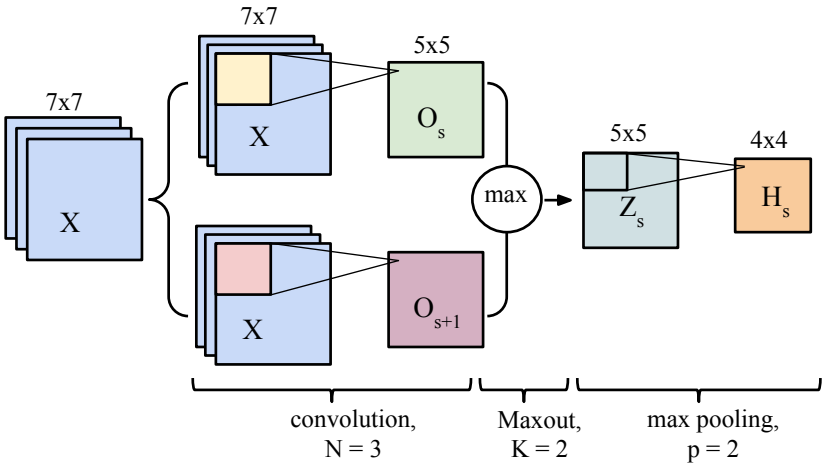


Figure 6.1 – A single convolution layer block showing computations for a single feature map. The input patch (here 7×7), is convolved with series of kernels (here 3×3) followed by Maxout and max-pooling.

feature map can be thought of as a topologically arranged map of responses of a particular spatially local non-linear feature extractor (the parameters of which are learned), applied identically to each spatial neighborhood of the input planes in a sliding window fashion. In the case of a first convolutional layer, the individual input planes correspond to different MRI modalities (in typical computer vision applications, the individual input planes correspond to the red, green and blue color channels). In subsequent layers, the input planes typically consist of the feature maps of the previous layer.

Computing a feature map in a convolutional layer (see Figure 6.1) consists of the following three steps:

1. *Convolution of kernels (filters):* Each feature map $\mathbf{O}_s$ is associated with one kernel (or several, in the case of Maxout). The feature map $\mathbf{O}_s$ is computed as follows:

$$\mathbf{O}_s = b_s + \sum_r \mathbf{W}_{sr} * \mathbf{X}_r \tag{6.1}$$

where $\mathbf{X}_r$ is the r^{th} input channel, $\mathbf{W}_{sr}$ is the sub-kernel for that channel, $*$ is the convolution operation and b_s is a bias term [3]. In other words, the affine operation

3. Since the convolutional layer is associated to R input channels, $\mathbf{X}$ contains $M \times M \times R$ gray-scale values and thus each kernel $\mathbf{W}_s$ contains $N \times N \times R$ weights. Accordingly, the number of parameters

being performed for each feature map is the *sum* of the application of R different 2-dimensional $N \times N$ convolution filters (one per input channel/modality), plus a bias term which is added pixel-wise to each resulting spatial position. Though the input to this operation is a $M \times M \times R$ 3-dimensional tensor, the spatial topology being considered is 2-dimensional in the X-Y axial plane of the original brain volume.

Whereas traditional image feature extraction methods rely on a fixed recipe (sometimes taking the form of convolution with a linear e.g. Gabor filter bank), the key to the success of convolutional neural networks is their ability to learn the weights and biases of individual feature maps, giving rise to data-driven, customized, task-specific dense feature extractors. These parameters are adapted via stochastic gradient descent on a surrogate loss function related to the misclassification error, with gradients computed efficiently via the backpropagation algorithm [134].

Special attention must be paid to the treatment of border pixels by the convolution operation. Throughout our architecture, we employ the so-called *valid-mode* convolution, meaning that the filter response is not computed for pixel positions that are less than $\lfloor N/2 \rfloor$ pixels away from the image border. An $N \times N$ filter convolved with an $M \times M$ input patch will result in a $Q \times Q$ output, where $Q = M - N + 1$. In Figure 6.1, $M = 7$, $N = 3$ and thus $Q = 5$. Note that the size (spatial width and height) of the kernels are hyper-parameters that must be specified by the user.

2. *Non-linear activation function:* To obtain features that are non-linear transformations of the input, an element-wise non-linearity is applied to the result of the kernel convolution. There are multiple choices for this non-linearity, such as the sigmoid, hyperbolic tangent and rectified linear functions [74], [52].

   Recently, Goodfellow et al. [53] proposed a Maxout non-linearity, which has been shown to be particularly effective at modeling useful features. Maxout features are associated with multiple kernels $\mathbf{W}_s$. This implies each Maxout map $\mathbf{Z}_s$ is associated with K feature maps : $\{\mathbf{O}_{Ks}, \mathbf{O}_{Ks+1}, ..., \mathbf{O}_{Ks+K-1}\}$. Note that in Figure 6.1, the Maxout maps are associated with $K = 2$ feature maps. Maxout features correspond to taking the max over the feature maps $\mathbf{O}$, individually for

in a convolutional block of consisting of S feature maps is equal to $R \times M \times M \times S$.

each spatial position:

$$Z_{s,i,j} = \max \{O_{Ks,i,j}, O_{Ks+1,i,j}, ..., O_{Ks+K-1,i,j}\} \tag{6.2}$$

where i, j are spatial positions. Maxout features are thus equivalent to using a convex activation function, but whose shape is adaptive and depends on the values taken by the kernels.

3. *Max pooling:* This operation consists of taking the maximum feature (neuron) value over sub-windows within each feature map. This can be formalized as follows:

$$H_{s,i,j} = \max_{p} Z_{s,Si+p,Sj+p}, \tag{6.3}$$

where p determines the max pooling window size and S is the stride hyper-parameter, which corresponds to the horizontal and vertical increments at which pooling sub-windows are positioned. The sub-windows can be overlapping or not (Figure 6.1 shows an overlapping configuration). The max-pooling operation shrinks the size of the feature map. This is controlled by the pooling size p and S. Let $Q \times Q$ be the shape of the feature map before max-pooling. The output of the max-pooling operation would be of size $D \times D$, where $D = (Q - p)/S + 1$. In Figure 6.1, since $Q = 5, p = 2, S = 1$, the max-pooling operation results into a $D = 4$ output feature map. The motivation for this operation is to introduce invariance to local translations. This subsampling procedure has been found beneficial in other applications [86].

Convolutional networks have the ability to extract a hierarchy of increasingly complex features which makes them very appealing. This is done by treating the output feature maps of a convolutional layer as input channels to the subsequent convolutional layer.

From the neural network perspective, feature maps correspond to a layer of hidden units or neurons. Specifically, each coordinate within a feature map corresponds to an individual neuron, for which the size of its receptive field corresponds to the kernel's size. A kernel's value also represents the weights of the connections between the layer's neurons and the neurons in the previous layer. It is often found in practice that the learned kernels resemble edge detectors, each kernel being tuned to a different spatial frequency, scale and orientation, as is appropriate for the statistics of the training data.

Finally, to perform a prediction of the segmentation labels, we connect the last convolutional hidden layer to a convolutional output layer followed by a non-linearity (i.e. no pooling is performed). It is necessary to note that, for segmentation purposes, a conventional CNN will not yield an efficient test time since the output layer is typically fully connected. By using a convolution at the end, for which we have an efficient implementation, the prediction at test time for a whole brain will be 45 times faster. The convolution uses as many kernels as there are different segmentation labels (in our case five). Each kernel thus acts as the ultimate detector of tissue from one of the segmentation labels. We use the *softmax* non-linearity which normalizes the result of the kernel convolutions into a multinominal distribution over the labels. Specifically, let $\mathbf{a}$ be the vector of values at a given spatial position, it computes $\mathrm{softmax}(\mathbf{a}) = \exp(\mathbf{a})/Z$ where $Z = \sum_c \exp(a_c)$ is a normalization constant. More details will be discussed in Section 6.4.

Noting $\mathbf{Y}$ as the segmentation label field over the input patch $\mathbf{X}$, we can thus interpret each spatial position of the convolutional output layer as providing a model for the likelihood distribution $p(Y_{ij}|\mathbf{X})$, where Y_{ij} is the label at position i, j. We get the probability of all labels simply by taking the product of each conditional $p(\mathbf{Y}|\mathbf{X}) = \prod_{ij} p(Y_{ij}|\mathbf{X})$. Our approach thus performs a multiclass labeling by assigning to each pixel the label with the largest probability.

6.3.1 The Architectures

Our description of CNNs so far suggests a simple architecture corresponding to a single stack of several convolutional layers. This configuration is the most commonly implemented architecture in the computer vision literature. However, one could imagine other architectures that might be more appropriate for the task at hand.

In this work, we explore a variety of architectures by using the concatenation of feature maps from different layers as another operation when composing CNNs. This operation allows us to construct architectures with multiple computational paths, which can each serve a different purpose. We now describe the two types of architectures that we explore in this work.

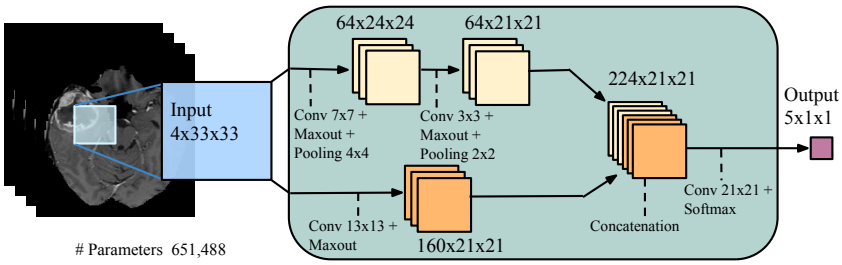


Figure 6.2 – Two-pathway CNN architecture (TWOPATHCNN). The figure shows the input patch going through two paths of convolutional operations. The feature-maps in the local and global paths are shown in yellow and orange respectively. The convolutional layers used to produce these feature-maps are indicated by dashed lines in the figure. The green box embodies the whole model which in later architectures will be used to indicate the TWOPATHCNN.

Two-pathway architecture

This architecture is made of two streams: a pathway with smaller 7×7 receptive fields and another with larger 13×13 receptive fields. We refer to these streams as the local pathway and the global pathway, respectively. The motivation for this architectural choice is that we would like the prediction of the label of a pixel to be influenced by two aspects: the visual details of the region around that pixel and its larger "context", i.e. roughly where the patch is in the brain.

The full architecture along with its details is illustrated in Figure 6.2. We refer to this architecture as the TWOPATHCNN. To allow for the concatenation of the top hidden layers of both pathways, we use two layers for the local pathway, with 3×3 kernels for the second layer. While this implies that the effective receptive field of features in the top layer of each pathway is the same, the global pathway's parametrization more directly and flexibly models features in that same area. The concatenation of the feature maps of both pathways is then fed to the output layer.

Cascaded architectures

One disadvantage of the CNNs described so far is that they predict each segmentation label separately from each other. This is unlike a large number of segmentation methods

in the literature, which often propose a joint model of the segmentation labels, effectively modeling the direct dependencies between spatially close labels. One approach is to define a conditional random field (CRF) over the labels and perform mean-field message passing inference to produce a complete segmentation. In this case, the final label at a given position is effectively influenced by the models beliefs about what the label is in the vicinity of that position.

On the other hand, inference in such joint segmentation methods is typically more computationally expensive than a simple feed-forward pass through a CNN. This is an important aspect that one should take into account if automatic brain tumor segmentation is to be used in a day-to-day practice.

Here, we describe CNN architectures that both exploit the efficiency of CNNs, while also more directly model the dependencies between adjacent labels in the segmentation. The idea is simple: since we'd like the ultimate prediction to be influenced by the model's beliefs about the value of nearby labels, we propose to feed the output probabilities of a first CNN as additional inputs to the layers of a second CNN. Again, we do this by relying on the concatenation of convolutional layers. In this case, we simply concatenate the output layer of the first CNN with any of the layers in the second CNN. Moreover, we use the same two-pathway structure for both CNNs. This effectively corresponds to a cascade of two CNNs, thus we refer to such models as cascaded architectures.

In this work, we investigated three cascaded architectures that concatenate the first CNN's output at different levels of the second CNN:

— *Input concatenation:* In this architecture, we provide the first CNN's output directly as input to the second CNN. They are thus simply treated as additional image channels of the input patch. The details are illustrated in Figure 6.3a. We refer to this model as INPUTCASCADECNN.
— *Local pathway concatenation:* In this architecture, we move up one layer in the local pathway and perform concatenation to its first hidden layer, in the second CNN. The details are illustrated in Figure 6.3b. We refer to this model as LOCALCASCADECNN.
— *Pre-output concatenation:* In this last architecture, we move to the very end of

the second CNN and perform concatenation right before its output layer. This architecture is interesting, as it is similar to the computations made by one pass of mean-field inference [173] in a CRF whose pairwise potential functions are the weights in the output kernels. From this view, the output of the first CNN is the first iteration of mean-field, while the output of the second CNN would be the second iteration. The difference with regular mean-field however is that our CNN allows the output at one position to be influenced by its previous value, and the convolutional kernels are not the same in the first and second CNN. The details are illustrated in Figure 6.3c. We refer to this model as MFCASCADECNN.

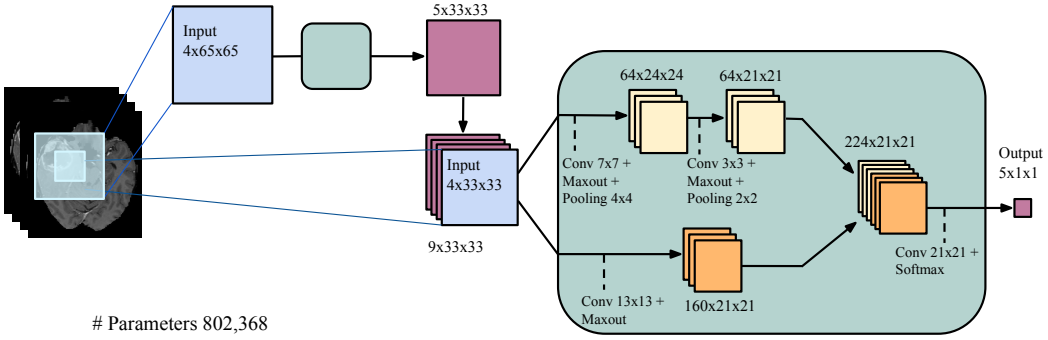
6.3.2 Training

Gradient Descent By interpreting the output of the convolutional network as a model for the distribution over segmentation labels, a natural training criteria is to maximize the probability of all labels in our training set or, equivalently, to minimize the negative log-probability $-\log p(\mathbf{Y}|\mathbf{X}) = \sum_{ij} -\log p(Y_{ij}|\mathbf{X})$ for each labeled brain.

