# CNN for Brain Tumor Segmentation

A Dissertation Report

Submitted by

**Mohit Singhaniya**
**CS1516**

in partial fulfillment for the award of the degree of

**Master of Technology**

in

**Computer Science**

**Indian Statistical Institute, Kolkata, India**

July 2017

## BONAFIDE CERTIFICATE

This is to certify that the thesis titled "**CNN for Brain Tumor Segmentation**" submitted by **Mr. Mohit Singhaniya (CS1516)** to the Indian Statistical Institute, Kolkata, for the award of **Master of Technology in Computer Science,** is a *bona fide* record of the project work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Prof. Sushmita Mitra**

Project Guide

Machine Intelligence Unit (MIU)
Indian Statistical Institute
Kolkata 700 108
India

Place: Kolkata

Date:

## ACKNOWLEDGEMENTS

The completion of this project could not have been possible without the guidance of **Prof. Sushmita Mitra** and **Mr. Subhashis Banerjee.** I would like to thank them for their motivation, guidance and assistance during the entire project.

I would like to thank all my friends especially Shaunak Gupta and Rajeev Baditha who helped me during my project. I would also like to thank my family members who supported me morally, financially and physically.

# ABSTRACT

Among brain tumors, gliomas is the most aggressive and common, leading to a very short life expectancy in their highest grade. MRI (Magnetic Resonance Imaging) is a widely used imaging technique to access such tumors but the amount of data produced by MRI is huge which prevents manual segmentation in a reasonable amount of time. So, automatic and reliable methods are required, but the variation in the structure and location of such tumors makes automatic segmentation a very challenging task. In this report, we have proposed four different methods for extracting patches which can be used to train Convolution Neural Networks (CNN) to do the automatic segmentation of tumor in the HGG (Higher Grade Gliomas) and the LGG (Lower Grade Gliomas) patients. We have also proposed a Convolution Neural Network (CNN) based on Transfer Learning which does automatic segmentation in a reasonable amount of time with promising results for the LGG patients.

# Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

A brain tumor or intracranial neoplasm occurs when abnormal cells form within the brain. The symptoms may include headache, vomiting, problem with vision and mental changes. More specific problems may include difficulty in walking, speaking and with sensation. As the disease progresses unconsciousness may occur. The cause of brain tumor is not known. An example of brain tumor is shown in the figure below.

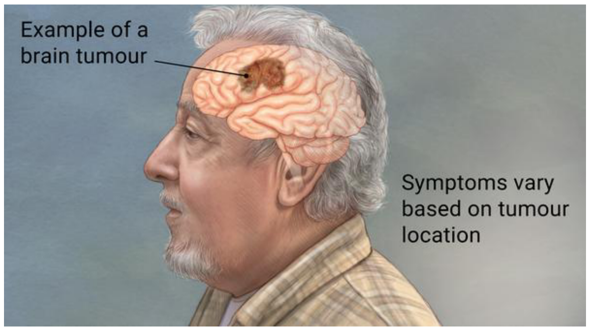

Example of a brain tumour
Symptoms vary based on tumour location

Figure 1.1 Brain Image

Among the several brain tumors in human being, gliomas is the most common and aggressive. Based on the nature of the tumor, gliomas can be classified into two broad categories:

1. Higher Grade Gliomas (HGG)
2. Lower Grade Gliomas (LGG)

where LGG is less aggressive than HGG [14]. Presently, surgery, radiotherapy, chemotherapy or a combination of them is used for the treatment of the patients. MRI (Magnetic Resonance Imaging) is a widely used imaging technique for detecting such tumors. The accurate segmentation of this tumor is very important for planning the treatment and also for further evaluations. But, manual segmentation is very time consuming since the data produced by MRI is huge. For this reason, generally physicians use rough measures to evaluate such tumors [13]. Also, the segmentation of brain tumor done by different expert raters using the MR images varies. There are variations in the intra-tumoral structure, shape and sometimes in the location of the tumor reported by different expert physicians. So, we need some automatic methods for doing the segmentation accurately in a small amount of time. But, since such tumors are highly variable in their shape, size, structure and locality, so automatic segmentation of such tumors is not an easy task.