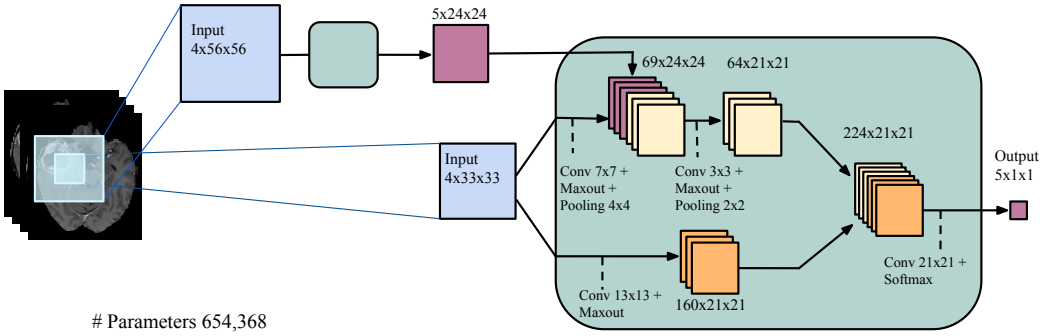
To do this, we follow a stochastic gradient descent approach by repeatedly selecting labels Y_{ij} at a random subset of patches within each brain, computing the average negative log-probabilities for this mini-batch of patches and performing a gradient descent step on the CNNs parameters (i.e. the kernels at all layers).

Performing updates based only on a small subset of patches allows us to avoid having to process a whole brain for each update, while providing reliable enough updates for learning. In practice, we implement this approach by creating a dataset of mini-batches of smaller brain image patches, paired with the corresponding center segmentation label as the target.

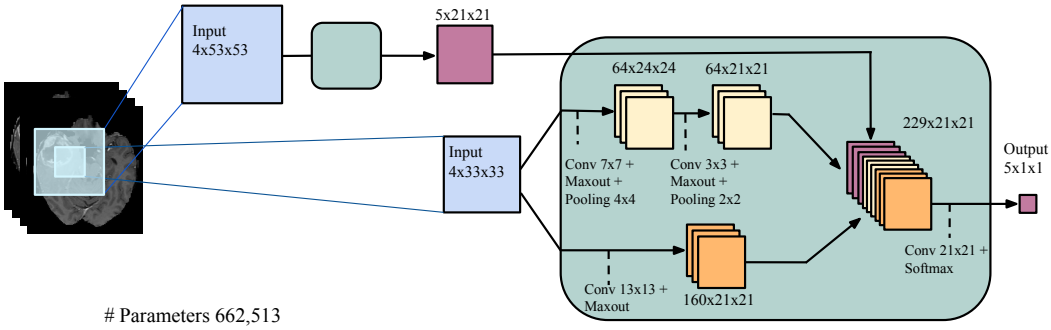
To further improve optimization, we implemented a so-called momentum strategy which has been shown successful in the past [86]. The idea of momentum is to use a temporally averaged gradient in order to damp the optimization velocity:



(a) Cascaded architecture, using input concatenation (INPUTCASCADECNN).



(b) Cascaded architecture, using local pathway concatenation (LOCALCASCADECNN).



(c) Cascaded architecture, using pre-output concatenation, which is an architecture with properties similar to that of learning using a limited number of mean-field inference iterations in a CRF (MFCASCADECNN).

Figure 6.3 – Cascaded architectures.

$$\begin{aligned} \mathbf{V}_{i+1} &= \mu * \mathbf{V}_i - \alpha * \nabla \mathbf{W}_i \\ \mathbf{W}_{i+1} &= \mathbf{W}_i + \mathbf{V}_{i+1} \end{aligned}$$

where $\mathbf{W}_i$ stands for the CNNs parameters at iteration i, $\nabla \mathbf{W}_i$ the gradient of the loss function at $\mathbf{W}_i$, $\mathbf{V}$ is the integrated velocity initialized at zero, α is the learning rate, and μ the momentum coefficient. We define a schedule for the momentum μ where the momentum coefficient is gradually increased during training. In our experiments the initial momentum coefficient was set to $\mu = 0.5$ and the final value was set to $\mu = 0.9$.

Also, the learning rate α is decreased by a factor at every epoch. The initial learning rate was set to $\alpha = 0.005$ and the decay factor to 10^{-1}.

Two-phase training Brain tumor segmentation is a highly data imbalanced problem where the healthy voxels (i.e. label 0) comprise 98% of total voxels. From the remaining 2% pathological voxels, 0.18% belongs to necrosis (label 1), 1.1% to edema (label 2), 0.12% to non-enhanced (label 3) and 0.38% to enhanced tumor (label 4). Selecting patches from the true distribution would cause the model to be overwhelmed by healthy patches and causing problem when training out CNN models. Instead, we initially construct our patches dataset such that all labels are equiprobable. This is what we call the first training phase. Then, in a second phase, we account for the un-balanced nature of the data and re-train only the output layer (i.e. keeping the kernels of all other layers fixed) with a more representative distribution of the labels. This way we get the best of both worlds: most of the capacity (the lower layers) is used in a balanced way to account for the diversity in all of the classes, while the output probabilities are calibrated correctly (thanks to the re-training of the output layer with the natural frequencies of classes in the data).

Regularization Successful CNNs tend to be models with a lot of capacity, making them vulnerable to overfitting in a setting like ours where there clearly are not enough training examples. Accordingly, we found that regularization is important in obtaining

good results. Here, regularization took several forms. First, in all layers, we bounded the absolute value of the kernel weights and applied both L1 and L2 regularization to prevent overfitting. This is done by adding the regularization terms to the negative log-probability (i.e. $-\log p(\mathbf{Y}|\mathbf{X}) + \lambda_1|\mathbf{W}|_1 + \lambda_2|\mathbf{W}|^2$, where λ_1 and λ_2 are coefficients for L1 and L2 regularization terms respectively). L1 and L2 affect the parameters of the model in different ways, while L1 encourages sparsity, L2 encourages small values. We also used a validation set for early stopping, i.e. stop training when the validation performance stopped improving. The validation set was also used to tune the other hyper-parameters of the model. The reader shall note that the hyper-parameters of the model which includes using or not L2 and/or L1 coefficients were selected by doing a grid search over range of parameters. The chosen hyper-parameters were the ones for which the model performed best on a validation set.

Moreover, we used *Dropout* [147], a recent regularization method that works by stochastically adding noise in the computation of the hidden layers of the CNN. This is done by multiplying each hidden or input unit by 0 (i.e. masking) with a certain probability (e.g. 0.5), independently for each unit and training update. This encourages the neural network to learn features that are useful “on their own", since each unit cannot assume that other units in the same layer won’t be masked as well and co-adapt its behavior. At test time, units are instead multiplied by one minus the probability of being masked. For more details, see Srivastava et al. [147].

Considering the large number of parameters our model has, one might think that even with our regularization strategy, the 30 training brains from BRATS 2013 are too few to prevent overfitting. But as will be shown in the results section, our model generalizes well and thus do not overfit. One reason for this is the fact that each brain comes with 200 2d slices and thus, our model has approximately 6000 2D images to train on. We shall also mention that by their very nature, MRI images of brains are very similar from one patient to another. Since the variety of those images is much lower than those in real-image datasets such as CIFAR and ImageNet, a fewer number of training samples is thus needed.

Cascaded Architectures To train a cascaded architecture, we start by training the TWOPATHCNN with the two phase stochastic gradient descent procedure described

previously. Then, we fix the parameters of the TWOPATHCNN and include it in the cascaded architecture (be it the INPUTCASCADECNN, the LOCALCASCADECNN, or the MFCASCADECNN) and move to training the remaining parameters using a similar procedure. It should be noticed however that for the spatial size of the first CNN's output and the layer of the second CNN to match, we must feed to the first CNN a much larger input. Thus, training of the second CNN must be performed on larger patches. For example in the INPUTCASCADECNN (Figure 6.3a), the input size to the first model is of size 65×65 which results into an output of size 33×33. Only in this case the outputs of the first CNN can be concatenated with the input channels of the second CNN.

6.4 Implementation details

Our implementation is based on the Pylearn2 library [55]. Pylearn2 is an open-source machine learning library specializing in deep learning algorithms. It also supports the use of GPUs, which can greatly accelerate the execution of deep learning algorithms.

Since CNN's are able to learn useful features from scratch, we applied only minimal pre-processing. We employed the same pre-processing as Tustison et al., the winner of the 2013 BRATS challenge [104]. The pre-processing follows three steps. First, the 1% highest and lowest intensities are removed. Then, we apply an N4ITK bias correction [7] to T1 and T1C modalities. The data is then normalized within each input channel by subtracting the channel's mean and dividing by the channel's standard deviation.

As for post-processing, a simple method based on connected components was implemented to remove flat blobs which might appear in the predictions due to bright corners of the brains close to the skull.

The hyper-parameters of the different architectures (kernel and max pooling size for each layer and the number of layers) can be seen in Figure 6.3. Hyper-parameters were tuned using grid search and cross-validation on a validation set (see Bengio [13]). The chosen hyper-parameters were the ones for which the model performed best on the validation set. For max pooling, we always use a stride of 1. This is to keep per-pixel accuracy during full image prediction. We observed in practice that max pooling

in the global path does not improve accuracy. We also found that adding additional layers to the architectures or increasing the capacity of the model by adding additional feature maps to the convolutional blocks do not provide any meaningful performance improvement.

Biases are initialized to zero except for the softmax layer for which we initialized them to the $\log$ of the label frequencies. The kernels are randomly initialized from $U\left(-0.005, 0.005\right)$. Training takes about 3 minutes per epoch for the TWOPATHCNN model on an NVIDIA Titan black card.

At test time, we run our code on a GPU in order to exploit its computational speed. Moreover, the convolutional nature of the output layer allows us to further accelerate computations at test time. This is done by feeding as input a full image and not individual patches. Therefore, convolutions at all layers can be extended to obtain all label probabilities $p(Y_{ij}|\mathbf{X})$ for the entire image. With this implementation, we are able to produce a segmentation in 25 seconds per brain on the Titan black card with the TWOPATHCNN model. This turns out to be 45 times faster than when we extracted a patch at each pixel and processed them individually for the entire brain.

Predictions for the MFCASCADECNN model, the LOCALCASCADECNN model, and INPUTCASCADECNN model take on average 1.5 minutes, 1.7 minutes and 3 minutes respectively.

6.5 Experiments and Results

The experiments were carried out on real patient data obtained from the 2013 brain tumor segmentation challenge (BRATS2013), as part of the MICCAI conference [44]. The BRATS2013 dataset is comprised of 3 sub-datasets. The training dataset, which contains 30 patient subjects all with pixel-accurate ground truth (20 high grade and 10 low grade tumors); the test dataset which contains 10 (all high grade tumors) and the leaderboard dataset which contains 25 patient subjects (21 high grade and 4 low grade tumors). There is no ground truth provided for the test and leaderboard datasets. All brains in the dataset have the same orientation. For each brain there exists 4 modalities,

namely T1, T1C, T2 and Flair which are co-registered. The training brains come with groundtruth for which 5 segmentation labels are provided, namely *non-tumor*, *necrosis*, *edema*, *non-enhancing tumor* and *enhancing tumor*. Figure 6.4 shows an example of the data as well as the ground truth. In total, the model iterates over about 2.2 million examples of tumorous patches (this consists of all the 4 sub-tumor classes) and goes through 3.2 million of the healthy patches. As mentioned before during the first phase training, the distribution of examples introduced to the model from all 5 classes is uniform.

Please note that we could not use the BRATS 2014 dataset due to problems with both the system performing the evaluation and the quality of the labeled data. For these reasons the old BRATS 2014 dataset has been removed from the official website and, at the time of submitting this manuscript, the BRATS website still showed: "Final data for BRATS 2014 to be released soon". Furthermore, we have even conducted an experiment where we trained our model with the old 2014 dataset and made predictions on the 2013 test dataset; however, the performance was worse than our results mentioned in this paper. For these reasons, we decided to focus on the BRATS 2013 data.

As mentioned in Section 6.3, we work with 2D slices due to the fact that the MRI volumes in the dataset do not posses an isotropic resolution and the spacing in the third dimension is not consistent across the data. We explored the use of 3D information (by treating the third dimension as extra input channels or by having an architecture which takes orthogonal slices from each view and makes the prediction on the intersecting center pixel), but that didn't improve performance and made our method very slow.

Note that as suggested by Krizhevsky et al. [86], we applied data augmentation by flipping the input images. Unlike what was reported by Zeiler and Fergus [178], it did not improve the overall accuracy of our model.

Quantitative evaluation of the models performance on the test set is achieved by uploading the segmentation results to the online BRATS evaluation system [43]. The online system provides the quantitative results as follows: The tumor structures are grouped in 3 different tumor regions. This is mainly due to practical clinical applications. As described by Menze et al. [104], tumor regions are defined as:

a) The *complete* tumor region (including all four tumor structures).

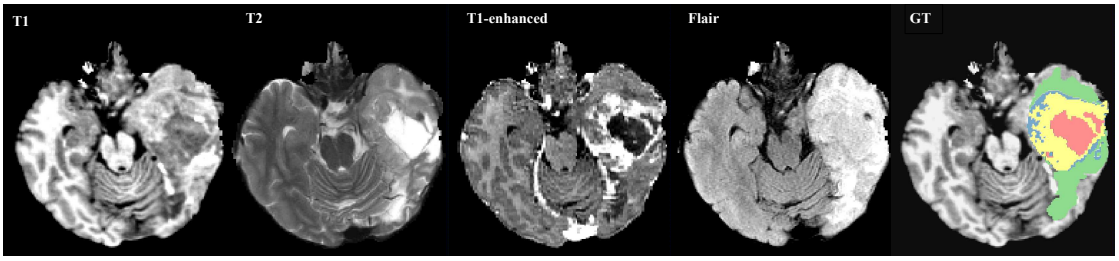


Figure 6.4 – The first four images from left to right show the MRI modalities used as input channels to various CNN models and the fifth image shows the ground truth labels where ■ edema, ■ enhanced tumor, ■ necrosis, ■ non-enhanced tumor.

b) The *core* tumor region (including all tumor structures exept “edema").

c) The *enhancing* tumor region (including the “enhanced tumor" structure).

For each tumor region, *Dice* (identical to F measure), *Sensitivity* and *Specificity* are computed as follows :

$$\begin{aligned} Dice(P,T) &= \frac{|P_1 \wedge T_1|}{(|P_1| + |T_1|)/2}, \\ Sensitivity(P,T) &= \frac{|P_1 \wedge T_1|}{|T_1|}, \\ Specificity(P,T) &= \frac{|P_0 \wedge T_0|}{|T_0|}, \end{aligned}$$

where P represents the model predictions and T represents the ground truth labels. We also note as T_1 and T_0 the subset of voxels predicted as positives and negatives for the tumor region in question. Similarly for P_1 and P_0. The online evaluation system also provides a ranking for every method submitted for evaluation. This includes methods from the 2013 BRATS challenge published in [104] as well as anonymized unpublished methods for which no reference is available. In this section, we report experimental results for our different CNN architectures.

6.5.1 The TWOPATHCNN architecture

As mentioned previously, unlike conventional CNNs, the TWOPATHCNN architecture has two pathways: a “local" path focusing on details and a “global" path more focused on the context. To better understand how joint training of the global and local pathways benefits the performance, we report results on each pathway as well as results on averaging the outputs of each pathway when trained separately. Our method also deals with the unbalanced nature of the problem by training in two phases as discussed in Section 6.3.2. To see the impact of the two phase training, we report results with and without it. We refer to the CNN model consisting of only the local path (i.e. conventional CNN architecture) as LOCALPATHCNN, the CNN model consisting of only the global path as GLOBALPATHCNN, the model averaging the outputs of the local and global paths (i.e. LOCALPATHCNN and GLOBALPATHCNN) as AVERAGECNN and the two-pathway CNN architecture as TWOPATHCNN. The second training phase is noted by appending ‘*’ to the architecture name. Since the second phase training has a substantial effect and always improves the performance, we only report results on GLOBALPATHCNN and AVERAGECNN with the second phase.

Table 6.1 presents the quantitative results of these variations. This table contains results for the TWOPATHCNN with one and two training phases, the common single path CNN (i.e. LOCALPATHCNN) with one and two training phases, the GLOBALPATHCNN* which is a single path CNN model following the global pathway architecture and the output average of each of the trained single-pathway models (AVERAGECNN*). Without much surprise, the single path with one training phase CNN was ranked last with the lowest scores on almost every region. Using a second training phase gave a significant boost to that model with a rank that went from 15 to 9. Also, the table shows that joint training of the local and global paths yields better performance compared to when each pathway is trained separately and the outputs are averaged. One likely explanation is that by joint training the local and global paths, the model allows the two pathways to co-adapt. In fact, the AVERAGECNN* performs worse than the LOCALPATHCNN* due to the fact that the GLOBALPATHCNN* performs very badly. The top performing method in the uncascaded models is the TWOPATHCNN* with a rank of 4.

Also, in some cases results are less accurate over the Enhancing region than for the Core

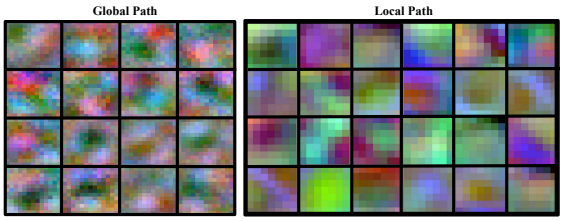


Figure 6.5 – Randomly selected filters from the first layer of the model. From left to right the figure shows visualization of features from the first layer of the global and local path respectively. Features in the local path include more edge detectors while the global path contains more localized features.

and Complete regions. There are 2 main reasons for that. First, borders are usually diffused and there are no clear cut between enhanced tumor and non-enhanced tissues. This creates problems for both user labeling, ground truth, as well as the model. The second reason is that the model learns what it sees in the ground truth. Since the labels are created by different people and since the borders are not clear, each user has a slightly different interpretation of the borders of the enhanced tumor and so sometimes we see overly thick enhanced tumor in the ground truth.

Figure 6.5 shows representation of low level features in both local and global paths. As seen from this figure, features in the local path include more edge detectors while the ones in the global path are more localized features. Unfortunately, visualizing the learned mid/high level features of a CNN is still very much an open research problem. However, we can study the impact these features have on predictions by visualizing the segmentation results of different models. The segmentation results on two subjects from our validation set, produced by different variations of the basic model can be viewed in Figure 6.7 [4]. As shown in the figure, the two-phase training procedure allows the model to learn from a more realistic distribution of labels and thus removes false positives produced by the model which trains with one training phase. Moreover, by having two pathways, the model can simultaneously learn the global contextual features as well as the local detailed features. This gives the advantage of correcting labels at a global scale as well as recognizing fine details of the tumor at a local scale, yielding a better segmentation as oppose to a single path architecture which results in smoother

4. It is important to note that we do not train the model on the validation set and thus the quality of the results is not due to overfitting

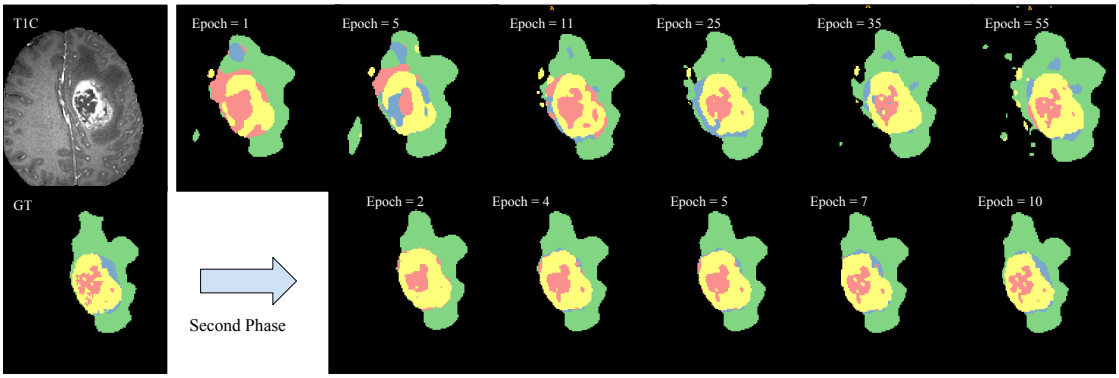


Figure 6.6 – Progression of learning in INPUTCASCADECNN*. The stream of figures on the first row from left to right show the learning process during the first phase. As the model learns better features, it can better distinguish boundaries between tumor sub-classes. This is made possible due to uniform label distribution of patches during the first phase training which makes the model believe all classes are equiprobable and causes some false positives. This drawback is alleviated by training a second phase (shown in second row from left to right) on a distribution closer to the true distribution of labels. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

boundaries. Joint training of the two convolutional pathways and having two training phases achieves better results.

6.5.2 Cascaded architectures

We now discuss our experiments with the three cascaded architectures namely INPUTCASCADECNN, LOCALCASCADECNN and MFCASCADECNN. Table 6.2 provides the quantitative results for each architecture. Figure 6.7 also provides visual examples of the segmentation generated by each architecture.

We find that the MFCASCADECNN* model yields smoother boundaries between classes. We hypothesize that, since the neurons in the softmax output layer are directly connected to the previous outputs within each receptive field, these parameters are more likely to learn that the center pixel label should have a similar label to its surroundings.

As for the LOCALCASCADECNN* architecture, while it resulted in fewer false positives in the complete tumor category, the performance in other categories (i.e. tumor

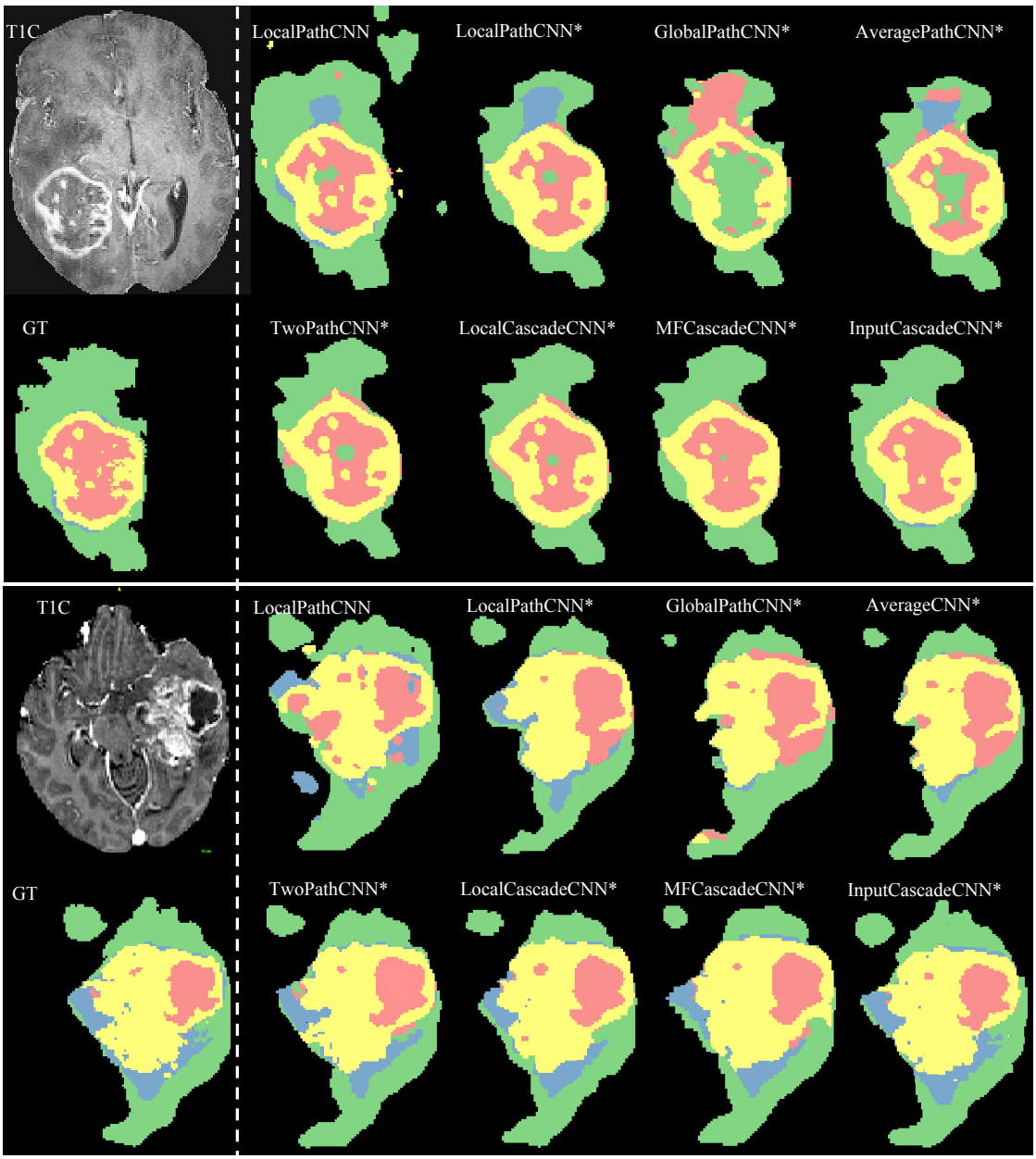


Figure 6.7 – Visual results from our CNN architectures from the Axial view. For each sub-figure, the top row from left to right shows T1C modality, the conventional one path CNN, the Conventional CNN with two training phases, and the TWOPATHCNN model. The second row from left to right shows the ground truth, LOCALCASCADECNN model, the MFCASCADECNN model and the INPUTCASCADECNN. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

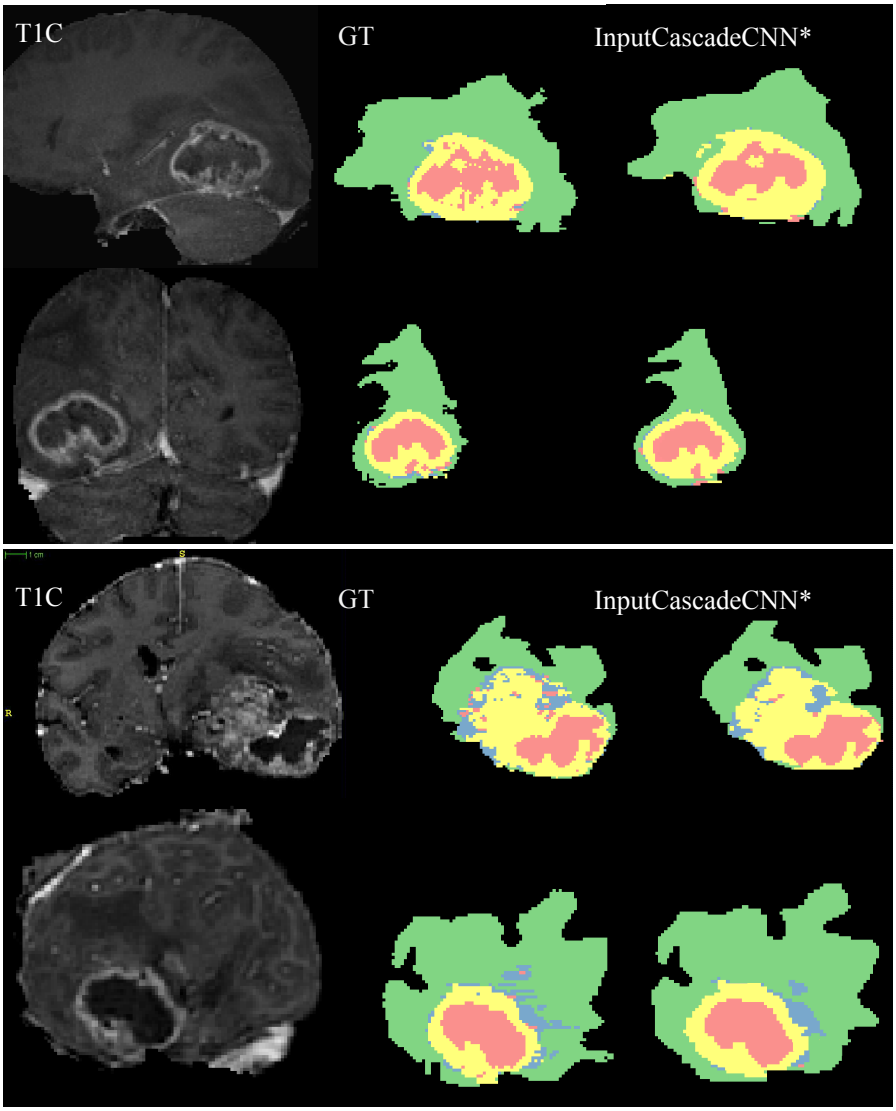


Figure 6.8 – Visual results from our top performing model, INPUTCASCADECNN* on Coronal and Sagittal views. The subjects are the same as in Figure 6.7. In every sub-figure, the top row represents the Sagital view and the bottom row represents the Coronal view. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