In the field of brain tumor segmentation, recent proposals also investigate the use of CNNs. Zikic et al. [17] used a shallow CNN with two convolution layers separated by a max pooling layer with stride 3, followed by a fully connected layer and a softmax layer. Urban et al. [18] evaluated the use of 3D filters, although the majority of authors opted for 2D filters. 3D filters can take the advantage of the 3D nature of the images, but it increases the computational load. Some proposals evaluated two-pathway networks to allow one of the branches to receive bigger patches and than the other, thus having a larger context view over the image. In addition to their two-pathway network, Havaei et al. [19] built a cascade of two networks and performed two-stage training, by training with balanced classes and then refining it with proportions near the originals [11]. The deep learning models proposed by Pereira et al. [11] are the only deep learning models for brain tumor segmentation to the best of our knowledge. Pereira et al. have proposed two different CNN one for the HGG patients and the other for the LGG patients. Their CNN for the HGG patients consists of 3 convolution layer followed by a max pooling layer which is again followed by 3 convolution and a max pooling layer and finally 3 FC layers, while their CNN for LGG patients consists of 2 convolution layer followed by a max pooling layer which is again followed by 2 convolution and a max pooling layer and finally 3 FC layers. Our work on HGG patients is inspired from their model on HGG patients.

Also, training a deep convolution neural network (CNN) from scratch could be difficult as it requires a large set of labelled data for training. An alternative to this problem is to fine-tune a CNN which has been pre-trained using, for example, a large set of labelled natural images. Although the difference between natural images and medical images may advice against such fine-tuning or transfer. But, Nima Tajbakhsh et al. [16] by performing several experiments have shown that a pre-trained CNN with adequate fine-tuning always outperforms or in the worst case performs as well as a CNN trained from scratch.

## 1.1 Motivation

Since the amount of time required for manually segmenting the MRI is huge and since gliomas is a type of tumor which requires treatment as soon as it is detected in the patients, so automatic methods are required to do the segmentation in a very small amount of time. It is also found that the segmentation of tumor using MRI's by experts varies, so, some automatic segmentation method is required to do the segmentation accurately. Also, an application such as brain tumor leaves no room for error. In a surgery it is important to remove as much as tumor as possible without damaging any healthy tissues and for that we have to precisely identify the location and shape of the tumor.

## 1.2 Contributions of the Thesis

In this thesis, we will present several different patch extracting algorithms proposed by us. We will also present a CNN for the HGG patients which is inspired by the ground breaking work of Pereira et al. [11]. Also, we will present the proposed CNN based on transfer learning for the LGG patients, which can do the segmentation in a very small amount of time, achieving high dice scores. We will also show how transfer learning has been used for improving the results in case of LGG patients.

## 1.3 Organisation of the thesis

In chapter 2, we have talked about neural networks, CNN, MRI, deep learning and the challenges in brain tumor segmentation. It also discusses about the digital representation of tumor and the evaluation technique used for evaluating different brain tumor segmentation algorithms along with the dataset that has been used for the project. The model designed for automatic segmentation of brain tumor in HGG and LGG patients is presented in chapter 3. The different proposed algorithms for patch extraction and the use of transfer learning for segmentation of tumor in LGG patients are also explained in this chapter. The results of the proposed model are presented in chapter 4 and finally, we conclude with chapter 5.

# Chapter 2 Literature Review

In this chapter, we will discuss some of the concepts which are required for further explanation of the work. The concepts of neural network such as multilayer perceptron and Convolution Neural Network (CNN) are discussed in this chapter. The challenges involved in brain tumor segmentation and the representation of tumor is also discussed in this chapter. The concept of Dice score which is used for the evaluation of different brain tumor segmentation algorithms, concept of MR images, Deep Learning, Imagenet and the dataset used for training the model is also explained in this chapter.