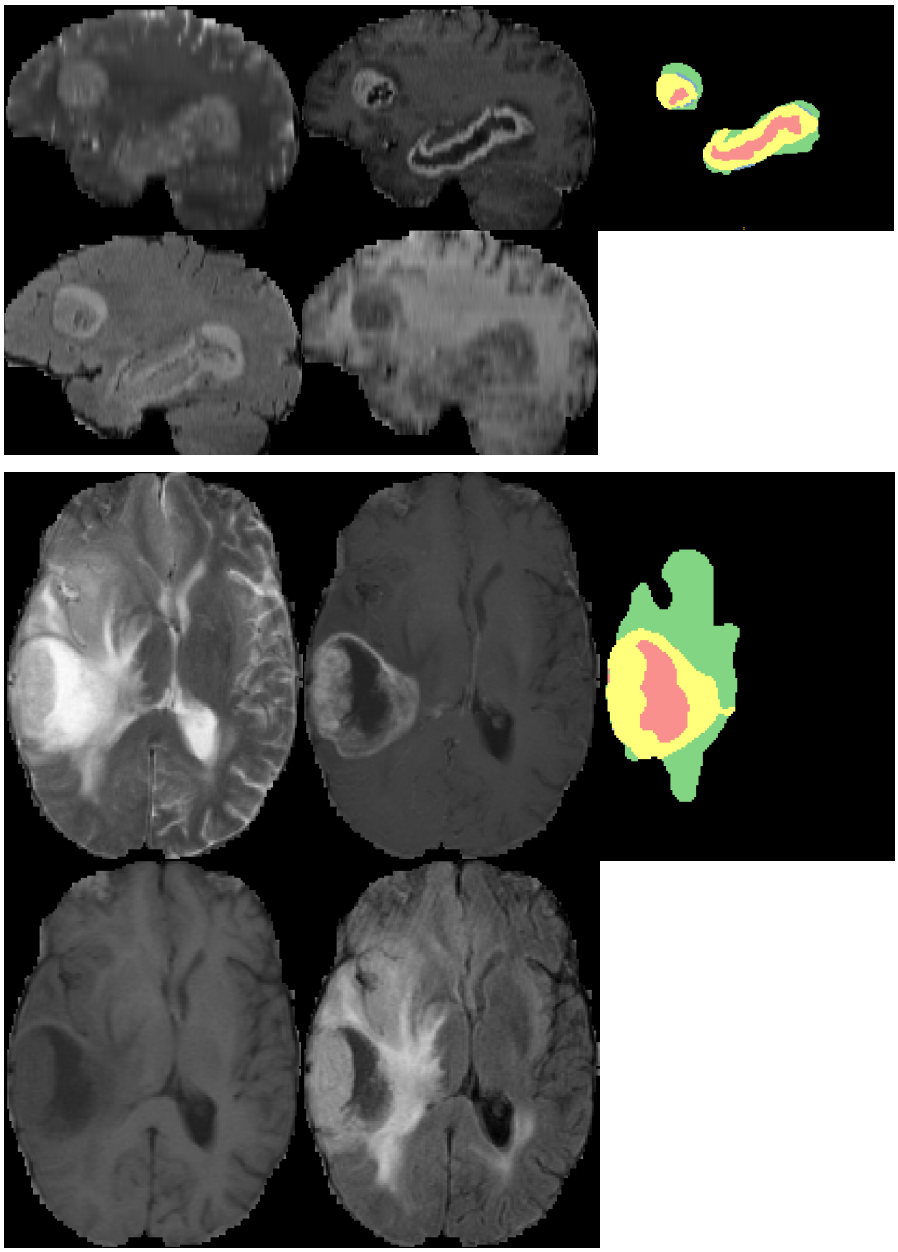


Figure 6.9 – Visual segmentation results from our top performing model, INPUTCASCADECNN*, on examples of the BRATS2013 test dataset in Saggital (top) and Axial (bottom) views. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

Table 6.1 – Performance of the TWOPATHCNN model and variations. The second phase training is noted by appending '*' to the architecture name. The 'Rank' column represents the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
4	TWOPATHCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
9	LOCALPATHCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
10	AVERAGECNN*	0.84	0.75	0.70	0.95	0.83	0.73	0.77	0.74	0.73
14	GLOBALPATHCNN*	0.82	0.73	0.68	0.93	0.81	0.70	0.75	0.65	0.70
14	TWOPATHCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
15	LOCALPATHCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80

Table 6.2 – Performance of the cascaded architectures. The reported results are from the second phase training. The 'Rank' column shows the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
2	INPUTCASCADECNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
4-a	MFCASCADECNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
4-c	LOCALCASCADECNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75

core and enhanced tumor) did not improve.

Figure 6.8 shows segmentation results from the same brains (as in Figure 6.7) in Sagittal and Coronal views. The INPUTCASCADECNN* model was used to produce these results. As seen from this figure, although the segmentation is performed on Axial view but the output is consistent in Coronal and Sagittal views. Although subjects in Figure 5 and Figure 6 are from our validation set for which the model is not trained on and the segmentation results from these subjects can give a good estimate of the models performance on a test set, however, for further clarity we visualise the models performance on two subjects from BRATS-2013 testst. These results are shown in Figure 6.9 in Saggital (top) and Axial (bottom) views.

To better understand the process for which INPUTCASCADECNN* learns features, we present in Figure 6.6 the progression of the model by making predictions at every few epochs on a subject from our validation set.

Overall, the best performance is reached by the INPUTCASCADECNN* model. It improves the Dice measure on all tumor regions. With this architecture, we were able to reach the second rank on the BRATS 2013 scoreboard. While MFCASCADECNN*, TWOPATHCNN* and LOCALCASCADECNN* are all ranked 4, the inner ranking be-

tween these three models is noted as 4a, 4b and 4c respectively.

Table 6.3 shows how our implemented architectures compare with currently published state-of-the-art methods as mentioned in [104] [5]. The table shows that INPUTCASCADECNN* out performs Tustison et al. the winner of the BRATS 2013 challenge and is ranked first in the table. Results from the BRATS-2013 leaderboard presented in Table 6.4 shows that our method outperforms other approaches on this dataset. We also compare our top performing method in Table 6.5 with state-of-the-art methods on BRATS-2012, "4 label" test set as mentioned in [104]. As seen from this table, our method out performs other methods in the tumor Core category and gets competitive results on other categories.

Let us mention that Tustison's method takes 100 minutes to compute predictions per brain as reported in [104], while the INPUTCASCADECNN* takes 3 minutes, thanks to the fully convolutional architecture and the GPU implementation, which is over 30 times faster than the winner of the challenge. The TWOPATHCNN* has a performance close to the state-of-the-art. However, with a prediction time of 25 seconds, it is over 200 times faster than Tustison's method. Other top methods in the table are that of Meier et al and Reza et al with processing times of 6 and 90 minutes respectively. Recently Subbanna et al. [154] published competitive results on the BRATS 2013 dataset, reporting dice measures of $0.86, 0.86, 0.77$ for Complete, Core and Enhancing tumor regions. Since they do not report Specificity and Sensitivity measures, a completely fair comparison with that method is not possible. However, as mentioned in [154], their method takes 70 minutes to process a subject, which is about 23 times slower than our method.

Regarding other methods using CNNs, Urban et al. [161] used an average of two 3D convolutional networks with dice measures of $0.87, 0.77, 0.73$ for Complete, Core and Enhancing tumor regions on BRATS 2013 test dataset with a prediction time of about 1 minute per model which makes for a total of 2 minutes. Again, since they do not report Specificity and Sensitivity measures, we can not make a full comparison. However,

5. Please note that the results mentioned in Table 6.3 and Table 6.4 are from methods competing in the BRATS 2013 challenge for which a static table is provided [https://www.virtualskeleton.ch/BRATS/StaticResults2013]. Since then, other methods have been added to the score board but for which no reference is available.

based on their dice scores our TWOPATHCNN* is similar in performance while taking only 25 seconds, which is four times faster. And the INPUTCASCADECNN* is better or equal in accuracy while having the same processing time. As for [183], they do not report results on BRATS 2013 test dataset. However, their method is very similar to the LOCALPATHCNN which, according to our experiments, has worse performance.

Using our best performing method, we took part in the BRATS 2015 challenge. The BRATS 2015 training dataset comprises of 220 subjects with high grade and 54 subjects with low grade gliomas. There are 53 subjects with mixed high and low grade gliomas for testing. Every participating group had 48 hours from receiving the test subjects to process them and submit their segmentation results to the online evaluation system. BRATS'15 contains the training data of 2013. The ground truth for the rest of the training brains is generated by a voted average of segmented results of the top performing methods in BRATS'13 and BRATS'12. Some of these automatically generated ground truths have been refined manually by a user.

Because distribution of the intensity values in this dataset is very variable from one subject to another, we used a 7 fold cross validation for training. At test time, a voted average of these models was made to make prediction for each subject in the test dataset. The results of the challenge are presented in Figure 6.10. The semi-automatic methods participating in the challenge have been highlighted in grey. Please note since these results are not yet publicly available, we refrain from disclosing the name of the participants. In this figure the semi-automatic methods are highlighted in gray. As seen from the figure, our method ranks either first or second on Complete tumor and tumor Core categories and gets competitive results on active tumor category. Our method has also less outliers than most other approaches.

6.6 Conclusion

In this paper, we presented an automatic brain tumor segmentation method based on deep convolutional neural networks. We considered different architectures and investigated their impact on the performance. Results from the BRATS 2013 online evaluation system confirms that with our best model we managed to improve on the currently

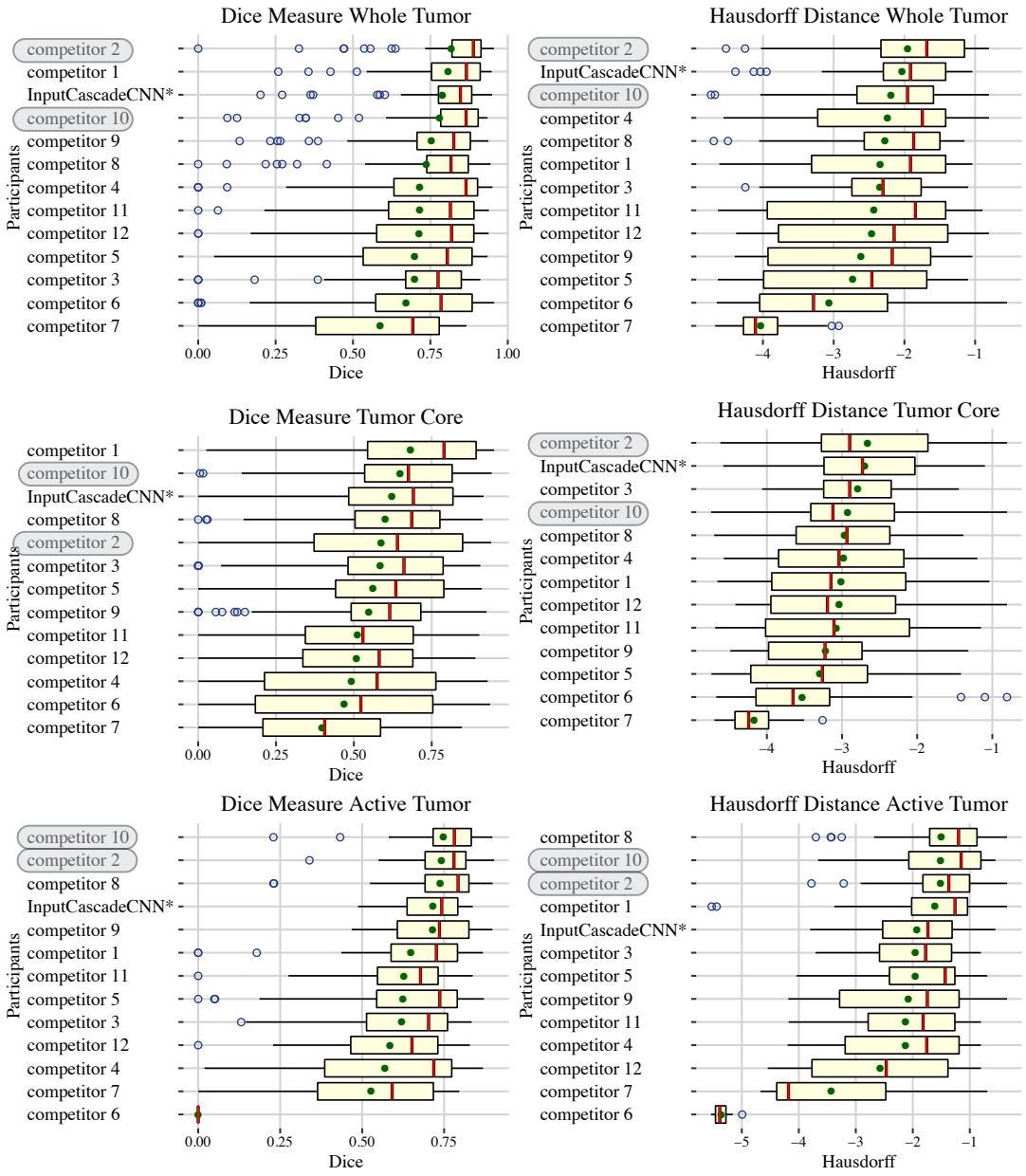


Figure 6.10 – Our BRATS'15 challenge results using INPUTCASCADECNN*. Dice scores and negative log Hausdorff distances are presented for the three tumor categories. Since the results of the challenge are not yet publicly available, we are unable to disclose the name of the participants. The semi-automatic methods are highlighted in gray. In each sub-figure, the methods are ranked based on the mean value. The mean is presented in green, the median in red and outliers in blue.

Table 6.3 – Comparison of our implemented architectures with the state-of-the-art methods on the BRATS-2013 test set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
INPUTCASCADECNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
Tustison	0.87	0.78	0.74	0.85	0.74	0.69	0.89	0.88	0.83
MFCASCADECNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
TWOPATHCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
LOCALCASCADECNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75
LOCALPATHCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
Meier	0.82	0.73	0.69	0.76	0.78	0.71	0.92	0.72	0.73
Reza	0.83	0.72	0.72	0.82	0.81	0.70	0.86	0.69	0.76
Zhao	0.84	0.70	0.65	0.80	0.67	0.65	0.89	0.79	0.70
Cordier	0.84	0.68	0.65	0.88	0.63	0.68	0.81	0.82	0.66
TWOPATHCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
LOCALPATHCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80
Festa	0.72	0.66	0.67	0.77	0.77	0.70	0.72	0.60	0.70
Doyle	0.71	0.46	0.52	0.66	0.38	0.58	0.87	0.70	0.55

Table 6.4 – Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2013 leaderboard set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
INPUTCASCADECNN*	0.84	0.71	0.57	0.88	0.79	0.54	0.84	0.72	0.68
Tustison	0.79	0.65	0.53	0.83	0.70	0.51	0.81	0.73	0.66
Zhao	0.79	0.59	0.47	0.77	0.55	0.50	0.85	0.77	0.53
Meier	0.72	0.60	0.53	0.65	0.62	0.48	0.88	0.69	0.6
Reza	0.73	0.56	0.51	0.68	0.64	0.48	0.79	0.57	0.63
Cordier	0.75	0.61	0.46	0.79	0.61	0.43	0.78	0.72	0.52

Table 6.5 – Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2012 "4 label" test set as discussed in [104].

Method	Dice		
	Complete	Core	Enhancing
INPUTCASCADECNN*	0.81	0.72	0.58
Subbanna	0.75	0.70	0.59
Zhao	0.82	0.66	0.42
Tustison	0.75	0.55	0.52
Festa	0.62	0.50	0.61

published state-of-the-art method both on accuracy and speed as presented in MICCAI 2013. The high performance is achieved with the help of a novel two-pathway architecture (which can model both the local details and global context) as well as modeling local label dependencies by stacking two CNN's. Training is based on a two phase procedure, which we've found allows us to train CNNs efficiently when the distribution of labels is unbalanced.