## 2.1 Neural Network

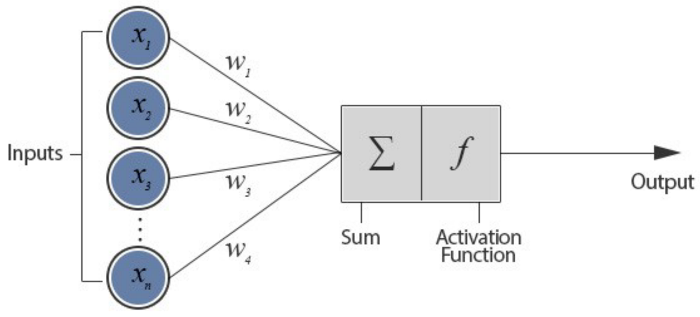


Figure 2.1 Simple Neural Network

The figure depicts a neuron connected with n other neurons and thus receives n inputs ($x_1$, $x_2$, ...., $x_n$). This configuration is called a Perceptron. The input ($x_1$, $x_2$, ...., $x_n$) and weights ($w_1$, $w_2$, ...., $w_n$) are real numbers and can be positive or negative. All the inputs are individually, added together and passes into the activation function. There are many different types of activation functions such as sigmoid, relu, etc which are used in practice. Input vectors from a training set are presented to the preceptron one after the other and weights are modified according to the following equation, W(i) = W(i) + a*g'(sum of all inputs)*(T-A)*P(i), where g' is the derivative of the activation function and 'a' is the learning rate of the perceptron, W is the weight vector, P is the input vector, T is the correct output that the perceptron should have known and A is the output given by the perceptron. When an entire pass through all the input training vectors is completed without an error, the perceptron has learnt. At this time, if an input vector which is present in the training set is given to the perceptron, it will output the correct value. Whereas, if P is not in the training set, the network will give an output similar to other training vectors close to P. The perceptron is adding all the inputs and separting then into 2 categories, those that cause it to fire and those that don't. That is, it is drawing the line $w_1x_1 + w_2x_2 = t$, where t is the threshold. Points on one side of the line fall into one category and the points on the other side fall into the other category. But, not every set of inputs can be divided by a line. One example of such a problem is the Boolean XOR problem. This is where multi-layered neural network come into picture.

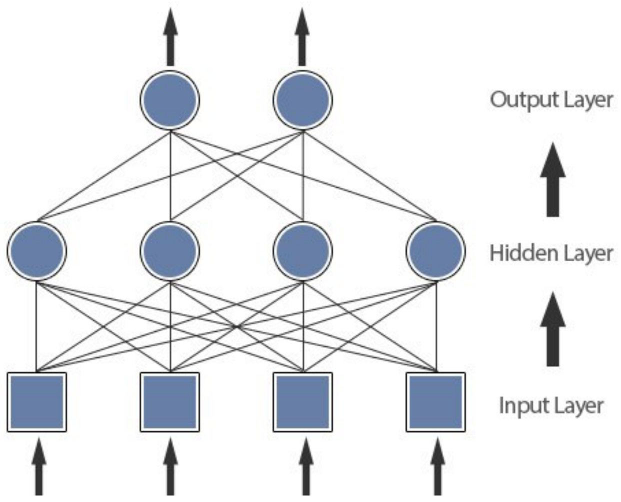


Figure 2.2 Multilayer perceptron

Each input from the input layer is fed to each node in the hidden layer, and from there to each node on the output layer. There can be any number of nodes per layer and there are usually multiple hidden layers to pass through before ultimately reaching the output layer.

To tune weights between the hidden layer and the input layer, we need to know the error at the hidden layer, but we know the error only at the output layer. So, to deal with this the errors at the output layer is taken and proportionally propagated backwards to the hidden layer. Below we have shown equations for a 2 layered network but the same concept will be applicable for a network having any numbers of layers.

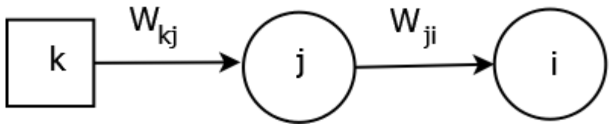


Figure 2.3 One segment of a 2 layered network

For a particular neuron, 'i' in the output layer weights will be adjusted as $W_{ji} = W_{ji} + a*g'$ (sum of all inputs)*(T-A)*P (j) for all j connected to 'i', where, g' is the derivative of the activation function and 'a' is the learning rate of the perceptron, W is the weight vector, P is the input vector, T is the correct output that the perceptron should have known and A is the output given by the perceptron. This equation tunes the weights between the output layer and the hidden layer.

Now, for a particular neuron 'j' in the hidden layer, we will propagate the error backwards from the output layer such that, Error = $W_{j1}xE_1 + W_{j2}xE_2 + ... + W_{jn}xE_n$. Therefore, for a particular neuron 'j' in the hidden layer $W_{kj} = W_{kj} + a*g'$ (sum of all inputs)*(T-A)*P (k) for all k connected to 'j', where, g' is the derivative of the activation function and 'a' is the learning rate of the perceptron, W is the weight vector, P is the input vector, T is the correct output that the perceptron should have known and A is the output given by the perceptron. This equation tunes the weights between the hidden layer and the input layer.

## 2.2 CNN (Convolution Neural Network)

CNN (Convolution Neural Network) is a feed-forward neural network and is widely used for image recognition. CNN, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. CNN operates over volumes, unlike neural networks, where the input is a vector but in case of CNN the input is a multi-channelled image.

### 2.2.1 Convolution Layer

The convolution layer is the main building block of a convolution neural network. The working of the convolution layer is explained below.

We define a filter (5x5x3 in this example) and slide it over the complete image and along the way take dot product between the filter and chunks of the input image to generate a new image. For every dot product taken, the result is a scalar (Each dot product gives a pixel of the new image). Filters always extend the full depth of the input volume.

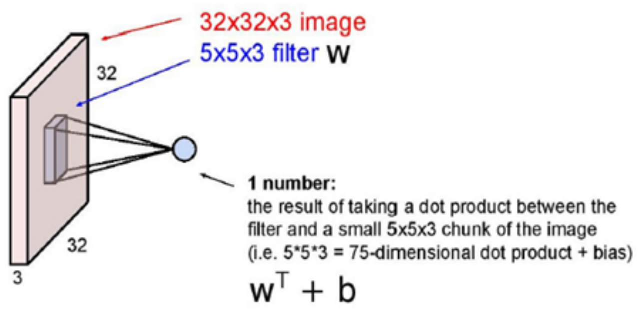

32x32x3 image
5x5x3 filter w
32
32
3
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T + b$

Figure 2.4 Convolution

So, after convolving the complete image with the filter we will get an output image (also called as the activation map) which is 28x28x1 for our considered example. The size of the output image can we computed using the simple formula (N-F)/stride + 1, where N is the length of the input, F is the length of the filter and stride is the sliding length. For the example shown below, N=32, F=5 and stride=1, therefore output length = (32-5)/1+1 = 28 i.e. the dimension of output image will be 28x28.

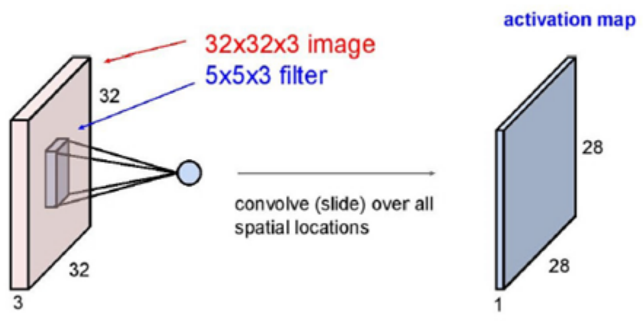


Figure 2.5 Convolution of an image

The convolution layer comprises a set of independent filters (6 in the example considered). Each filter is independently convolved with the image. In the example shown below, we end up with 6 feature maps of shape 28x28x1, which are then stacked to get a new image of size 28x28x6.