Thanks to the convolutional nature of the models and by using an efficient GPU implementation, the resulting segmentation system is very fast. The time needed to segment an entire brain with any of the these CNN architectures varies between 25 seconds and 3 minutes, making them practical segmentation methods.

Chapter 7

HeMIS:
Hetero-Modal Image Segmentation

Résumé

Most machine learning methods require a fixed number of input modalities. Also, some input modalities do not provide a significant amount of information to the model, yet their presence is necessary. Unfortunately in practice, it is hardly the case that all modalities are available. Acquiring new acquisitions means additional cost and time for the patient, which should be avoided unless absolutely necessary. In this work we address that problem by presenting a framework which is flexible in terms of the input modalities to the model. Using this framework, the model does its best to use the information it has to do a reasonable prediction and the accuracy improves by adding modalities. In some cases, the model is able to compensate for the missing modalities.

Commentaires

This article was accepted for oral presentation at the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) 2016.
The initial idea was proposed by the seniors involved in the project (i.e. Yoshua

Bengio and Nicolas Chapados). The idea was refined and perfected by the Ph.D. candiate and the second author (Nicolas Guizard). The main body of the python code was written by the Ph.D. candidate. The Ph.D. candidate carried out experiments for BRATS dataset. The Ph.D. candidate took active part in writing the paper.

HeMIS: Hetero-Modal Image Segmentation

Mohammad Havaei
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
seyed.mohammad.havaei@usherbrooke.ca

Nicolas Guizard
Imagia Inc., Montreal, Qc, Canada
nicolas.chapados@imagia.com

Nicolas Chapados
Imagia Inc., Montreal, Qc, Canada
nicolas.chapados@imagia.com

Yoshua Bengio
Université de Montréal, Montréal, Canada
yoshua.bengio@umontreal.ca

Keywords: Segmentation, multi-modal, deep learning, convolutional neural networks, data abstraction, data imputation

Abstract

We introduce a deep learning image segmentation framework that is extremely robust to missing imaging modalities. Instead of attempting to impute or synthesize missing data, the proposed approach learns, for each modality, an embedding of the input image into a single latent vector space for which arithmetic operations (such as taking the mean) are well defined. Points in that space, which are averaged over modalities available at inference time, can then be further processed to yield the desired segmentation. As such, any combinatorial subset of available modalities can be provided as input, without having to learn a combinatorial number of imputation models. Evaluated on two neurological MRI datasets (brain tumors and MS lesions), the approach yields state-of-the-art segmentation results when provided with all modalities; moreover, its performance degrades remarkably gracefully when modalities are removed, significantly more so than alternative mean-filling or other synthesis approaches.

7.1 Introduction

In medical image analysis, image segmentation is an important task and is primordial to visualize and quantify the severity of the pathology in clinical practice. Multi-modality imaging provides complementary information to discriminate specific tissues, anatomies and pathologies. However, manual segmentation is long, painstaking and subject to human variability. In the last decades, numerous automatic approaches have been developed to speed up medical image segmentation. These methods can be grouped into two categories: The first, *multi-atlas approaches* estimate on-line intensity similarities between the subject being segmented and multi-atlases (images with expert labels). These techniques have shown excellent results in structural segmentation when using non-linear registration [73]; when combined with non-local approaches they have proven effective in segmenting diffuse and sparse pathologies (ie. multiple sclerosis (MS) lesions [58]) as well as more complex multi-label pathology (ie. Glioblastoma [31]). In contrast, *model-based approaches* are typically trained offline to identify a discriminative model of image intensity features. These features can be predefined by the user (e.g. with random forests [50]) or extracted and learned hierarchically directly from the images [16].

Both strategies are typically optimized for a specific set of multi-modal images and usually require these modalities to be available. In clinical settings, image acquisition and patient artifacts, among other hurdles, make it difficult to fully exploit all the modalities; as such, it is common to have one or more modalities to be missing for a given instance. This problem is not new, and the subject of missing data analysis has spawned an immense literature in statistics (e.g. [165]). In medical imaging, a number of approaches have been proposed, some of which require to re-train a specific model with the missing modalities or to synthesize them [68]. Synthesis can improve multi-modal classification by adding information of the missing modalities in the context of a simple classifier such as random forests [158]. Approaches to imitate with fewer features a classifier trained with a complete set of features have also been proposed [70]. Nevertheless, it should stand to reason that a more complex model should be capable of extracting relevant features from just the available modalities without relying on artificial intermediate steps such as imputation or synthesis.

This paper proposes a deep learning framework (HeMIS) that can segment medical images from incomplete multi-modal datasets. Deep learning [54] has shown an increasing popularity in medical image processing for segmenting but also to synthesize missing modalities [158]. Here, the proposed approach learns, separately for each modality, an embedding of the input image into a latent space. In this space, arithmetic operations (such as computing first and second moments of a collection of vectors) are well defined and can be taken over the different modalities available at inference time. These computed moments can then be further processed to estimate the final segmentation. This approach presents the advantage of being robust to any combinatorial subset of available modalities provided as input, without the need to learn a combinatorial number of imputation models. We start by describing the method (§7.2), follow with a description of the datasets (§7.3) and experiments (§7.4) and finally offer concluding remarks (§7.5).

7.2 Method

7.2.1 Hetero-Modal Image Segmentation

Typical convolutional neural network (CNN) architectures take a multiplane image as input and process it through a sequence of convolutional layers (followed by nonlinearities such as $\text{ReLU}(\cdot) \equiv \max(0, \cdot)$), alternating with optional pooling layers, to yield a per-pixel or per-image output [54]. In such networks every input plane is assumed to be present within a given instance: since the very first convolutional layer mixes input values coming from all planes, any missing plane introduces a bias in the computation that the network is not equipped to deal with.

We propose an approach wherein each modality is initially processed by its own convolutional pipeline, independently of all others. After a few independent stages, feature maps from all available modalities are *merged* by computing mapwise statistics such as the mean and the variance, quantities whose expectation does not depend on the number of terms (i.e. modalities) that are provided. After merging, the mean and variance feature maps are concatenated and fed into a final set of convolutional stages to obtain network output. This is illustrated in Fig. 7.1. In this procedure, each modality contributes

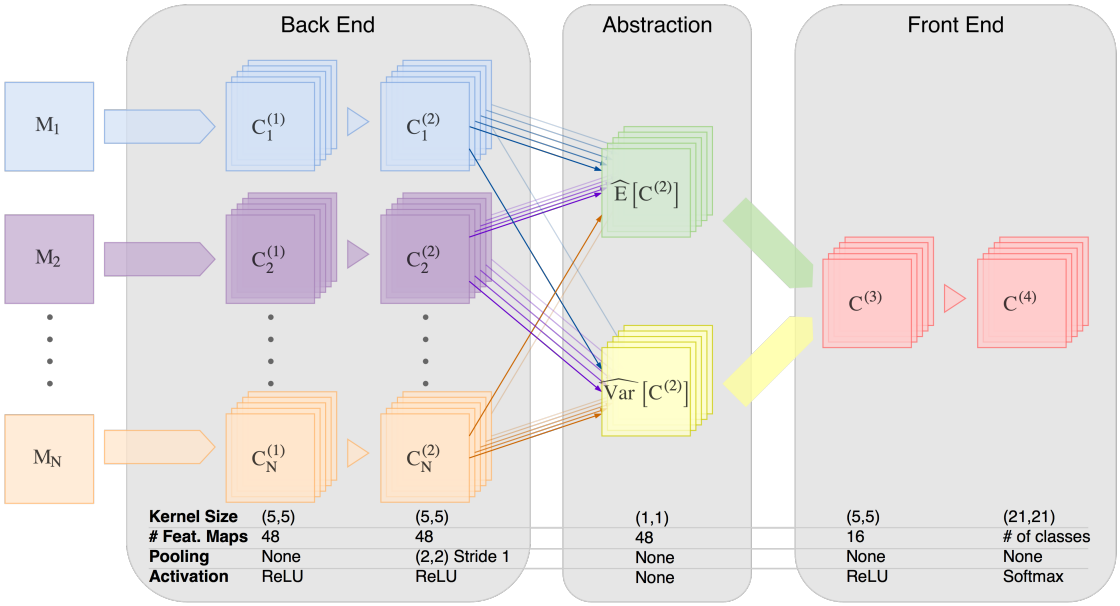


Figure 7.1 – Illustration of the Hetero-Modal Image Segmentation architecture. Modalities available at inference time, M_k, are provided to independent modality-specific convolutional layers in the **back end**. Feature maps statistics (first & second moments) are computed in the **abstraction layer**, which after concatenation are processed by further convolutional layers in the **front end**, yielding pixelwise classifications outputs.

a separate term to the mean and variance; in contrast to a vanilla CNN architecture, a missing modality does not throw this computation off: the mean and variance terms simply become estimated with larger uncertainty. In seeking to be robust to any subset of missing modalities, we call this approach *hetero-modal* rather than multi-modal, recognizing that in addition to taking advantage of several modalities, it can take advantage of a diverse, instance-varying, set of modalities. In particular, it does not require that a "least common denominator" modality be present for every instance, as sometimes needed by common imputation methods.

Let $k \in \mathcal{K} \subseteq \{1, \ldots, N\}$ denote a modality within the set of available modalities for a given instance, and M_k represent the image of the k-th modality. For simplicity, in this work we assume 2D data (e.g. a single slice of a tomographic image), but it can be extended in an obvious way to full 3D sections. As shown on Fig. 7.1, HeMIS proceeds in three stages:

1. Back End: In our implementation, this consists of two convolutional layers with

ReLU, the second followed with a $(2, 2)$ max-pooling layer, denoted respectively $C_k^{(1)}$ and $C_k^{(2)}$. To ensure that the output layer consists of the same number of pixels as the input image, the convolutions are zero-padded and the stride for all operations (including max-pooling) is 1. In particular, pooling with a stride of 1 *does not downsample*, but simply "thickens" the feature maps; this is found to add some robustness to the results. The number of feature maps in each layer is given in Fig. 7.1. Let $C_{k,\ell}^{(j)}$ be the the ℓ-th feature map of $C_k^{(j)}$.

2. Abstraction Layer: Modality fusion is computed here, as first and second moments across available modalities in $C^{(2)}$, separately for each feature map ℓ,

$$\widehat{\mathrm{E}}_\ell \left[C^{(2)}\right] = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} C_{k,\ell}^{(2)} \quad \text{and} \quad \widehat{\mathrm{Var}}_\ell \left[C^{(2)}\right] = \frac{1}{|\mathcal{K}| - 1} \sum_{k \in \mathcal{K}} \left(C_{k,\ell}^{(2)} - \widehat{\mathrm{E}}_\ell \left[C^{(2)}\right]\right)^2,$$

with $\widehat{\mathrm{Var}}_\ell[C^{(2)}]$ defined to be zero if $|\mathcal{K}| = 1$ (a single available modality).

3. Front End: Finally the front end combines the merged modalities to produce the final model output. In our implementation, we concatenate all $\widehat{\mathrm{E}}\left[C^{(2)}\right]$ and $\widehat{\mathrm{Var}}\left[C^{(2)}\right]$ feature maps, pass them through a convolutional layer $C^{(3)}$ with ReLU activation, to finish with a final layer $C^{(4)}$ that has as many feature maps as there are target segmentation classes. The pixelwise posterior class probabilities are given by applying a softmax function across the $C^{(4)}$ feature maps, and a full image segmentation is obtained by taking the pixelwise most likely posterior class. No further postprocessing on the resulting segment classes (such as smoothing) is done.

Pseudo-Curriculum Training Procedure To carry out segmentation efficiently, the model is trained fully convolutionnally to minimize a pixelwise class cross-entropy loss, in the spirit of [96]. It has long been known that noise injection during training is a powerful technique to make neural networks more robust, as shown among others with denoising autoencoders [168], and dropout and related procedures [148]. Here, we make the HeMIS architecture robust to missing modalities by randomly dropping any number for a given training example. Inspired by previous works on curriculum learning [14]—where the model starts learning from easy scenarios before turning to more difficult ones—we used a pseudo-curriculum learning scheme where after few warmup epochs where all modalities are shown to the model, we start randomly dropping modalities,

ensuring a higher probability of dropping zero or one modality only.

Interpretation as an Embedding An embedding is a mapping from an arbitrary source space to a target real-valued vector space of fixed dimensionality. In recent years, embeddings have been shown to yield unexpectedly powerful representations for a wide array of data types, including single words [105], variable-length word sequences and images [174], and more.

In the context of HeMIS, the back end can be interpreted as learning to separately map each modality into an *embedding common to all modalities*, within which vector algebra operations carry well-defined semantics. As such, computing empirical moments to carry out modality fusion is sensible. Since the model is trained entirely end-to-end with backpropagation, the key aspect of the architecture is that this embedding only needs be defined implicitly as that which minimizes the overall training loss. Cross-modality interactions can be captured within specific embedding dimensions, as long as there are a sufficient number of them (i.e. enough feature maps within $C^{(2)}$), as they can be combined by $C^{(3)}$.

With this interpretation, the back end consists of a modular assembly of operators, viewed as reusable building blocks that may or may not be needed for a given instance, each computing the embedding from its own input modality. These projections are summarized in the abstraction layer (with a mean and variance, although additional summary statistics are simple to entertain), and this summary further processed in the front end to yield final model output.

7.3 Data and Implementation details

We studied the HeMIS framework on two neurological pathologies: Multiple Sclerosis (MS) with the MS Grand Challenge (MSGC) and a large Relapsing Remitting MS (RRMS) cohort, as well as glioma with the Brain Tumor Segmentation (BRATS) dataset [104].

MS *MSGC:* The MSGC dataset [150] provides 20 training MR cases with manual ground truth lesion segmentation and 23 testing cases from the Boston Children's Hos-

pital (CHB) and the University of North Carolina (UNC). We downloaded [1] the co-registered T1W, T2W, FLAIR images for all 43 cases as well as the ground truth lesion mask images for the 20 training cases. While lesions masks for the 23 testing cases are not available for download, an automated system is available to evaluate the output of a given segmentation algorithm.

RRMS: This dataset is obtained from a multi-site clinical study with 300 relapsing-remitting MS (RRMS) patients (mean age 37.5 yrs, SD 10.0 yrs). Each patient underwent an MRI that included sagittal T1W , T2W and T1 post-contrast (T1C) images. The MRI data were acquired on 1.5T scanners from different manufacturers (GE, Philips and Siemens).

BRATS The *BRATS-2015* dataset contains 220 subjects with high grade and 54 subjects with low grade tumors. Each subject contains four MR modalities (FLAIR, T1W, T1C and T2) and comes with a voxel level segmentation ground truth of 5 labels: *healthy*, *necrosis*, *edema*, *non-enhancing tumor* and *enhancing tumor*. As done in [104], we transform each segmentation map into 3 binary maps which correspond to 3 tumor categories, namely; *Complete* (which contains all tumor classes), *Core* (which contains all tumor subclasses except "edema") and *Enhancing* (which includes the "enhanced tumor" subclass). For each binary map, the Dice Similarity Coefficient (DSC) is calculated [104].

BRATS-2013 contains two test datasets; Challenge and Leaderboard. The Challenge dataset contains 10 subjects with high grade tumors while the Leaderboard dataset contains 15 subjects with high grade tumors and 10 subject with low grade tumors. There are no ground truth provided for these datasets and thus quantitative evaluation can be achieved via an online evaluation system [104]. In our experiments we used Challenge and Leaderboard datasets to compare the HeMIS segmentation performance to the state-of-the-art, when trained on all modalities.

Pre-processing and implementation details Before being provided to the network, bias field correction [143] and intensity normalization with a zero mean, truncation of 0.001 quantile and unit variance is applied to the image intensity. The multi-modal images are co-registered to the T1W and interpolated to 1mm isotropic resolution.

1. http://www.nitrc.org/projects/msseg/

We used Keras library [23] for our implementation. To deal with class imbalance, we adopt the patch-wise training procedure mentioned in [66]. We first train the model with a balanced dataset which allows learning features that are agnostic to the class distribution. In a second phase, we train only the final classification layer with a distribution close to the ground truth. This ensures that we learn good features yet keep the correct class priors. The method was trained using an Nvidia TitanX GPU, with a stochastic gradient learning rate of 0.001, decay constant of 0.0001 and Nesterov momentum coefficient of 0.9 [155]. For both BRATS-2015 and MS, we split the dataset into three separate subsets—train, valid and test—with ratios of 70%, 10% and 20% respectively. To avoid over-fitting we used early stopping on the validation set.

7.4 Experiments and Results

We first validate HeMIS performance against state-of-the-art segmentation methods on the two challenge datasets: MSGC and BRATS. Since the test data and the ranking table for BRATS 2015 are not available, we submitted results to BRATS 2013 challenge and leaderboard. These results are presented in Table 7.1. [2] As we observe, HeMIS outperforms Tustison et al. [160], the winner of the BRATS 2013 challenge, on most tumor region categories.

The MSGC dataset illustrates a direct application of HeMIS flexibility as only three modalities (T1W, T2W and FLAIR) are provided for a small training set. Therefore, given the small number of subjects, we first trained HeMIS on RRMS dataset with four modalities and fine-tuned on MSGC. Our results were submitted to the MSGC website [3], with a resuts summary appearing in Table 7.2. The MSGC segmentation results include three other supervised approaches; when compared to them, HeMIS obtains highly competitive results with a combined score of 83.2%, where 90.0% would represent human performance given inter-rater variability.

The main advantage of HeMIS lies in its ability to deal with missing modalities, specif-

2. Note that the results mentioned in Table 7.1 are from methods competing in the BRATS 2013 challenge for which a static table is provided at https://www.virtualskeleton.ch/BraTS/StaticResults2013. Since then, other methods have been added to the scoreboard but for which no reference is available.

3. http://www.ia.unc.edu/MSseg

Table 7.1 – Comparison of HeMIS when trained on all modalities against BRATS-2013 Leaderboard and Challenge winners, in terms of Dice Similarity (scores from [104]).

	Leaderboard			**Challenge**		
Method	Complete	Core	Enhancing	Complete	Core	Enhancing
Tustison [160]	79	65	53	87	**78**	**74**
Zhao [182]	79	59	47	84	70	65
Meier [104]	72	60	53	82	73	69
HeMIS	**83**	**67**	**57**	**88**	75	**74**

Figure 7.2 – MLP-imputed FLAIR for an MS patient. The figure shows from left to right the original modality and the predicted FLAIR given other modalities.

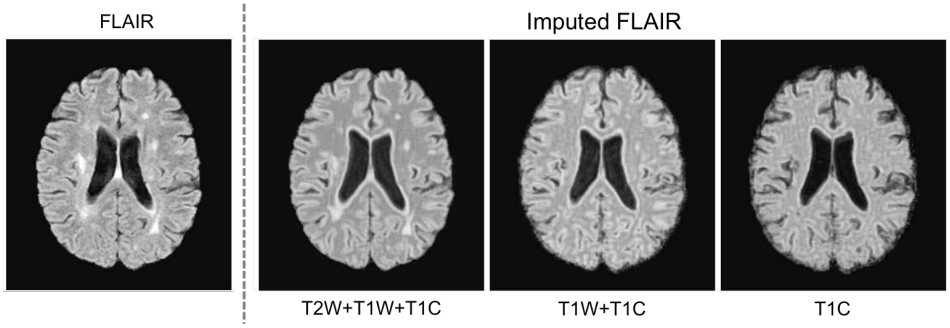


Table 7.2 – Results of the full dataset training on the MSGC. For each rater (CHB and UNC), we provide the volume difference (VD), surface distance (SD), true positive rate (TPR), false positive rate (FPR) and the method's score as in [150].

Method	Rater	VD (%)	SD (mm)	TPR (%)	FPR (%)	Score
Souplet et al. [145]	CHB	86.4	8.4	58.2	70.6	80.0
	UNC	57.9	7.5	49.1	76.3	
Geremia et al. [50]	CHB	**52.4**	**5.4**	59.0	71.5	82.1
	UNC	**45.0**	**5.7**	51.2	76.7	
Brosch et al. [16]	CHB	63.5	7.4	47.1	**52.7**	**84.0**
	UNC	52.0	6.4	**56.0**	**49.8**	
HeMIS	CHB	127.4	7.5	**66.1**	55.3	83.2
	UNC	68.2	6.6	52.3	61.3	

ically when different subjects are missing different modalities. To illustrate the model's flexibility in such circumstances, we compare HeMIS performance to two common ap-

Table 7.3 – Dice similarity coefficient (DSC) results on the RRMS and BRATS test sets (%) when modalities are dropped. The table shows the DSC for all possible configurations of MRI modalities being either absent (○) or present (●), in order of FLAIR (F), T1W (T_1), T1C (T_1c), T2W (T_2). Results are reported for HeMIS, Mean (mean-filling) and the imputation MLP (MLP).

				RRMS			BRATS								
Modalities				Lesion			Complete			Core			Enhancing		
F	T_1	T_1c	T_2	HeMIS	Mean	MLP	HeMIS	Mean	MLP	HeMIS	Mean	MLP	HeMIS	Mean	MLP
○	○	○	●	1.74	2.66	**12.77**	58.48	2.70	**61.50**	**40.18**	4.00	37.32	**20.31**	6.25	18.62
○	○	●	○	2.67	0.00	**3.51**	**33.46**	23.11	2.04	**44.55**	23.90	17.70	**49.93**	30.02	32.92
○	●	○	○	3.89	0.00	**6.64**	**33.22**	0.00	2.07	**17.42**	0.00	10.52	4.67	6.25	**10.78**
●	○	○	○	34.48	9.77	**38.46**	71.26	**72.30**	63.81	**37.45**	0.00	34.26	5.57	6.25	**15.90**
○	○	●	●	**27.52**	4.31	25.83	**67.59**	35.01	64.97	**63.39**	30.92	49.38	**65.38**	39.00	60.30
○	●	●	○	8.21	0.00	**8.26**	**45.93**	23.63	1.99	**55.06**	41.89	26.55	**62.40**	43.80	40.93
●	●	○	○	38.81	11.62	**39.15**	**80.28**	75.58	78.13	**49.52**	0.00	48.97	22.26	6.25	**25.18**
○	●	○	●	**31.25**	8.31	29.39	**69.56**	1.77	66.88	**47.26**	2.63	43.66	23.56	6.25	**26.37**
●	○	○	●	**39.64**	33.31	38.55	**82.1**	81.01	81.35	**53.42**	25.94	52.41	23.19	6.25	**25.01**
●	○	●	○	**41.38**	6.42	39.33	79.8	45.97	**81.13**	**66.12**	29.85	65.51	**67.12**	35.14	66.19
●	●	●	○	**41.97**	9.00	40.63	80.88	81.57	**82.19**	69.26	62.13	**69.34**	**71.30**	67.13	70.93
●	●	○	●	**46.6**	41.12	41.83	**83.87**	77.84	80.40	**57.76**	20.66	53.46	**28.46**	6.25	28.34
●	○	●	●	**41.90**	38.95	41.47	82.78	64.19	**83.37**	**70.62**	42.36	70.45	70.52	49.62	**70.56**
○	●	●	●	**34.98**	5.78	29.46	**70.98**	30.86	67.85	**66.60**	45.79	55.40	**67.84**	50.21	64.81
●	●	●	●	**48.66**	43.48	43.48	**83.15**	82.43	82.43	**72.5**	71.46	71.46	**75.37**	72.08	72.08
# Wins / 15				**9**	0	6	**10**	1	4	**14**	0	1	**9**	0	6

proaches to deal with random missing modalities. The first, *mean-filling*, is to replace a missing modality by the modality's mean value. In our case since all means are zero by construction, replacing a missing modality by zeros can be viewed as imputing with the mean. The second approach is to train a *multi-layer perceptron* (MLP) to predict the expected value of specific missing modality given the available ones. Since neural networks are generally trained for a unique task, we need to train 28 different MLPs (one for each ○ in Table 7.3 for a given dataset) to account for different possibilities of missing modalities. We used the same MLP architecture for all these models, which consists of 2 hidden layers with 100 hidden units each, trained to minimize the mean squared error. Fig. 7.2 shows an example of predicted modalities for an MS patient.

Table 7.3 shows the DSC for this experiment on the test set. On the BRATS dataset, for the Core category, HeMIS achieves the best segmentation in almost all cases (14 out of 15) and for the Complete and Enhancing categories it leads in most cases (10

and 9 cases out of 15 respectively). Also, the mean-filling approach hardly outperforms HeMIS or MLP-imputation. These results are consistent with the MS lesion segmentation dataset, where HeMIS outperforms other imputation approaches in 9 out of 15 cases. In scenarios where only one or two modalities are missing, while both HeMIS and MLP-imputation obtain good results, HeMIS outperforms the latter in most cases on both datasets. On BRATS, when missing 3 out of 4 modalities, HeMIS outperforms the MLP in a majority of cases. Moreover, whereas the HeMIS performance only gradually drops as additional modalities become missing, the performance drop for MLP-imputation and mean-filling is much more severe. On the RRMS cohort, the MLP-imputation appears to obtain slightly better segmentations when only one modality is available.

Although it is expected that tumor sub-label segmentations should be less accurate with fewer modalities, we should still hope for the model to report a sensible characterization of the tumor "footprint". While MLP and mean-filling fail in this respect, HeMIS quite well achieves this goal by outperforming in almost all cases of the Complete and Core tumor categories. This can also be seen in Fig. 7.3 where we show how adding modalities to HeMIS improves its ability to achieve a more accurate segmentation. From Table 7.3, we can also infer that the FLAIR modality is the most relevant for identifying the Complete tumor while T1C is the most relevant for identifying Core and Enhancing tumor categories. On the RRMS dataset, HeMIS results are also seen to degrade slower than the other imputation approaches, preserving good segmentation when modalities go missing. Indeed, as seen in Fig. 7.3, even though with FLAIR alone HeMIS already produces good segmentations, it is capable of further refining its results when adding modalities, by removing false positives and improving outlines of the correctly identified lesions or tumor.

7.5 Conclusion

We have proposed a new fully automatic segmentation framework for heterogenous multi-modal MRI using a specialized convolutional deep neural network. The embedding of the multi-modal CNN back-end allows to train and segment datasets with miss-

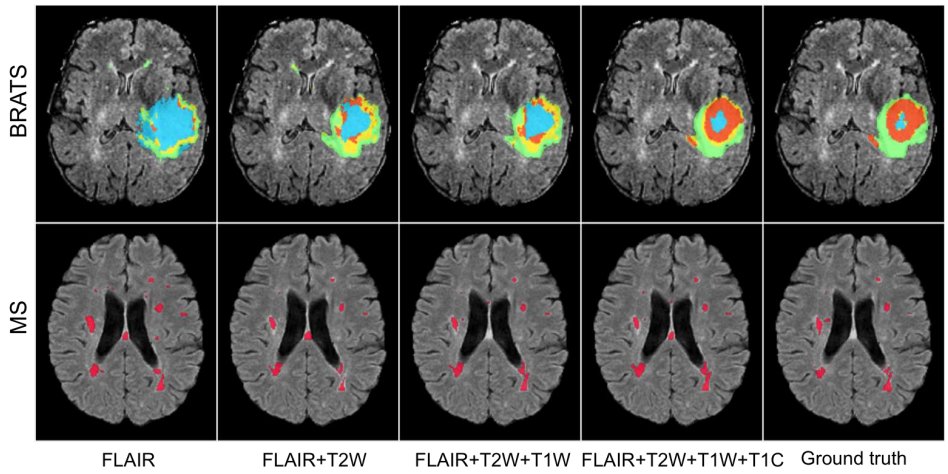


Figure 7.3 – Example of HeMIS segmentation results on BRATS and MS subjects for different combinations of input modalities. For both cohorts, an axial FLAIR slice of a subject is overlaid with the results where for BRATS (first row) the segmentation colors describe necrosis (blue), non-enhancing (yellow), active core (orange) and edema (green). For the MS case, the lesions are highlighted in red. The columns present the results for different combinations of input modalities, with ground truth in the last column.

ing modalities. We carried out an extensive validation on MS and glioma and achieved state-of-the art segmentation results on two challenging neurological pathology image processing tasks. Importantly, we contrasted the graceful performance degradation of the proposed approach as modalities go missing, compared with other popular imputation approaches, which it achieves without requiring training specific models for every potential missing modality combination. Future work should concentrate on extending the approach to broader modalities outside of MRI, such as CT, PET and ultrasound.

Conclusion

There are many challenges facing brain tumor segmentation. Some of these challenges are due to the natural properties of tumors themselves and some arise due to the image acquisition process. While machine learning methods have been a source of great interest for brain tumor segmentation, they are vulnerable to these challenges. This study was set out to explore solutions to various challenges facing machine learning methods applied to brain tumor segmentation.

We first explored an interactive method, where with minimum user interaction we were able to produce accurate segmentations in about 1 to 2 minutes on CPU. Since the training and generalization is done within each brain, we are free from performing extensive pre-processing steps. Considering the fact that the method can be highly parallelized, it is possible to have a real time implementation on GPU. This method can be used in health care institutions to help physicians to localize and segment brain tumors.

As interesting and beneficial that a semi-automatic method can be, it is still limited by its reliance on human intervention. As a second step in this doctorate, we developed a fully automatic method for brain tumor segmentation using deep neural networks. Instead of using hand crafting features, neural networks have the advantage of learning task specific features. Also, these models learn a hierarchy of increasingly complex features which resemble the way human visual cortex works. These capabilities allow models based on neural networks to achieve higher performance than traditional machine learning methods. Our proposed method was selected as one of the 4 winners of the brain tumor segmentation challenge (BRATS) in 2015.

In clinical settings, not all modalities are always present. This creates a challenge for machine learning methods which assume all modalities used during training are present

at test time. As the final part of this research, we developed a framework which is flexible with respect to the input modalities it requires. This framework can be used in any multi-modality problem scenario and can compete with state-of-the-art methods in brain tumor segmentation.

Although deep learning methods have proven to have potential in medical image analysis applications, their performance depends highly on the amount of training data and they tend to perform poorly when training data is limited. A potential solution is to utilize synthetically generated data. However, synthetic data do not lie on the same manifold as the real MRI data. This situation is also true when different subjects are gathered from different institutions where the acquisition protocol is different. Thus, learning robust representations which are invariant to the acquisition procedure is needed. Unsupervised learning might hold the key to this problem. Also methods based on domain adaptation as in [1], might help us learn representations which better explain the anatomy of the brain and can better generalize across datasets.

Bibliography

[1] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[2] H. Ali, M. Elmogy, E. El-Daydamony, and A. Atwan. Multi-resolution mri brain image segmentation based on morphological pyramid and fuzzy c-mean clustering. *Arabian Journal for Science and Engineering*, 40(11):3173–3185, 2015.

[3] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez. Road scene segmentation from a single image. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VII*, ECCV'12, pages 376–389, Berlin, Heidelberg, 2012. Springer-Verlag.

[4] E. D. Angelini, O. Clatz, E. Mandonnet, E. Konukoglu, L. Capelle, and H. Duffau. Glioma dynamics and computational models: a review of segmentation, registration, and in silico growth algorithms and their clinical applications. *Current Medical Imaging Reviews*, 3(4):262–276, 2007.

[5] N. Archip, F. A. Jolesz, and S. K. Warfield. A validation framework for brain tumor segmentation. *Academic radiology*, 14(10):1242–1251, 2007.

[6] J. Arevalo, F. A. Gonzalez, R. Ramos-Pollan, J. L. Oliveira, and M. A. Guevara Lopez. Convolutional neural networks for mammography mass lesion classification. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 797–800. IEEE, 2015.

[7] B. B. Avants, N. Tustison, and G. Song. Advanced normalization tools (ants). *Insight J*, 2:1–35, 2009.

[8] S. Bakas, K. Zeng, A. Sotiras, S. Rathore, H. Akbari, B. Gaonkar, M. Rozycki, S. Pati, and C. Davazikos. Segmentation of gliomas in multimodal magnetic resonance imaging volumes based on a hybrid generative-discriminative framework. *Proceeding of the Multimodal Brain Tumor Image Segmentation Challenge*, pages 5–12, 2015.

[9] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan. Deep learning with non-medical training used for chest pathology identification. In *SPIE Medical Imaging*, pages 94140V–94140V. International Society for Optics and Photonics, 2015.

[10] S. Bauer, L. Nolte, and M. Reyes. Fully automatic segmentation of brain tumor images using support vector machine classification in combination with hierarchical conditional random field regularization. In *proc. of MICCAI*, pages 354–361. Springer, 2011.

[11] S. Bauer, R. Wiest, and M. Reyes. Segmentation of brain tumor images based on integrated hierarchical classification and regularization. *in proc of BRATS-MICCAI*, pages 32–38, 2012.

[12] S. Bauer, R. Wiest, L. Nolte, and M. Reyes. A survey of mri-based medical image analysis for brain tumor studies. *Physics in medicine and biology*, 58(13): 97–129, 2013.

[13] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[14] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[15] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[16] T. Brosch, Y. Yoo, L. Y. Tang, D. K. Li, A. Traboulsee, and R. Tam. Deep convolutional encoder networks for multiple sclerosis lesion segmentation. In

International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 3–11. Springer, 2015.

[17] T. Brosch, L. Tang, Y. Yoo, D. Li, A. Traboulsee, and R. Tam. Deep 3d convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. *Medical Imaging, IEEE Transactions on*, 2016.

[18] N. Buduma. Fundamentals of deep learning. https://www.safaribooksonline.com, 2017.

[19] H. Cai, R. Verma, Y. Ou, S.-k. Lee, E. R. Melhem, and C. Davatzikos. Probabilistic segmentation of brain tumors based on multi-modality magnetic resonance images. In *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on*, pages 600–603. IEEE, 2007.

[20] A.-S. Capelle, O. Alata, C. Fernandez, S. Lefèvre, and J. Ferrie. Unsupervised segmentation for automatic detection of brain tumors in mri. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 1, pages 613–616. IEEE, 2000.

[21] G. Carneiro, J. Nascimento, and A. P. Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 652–660. Springer, 2015.

[22] H. Chen, D. Ni, J. Qin, S. Li, X. Yang, T. Wang, and P. A. Heng. Standard plane localization in fetal ultrasound via domain transferred deep neural networks. *Biomedical and Health Informatics, IEEE Journal of*, 19(5):1627–1636, 2015.

[23] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[24] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[25] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[26] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013*, pages 411–418. Springer, 2013.

[27] M. Clark, L. Hall, D. Goldgof, R. P. Velthuizen, F. Murtagh, and M. L. Silbiger. Automatic tumor segmentation using knowledge-based clustering. *IEEE Trans. Med. Imaging*, 17(2):187–201, 1998.

[28] J. Clarke, N. Butowski, and S. Chang. Recent advances in therapy for glioblastoma. *Archives of neurology*, 67(3):279–283, 2010.

[29] D. Cobzas, N. Birkbeck, M. Schmidt, M. Jägersand, and A. Murtha. 3d variational brain tumor segmentation using a high dimensional feature set. In *ICCV*, pages 1–8, 2007.

[30] A. A. Constantin, B. R. Bajcsy, and C. S. Nelson. Unsupervised segmentation of brain tissue in multivariate mri. In *Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on*, pages 89–92. IEEE, 2010.

[31] N. Cordier, H. Delingette, and N. Ayache. A patch-based approach for the segmentation of pathologies: Application to glioma labelling. *IEEE TMI*, PP(99): 1–1, 2016. ISSN 0278-0062.

[32] J. J. Corso, E. Sharon, and A. Yuille. Multilevel segmentation and integrated bayesian model classification with an application to brain tumor segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*, pages 790–798. Springer, 2006.

[33] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629–640, 2008.

[34] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629–640, 2008.

[35] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[36] A. Davy, M. Havaei, D. Warde-Farley, A. Biard, L. Tran, P.-M. Jodoin, A. Courville, H. Larochelle, C. Pal, and Y. Bengio. Brain tumor segmentation with deep neural networks. *in proc of BRATS-MICCAI*, 2014.

[37] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[38] S. Doyle, F. Vasseur, M. Dojat, and F. Forbes. Fully automatic brain tumor segmentation from multiple mr sequences using hidden markov fields and variational em. *in proc of BRATS-MICCAI*, 2013.

[39] P. Dvorak and B. Menze. Structured prediction with convolutional neural networks for multimodal brain tumor segmentation. *Proceeding of the Multimodal Brain Tumor Image Segmentation Challenge*, pages 13–24, 2015.

[40] K. Egger, O. Maier, M. Reyes, and R. Wiest. Isles challenge 2015: Ischemic stroke lesion segmentation. http://www.isles-challenge.org/ISLES2015/, 2015. Accessed: 2016-06-11.

[41] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[42] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

[43] K. Farahani, B. Menze, and M. Reyes. *Multimodal Brain Tumor Segmentation (BRATS 2013)*, 2013.

[44] K. Farahani, B. Menze, and M. Reyes. *Brats 2014 Challenge Manuscripts*, 2014.

[45] K. Farahani, B. Menze, and M. Reyes. *Brats 2015 Challenge Manuscripts*, 2015.

[46] J. Festa, S. Pereira, J. Mariz, N. Sousa, and C. Silva. Automatic brain tumor segmentation of multi-sequence mr images using random dicision forests. *Proc. Workshop on Brain Tumor Segmentation MICCAI.*, 2013.

[47] D. Gai and J. Jones. *T1 weighted images*, 2016.

[48] D. Gai and J. Jones. *T2 weighted images*, 2016.

[49] M. Gao, U. Bagci, L. Lu, A. Wu, M. Buty, H.-C. Shin, H. Roth, G. Z. Papadakis, A. Depeursinge, R. M. Summers, et al. Holistic classification of ct attenuation patterns for interstitial lung diseases via deep convolutional neural networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1–6, 2016.

[50] E. Geremia, B. H. Menze, and N. Ayache. Spatially adaptive random forests. In *2013 IEEE 10th International Symposium on Biomedical Imaging*, pages 1344–1347. IEEE, 2013.

[51] D. Girardi, J. Küng, R. Kleiser, M. Sonnberger, D. Csillag, J. Trenkler, and A. Holzinger. Interactive knowledge discovery with the doctor-in-the-loop: a practical example of cerebral aneurysms research. *Brain Informatics*, pages 1–11, 2016.

[52] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011.

[53] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[54] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[55] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[56] M. Gotz, C. Weber, J. Blocher, B. Stieltjes, H.-P. Meinzer, and K. Maier-Hein. Extremely randomized trees based brain tumor segmentation. In *in proc of BRATS Challenge - MICCAI*, 2014.

[57] S. Goyal, S. Shekhar, and K. Biswas. Automatic detection of brain abnormalities and tumor segmentation in mri sequences. In *Image and Vision Computing New Zealand Conference, IVCNZ*, 2011.

[58] N. Guizard, P. Coupé, V. S. Fonov, J. V. Manjón, D. L. Arnold, and D. L. Collins. Rotation-invariant multi-contrast non-local means for ms lesion segmentation. *NeuroImage: Clinical*, 8:376–389, 2015.

[59] A. Hamamci and G. Unal. Multimodal brain tumor segmentation using the tumor-cut method on the brats dataset. *Proc. Workshod on Brain Tumor Segmentation, MICCAI*, pages 19–23, 2012.

[60] A. Hamamci, N. Kucuk, K. Karaman, K. Engin, and G. Unal. Tumor-cut: Segmentation of brain tumors on contrast enhanced mr images for radiosurgery applications. *IEEE trans. Medical Imaging*, 31(3):790–804, 2012.

[61] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, pages 297–312. Springer, 2014.

[62] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[63] M. Havaei, P.-M. Jodoin, and H. Larochelle. Efficient interactive brain tumor segmentation as within-brain knn classification. In *2014 22nd International Conference on Pattern Recognition (ICPR)*, pages 556–561. IEEE, 2014.

[64] M. Havaei, F. Dutil, C. Pal, H. Larochelle, and P.-M. Jodoin. A convolutional neural network approach to brain tumor segmentation. In *Brainlesion: Glioma, Mul-*

tiple Sclerosis, Stroke and Traumatic Brain Injuries, pages 195–208. Springer, 2015.

[65] M. Havaei, H. Larochelle, P. Poulin, and P.-M. Jodoin. Within-brain classification for brain tumor segmentation. *International journal of computer assisted radiology and surgery*, pages 1–12, 2015.

[66] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, 2016. ISSN 1361-8415.

[67] S. Ho, E. Bullitt, and G. Gerig. Level-set evolution with region competition: automatic 3-d segmentation of brain tumors. In *Proc. Int. Conf. Pattern Recognition*, volume 1, pages 532–535, 2002.

[68] M. Hofmann, F. Steinke, V. Scheel, G. Charpiat, J. Farquhar, P. Aschoff, M. Brady, B. Schölkopf, and B. J. Pichler. MRI-based attenuation correction for PET/MRI: a novel approach combining pattern recognition and atlas registration. *J. Nuclear Medicine*, 49(11):1875–1883, 2008.

[69] A. Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[70] S. Hor and M. Moradi. Scandent tree: A random forest learning method for incomplete multimodal datasets. In *MICCAI 2015*, pages 694–701. Springer, 2015.

[71] G. B. Huang and V. Jain. Deep and wide multiscale recursive networks for robust image labeling. *ICLR, arXiv:1310.0354*, 2014.

[72] J. Huo, K. Okada, E. M. van Rikxoort, H. J. Kim, J. R. Alger, W. B. Pope, J. G. Goldin, and M. S. Brown. Ensemble segmentation for gbm brain tumors on mr images using confidence-based averaging. *Medical physics*, 40(9):093502, 2013.

[73] J. E. Iglesias and M. R. Sabuncu. Multi-atlas segmentation of biomedical images: A survey. *Medical image analysis*, 24(1):205–219, 2015.

[74] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153, Sept 2009.

[75] T. R. Jensen and K. M. Schmainda. Computer-aided detection of brain tumor invasion using multiparametric mri. *Journal of Magnetic Resonance Imaging*, 30 (3):481–489, 2009.

[76] C. Jiang, X. Zhang, W. Huang, and C. Meinel. Segmentation and quantification of brain tumor. In *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VECIMS). 2004 IEEE Symposium on*, pages 61–66, 2004.

[77] P.-M. Jodoin. Vision par ordinateur imn 786. In *Vision par ordinateur*, 2013.

[78] K. Kamnitsas, L. Chen, C. Ledig, D. Rueckert, and B. Glocker. Multi-scale 3d convolutional neural networks for lesion segmentation in brain mri. *Ischemic Stroke Lesion Segmentation*, page 13, 2015.

[79] M. Kaus, S. K. Warfield, F. A. Jolesz, and R. Kikinis. Adaptive template moderated brain tumor segmentation in mri. In *Bildverarbeitung für die Medizin 1999*, pages 102–106. Springer, 1999.

[80] M. R. Kaus, S. K. Warfield, A. Nabavi, P. M. Black, F. A. Jolesz, and R. Kikinis. Automated segmentation of mr images of brain tumors1. *Radiology*, 218(2): 586–591, 2001.

[81] H. Khotanlou, O. Colliot, J. Atif, and I. Bloch. 3d brain tumor segmentation in mri using fuzzy classification, symmetry analysis and spatially constrained deformable models. *Fuzzy Sets Syst.*, 160(10):1457–1473, 2009.

[82] H. Khotanlou, O. Colliot, and I. Bloch. Automatic brain tumor segmentation using symmetry analysis and deformable models. In *International Conference on Advances in Pattern Recognition ICAPR*, pages 198–202, 2007.

[83] J. Kleesiek, A. Biller, G. Urban, U. Kothe, M. Bendszus, and F. A. Hamprecht. ilastik for multi-modal brain tumor segmentation. *in proc of BRATS-MICCAI*, 2014.

[84] T. Klein, N. Batmanghelich, and W. W. III. Distributed deep learning framework for large-scale 3d medical image segmentation. In *MICCAI*, volume 18. Int Conf Med Image Comput Comput Assist Interv. MICCAI 2015, 10 2015.

[85] J. Kovalic, P. Grigsby, and B. Fineberg. Recurrent pituitary adenomas after surgical resection: the role of radiation therapy. *Radiology*, 177(1):273–275, 1990.

[86] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, volume 1, pages 1097–1105. NIPS, 2012.

[87] D. Kwon, H. Akbari, X. Da, B. Gaonkar, and C. Davatzikos. Multimodal brain tumor image segmentation using glistr. In *proc of BRATS Challenge - MICCAI*, 2014.

[88] C. H. Lampert. Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3):193–285, 2009.

[89] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[90] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[91] C.-H. Lee, M. Schmidt, A. Murtha, A. Bistritz, J. S, and R. Greiner. Segmenting brain tumor with conditional random fields and support vector machines. In *in Proc of Workshop on Computer Vision for Biomedical Image Applications*, 2005.

[92] C.-H. Lee, R. Greiner, and O. Zaïane. Efficient spatial classification using decoupled conditional random fields. In *Knowledge Discovery in Databases: PKDD 2006*, pages 272–283. Springer, 2006.

[93] C.-H. Lee, S. Wang, A. Murtha, M. R. Brown, and R. Greiner. Segmenting brain tumors using pseudo–conditional random fields. In *Medical Image Computing and Computer-Assisted Intervention*, pages 359–366. Springer, 2008.

[94] H. Lee, A. Smeaton, N. O'Connor, and N. Murphy. User-interface to a cctv video search system. In *IEE Int Symp on Imaging for Crime Detec. and Prev.*, pages 39–43, 2005.

[95] R. Li, W. Zhang, H.-I. Suk, L. Wang, J. Li, D. Shen, and S. Ji. Deep learning based imaging data completion for improved brain disease diagnosis. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pages 305–312. Springer, 2014.

[96] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE CVPR*, pages 3431–3440, 2015.

[97] D. N. Louis, H. Ohgaki, O. D. Wiestler, W. K. Cavenee, P. C. Burger, A. Jouvet, B. W. Scheithauer, and P. Kleihues. The 2007 who classification of tumours of the central nervous system. *Acta neuropathologica*, 114(2):97–109, 2007.

[98] J. Luts, A. Heerschap, J. Suykens, and S. V. Huffel. A combined mri and mrsi based multiclass system for brain tumour recognition using ls-svms with class probabilities and feature selection. *Artificial Intelligence in Medicine*, 40(2):87–102, 2007.

[99] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 1, 2013.

[100] J.-F. Mangin, O. Coulon, and V. Frouin. Robust brain segmentation using histogram scale-space analysis and mathematical morphology. In *Medical Image Computing and Computer-Assisted Interventation—MICCAI'98*, pages 1230–1241. Springer, 1998.

[101] J. Margeta, A. Criminisi, R. Cabrera Lozoya, D. C. Lee, and N. Ayache. Fine-tuned convolutional neural nets for cardiac mri acquisition plane recognition. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1–11, 2015.

[102] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

[103] R. Meier, S. Bauer, J. Slotboom, R. Wiest, and M. Reyes. A hybrid model for multimodal brain tumor segmentation. *Multimodal Brain Tumor Segmentation*, page 31, 2013.

[104] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *Medical Imaging, IEEE Transactions on*, 34(10):1993–2024, 2015.

[105] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[106] B. Morel. Fourier transformation and data processing. chapter 4, August 2011.

[107] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015.

[108] K. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[109] J. Nie, Z. Xue, T. Liu, G. S. Young, K. Setayesh, L. Guo, and S. T. Wong. Automated brain tumor segmentation using spatial accuracy-weighted hidden markov random field. *Computerized Medical Imaging and Graphics*, 33(6):431–441, 2009.

[110] C. D. N.Tustison, M. Wintermark and B. Avants. Ants and árboles. In *in proc of BRATS Challenge - MICCAI*, 2013.

[111] S. Parisot, H. Duffau, S. Chemouny, and N. Paragios. Joint tumor segmentation and dense deformable registration of brain mr images. In *MICCAI*, volume 7511, pages 651–658, 2012.

[112] S. Pereira, A. Pinto, V. Alves, and C. A. Silva. Deep convolutional neural networks for the segmentation of gliomas in multi-sequence mri. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 131–143. Springer, 2015.

[113] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *Proceedings of The 31st International Conference on Machine Learning*, pages 82–90, 2014.

[114] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Citeseer, 1999.

[115] K. Popuri, D. Cobzas, A. Murtha, and M. Jägersand. 3d variational brain tumor segmentation using dirichlet priors on a clustered feature set. *Int. J. Computer Assisted Radiology and Surgery*, 7(4):493–506, 2012.

[116] K. Popuri, D. Cobzas, M. Jagersand, S. L. Shah, and A. Murtha. 3d variational brain tumor segmentation on a clustered feature set. In *SPIE Medical Imaging*, pages 72591N–72591N. International Society for Optics and Photonics, 2009.

[117] M. Prastawa, E. Bullit, S. Ho, and G. Gerig. A brain tumor segmentation framework based on outlier detection. *Medical Image Anaylsis*, 8(3):275–283, 2004.

[118] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. Robust estimation for brain tumor segmentation. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pages 530–537. Springer, 2003.

[119] M. Prastawa, E. Bullitt, N. Moon, K. Van Leemput, and G. Gerig. Automatic brain tumor segmentation by subject specific modification of atlas priors< sup> 1</sup>. *Academic Radiology*, 10(12):1341–1348, 2003.

[120] J. Putaala, M. Kurkinen, V. Tarvos, O. Salonen, M. Kaste, and T. Tatlisumak. Silent brain infarcts and leukoaraiosis in young adults with first-ever ischemic stroke. *Neurology*, 72(21):1823–1829, 2009.

[121] V. Rao, M. Shari Sarabi, and A. Jaiswal. Brain tumor segmentation with deep learning. *MICCAI BraTS (Brain Tumor Segmentation) Challenge. Proceedings, winning contribution*, pages 31–35, 2014.

[122] J. Rexilius, H. K. Hahn, J. Klein, M. G. Lentschig, and H.-O. Peitgen. Multispectral brain tumor segmentation based on histogram model adaptation. In *Medical Imaging*, pages 65140V–65140V, 2007.

[123] S. Reza and K. Iftekharuddin. Multi-class abnormal brain tissue segmentation using texture features. In *in proc of BRATS Challenge - MICCAI*, 2013.

[124] J. P. Ridgway. Cardiovascular magnetic resonance physics for clinicians: part i. *Journal of cardiovascular magnetic resonance*, 12(1):1, 2010.

[125] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[126] R.Meier, S.Bauer, J.Slotboom, R.Wiest, and M.Reyes. Appearance- and context-sensitive features for brain tumor segmentation. In *in proc of BRATS Challenge - MICCAI*, 2014.

[127] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 234–241. Springer, 2015.

[128] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[129] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[130] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers. A new 2.5 d representation for lymph node detection using random sets of deep convolutional neural network observations. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pages 520–527. Springer, 2014.

[131] H. R. Roth, A. Farag, L. Lu, E. B. Turkbey, and R. M. Summers. Deep convolutional networks for pancreas segmentation in ct imaging. In *SPIE Medical Imaging*, pages 94131G–94131G. International Society for Optics and Photonics, 2015.

[132] H. R. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu, E. B. Turkbey, and R. M. Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 556–564. Springer, 2015.

[133] M. Rousson, T. Brox, and R. Deriche. Active unsupervised texture segmentation on a diffusion based feature space. In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, volume 2, pages II–699. IEEE, 2003.

[134] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

[135] B. N. Saha, N. Ray, R. Greiner, A. Murtha, and H. Zhang. Quick detection of brain tumors and edemas: A bounding box method using symmetry. *Computerized Medical Imaging and Graphics*, 36(2):95–107, 2012.

[136] T. Schlegl, J. Ofner, and G. Langs. Unsupervised pre-training across image domains improves lung tissue classification. In *Medical Computer Vision: Algorithms for Big Data*, pages 82–93. Springer, 2014.

[137] M. Schmidt, I. Levner, R. Greiner, A. Murtha, and A. Bistritz. Segmenting brain tumors using alignment-based features. In *Int. Conf on Machine Learning and Applications*, pages 6–pp, 2005.

[138] M. Schmidt. Automatic brain tumor segmentation. *Thesis*, 2005.

[139] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.

[140] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[141] A. W. Simonetti, W. J. Melssen, F. S. d. Edelenyi, J. J. van Asten, A. Heerschap, and L. Buydens. Combination of feature-reduced mr spectroscopic and mr imaging data for improved brain tumor classification. *NMR in Biomedicine*, 18(1): 34–43, 2005.

[142] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[143] J. G. Sled, A. P. Zijdenbos, and A. C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in MRI data. *IEEE Tr. on Medical Imaging*, 17(1):87–97, 1998.

[144] N. R. Smoll and K. J. Drummond. The incidence of medulloblastomas and primitive neurectodermal tumours in adults and children. *Journal of Clinical Neuroscience*, 19(11):1541–1544, 2012.

[145] J.-C. Souplet, C. Lebrun, N. Ayache, G. Malandain, et al. An automatic segmentation of t2-flair multiple sclerosis lesions. In *The MIDAS Journal-MS Lesion Segmentation (MICCAI 2008 Workshop)*. Citeseer, MICCAI, 2008.

[146] J. B. Springborg, L. Poulsgaard, and J. Thomsen. Nonvestibular schwannoma tumors in the cerebellopontine angle: a structured approach and management guidelines. *Skull Base*, 18(04):217–227, 2008.

[147] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[148] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learning Research*, 15(1):1929–1958, 2014.

[149] M. F. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in Neural Information Processing Systems*, pages 2980–2988, 2015.

[150] M. Styner, J. Lee, B. Chin, M. Chin, O. Commowick, H. Tran, S. Markovic-Plese, V. Jewells, and S. Warfield. 3D segmentation in the clinic: A grand challenge ii: Ms lesion segmentation. *MIDAS*, 2008:1–6, 2008.

[151] P. Su, Z. Xue, L. Chi, J. Yang, and S. T. Wong. Support vector machine (svm) active learning for automated glioblastoma segmentation. In *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*, pages 598–601. IEEE, 2012.

[152] N. Subbanna and T. Arbel. Probabilistic gabor and markov random fields segmentation of brain tumours in mri volumes. *Proc. Workshop on Brain Tumor Segmentation MICCAI.*, pages 47–50, 2012.

[153] N. Subbanna, D. Precup, L. Collins, and T. Arbel. Hierarchical probabilistic gabor and mrf segmentation of brain tumours in mri volumes. In *in proc of MICCAI*, volume 8149, pages 751–758, 2013.

[154] N. Subbanna, D. Precup, and T. Arbel. Iterative multilevel mrf leveraging context and voxel information for brain tumour segmentation in mri. In *in proc of CVPR*. IEEE, 2014.

[155] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proc. of the 30th int'l conf. on machine learning*, pages 1139–1147, 2013.

[156] N. Tajbakhsh, J. Shin, S. Gurudu, R. Hurst, C. Kendall, M. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Fine tuning or full training? *Medical Imaging, IEEE Transactions on*, 2016.

[157] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[158] G. Tulder and M. Bruijne. *MICCAI 2015 Proceedings*, chapter Why Does Synthesized Data Improve Multi-sequence Classification?, pages 531–538. Springer, 2015.

[159] N. Tustison, M. Wintermark, C. Durst, and B. Avants. Ants and arboles. *Proc. Workshop on Brain Tumor Segmentation MICCAI.*, pages 47–50, 2013.

[160] N. J. Tustison, K. Shrinidhi, M. Wintermark, C. R. Durst, B. M. Kandel, J. C. Gee, M. C. Grossman, and B. B. Avants. Optimal symmetric multimodal templates and concatenated random forests for supervised brain tumor segmentation with ANTsR. *Neuroinformatics*, 13(2):209–225, 2015.

[161] G. Urban, M. Bendszus, F. Hamprecht, and J. Kleesiek. Multi-modal brain tumor segmentation using deep convolutional neural networks. *in proc of BRATS-MICCAI*, 2014.

[162] K. Vaidhya, S. Thirunavukkarasu, V. Alex, and G. Krishnamurthi. Multi-modal brain tumor segmentation using stacked denoising autoencoders. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 181–194. Springer, 2015.

[163] S. Vaidya, A. Chunduru, R. Muthuganapathy, and G. Krishnamurthi. Longitudinal multiple sclerosis lesion segmentation using 3d convolutional neural networks. *proc. of THE 2015 LONGITUDINAL MS LESION SEGMENTATION CHALLENGE*, 2015.

[164] M. Vaidyanathan, L. Clarke, R. Velthuizen, S. Phuphanich, A. Bensaid, L. Hall, J. Bezdek, H. Greenberg, A. Trotti, and M. Silbiger. Comparison of supervised mri segmentation methods for tumor volume determination during therapy. *Magnetic resonance imaging*, 13(5):719–728, 1995.

[165] S. Van Buuren. *Flexible imputation of missing data*. CRC press, 2012.

[166] B. van Ginneken, A. A. Setio, C. Jacobs, and F. Ciompi. Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans. In *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, pages 286–289. IEEE, 2015.

[167] G. Van Tulder and M. de Bruijne. Why does synthesized data improve multi-sequence classification? In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pages 531–538. Springer, 2015.

[168] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. 25th int'l conf. on machine learning*, pages 1096–1103, 2008.

[169] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[170] S. Vinitski, C. F. Gonzalez, R. Knobler, D. Andrews, T. Iwanaga, and M. Curtis. Fast tissue segmentation based on a 4d feature map in characterization of intracranial lesions. *Journal of Magnetic Resonance Imaging*, 9(6):768–776, 1999.

[171] VSD. Virtual skeleton database. http://www.virtualskeleton.ch/, 2013.

[172] T. Wang, I. Cheng, and A. Basu. Fluid vector flow and applications in brain tumor segmentation. *IEEE Trans. Biomedical Eng.*, 56(3):781–789, 2009.

[173] E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 583–591. Morgan Kaufmann Publishers Inc., 2002.

[174] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. ICML*, pages 2048–2057, 2015.

[175] A. Yezzi, L. Zöllei, and T. Kapur. A variational framework for integrating segmentation and registration through active contours. *Medical Image Analysis*, 7 (2):171–185, 2003.

[176] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[177] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[178] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[179] J. Zhang, K.-K. Ma, M.-H. Er, V. Chong, et al. Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine. In *International Workshop on Advanced Image Technology (IWAIT'04)*, pages 207–211, 2004.

[180] L. Zhao, D.Sarikaya, and J. Corso. Automatic brain tumor segmentation with mef on supervoxels. *Proc. Workshop on Brain Tumor Segmentation MICCAI.*, pages 51–54, 2013.

[181] L. Zhao, W. Wu, and J. J. Corso. Brain tumor segmentation based on gmm and active contour method with a model-aware edge map. *BRATS MICCAI*, pages 19–23, 2012.

[182] L. Zhao, W. Wu, and J. J. Corso. *MICCAI 2013*, chapter Semi-automatic Brain Tumor Segmentation by Constrained MRFs Using Structural Trajectories, pages 567–575. Springer, 2013.

[183] D. Zikic, Y. Ioannou, M. Brown, and A. Criminisi. Segmentation of brain tumor tissues with convolutional neural networks. *in proc of BRATS-MICCAI*, 2014.

[184] D. Zikic, B. Glocker, E. Konukoglu, A. Criminisi, C. Demiralp, J. Shotton, O. Thomas, T. Das, R. Jena, and S. Price. Decision forests for tissue-specific segmentation of high-grade gliomas in multi-channel mr. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012*, pages 369–376. Springer, 2012.