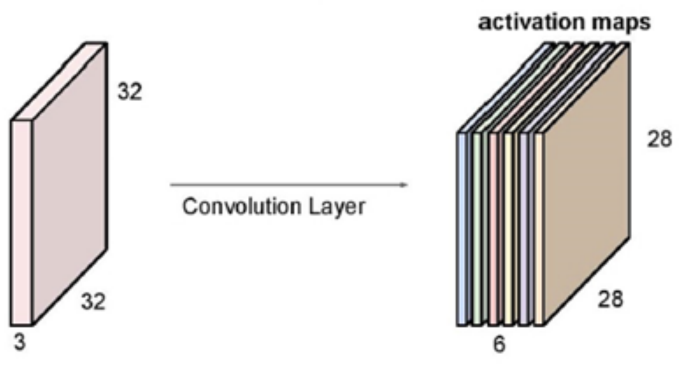


Figure 2.6 Convolution using 6 filters

Now, suppose if there are several convolution layers in sequential order then the convolution will happen as shown in the figure below.

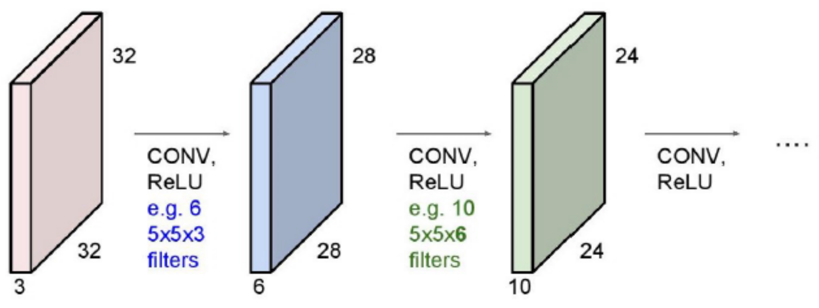


Figure 2.7 Multiple convolution layers in sequence

All these filters are initialised either randomly or in some other ways which become our parameters that are learned by the network subsequently.

## 2.2.2 Pooling Layer

A pooling layer is another building block of a CNN. It is common to periodically insert pooling layer in between successive convolution layers. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network and hence to also control overfitting.

**Overfitting:** One of the problems that occur during neural network training is overfitting. When the error on the training data is driven to a very small value, but the error of the network when presented with new data is large, such a situation is known as overfitting. In overfitting, the network basically memorizes the training examples, but it does not learn to generalize to new situations.

In pooling layer, we slide the filter over the complete image and generate an array of scalars according to the type of the filter. Every scalar represents a pixel of the output/new image.

Pooling layer operates on each map independently. The most common approach used in pooling is max pooling. In the figure below, we have shown a max pool with 2x2 filters and stride 2 being applied to an image. The dimension of the output image of the max pooling layer can be computed using the formula (N-F)/stride + 1, where N is the length of the input, F is the length of the filter and stride is the sliding length. For the example shown below, N=4, F=2 and stride=2, therefore output length = (4-2)/2+1 = 2 i.e. the dimension of output image will be 2x2.

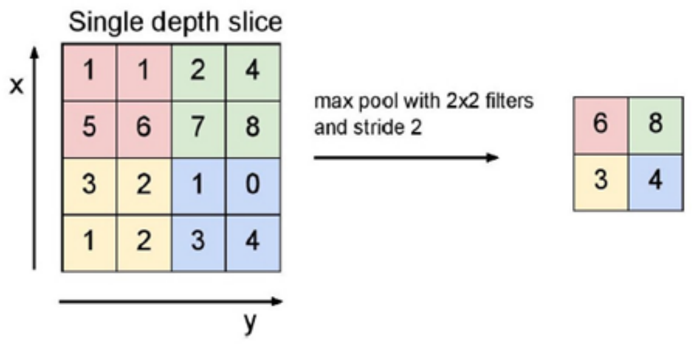


Figure 2.8 Max-pooling

### 2.2.3 FC Layer

A CNN also consists of fully connected layer/layers of neuron at the end. Neurons in fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks and work in a similar way.

## 2.3 Why are we using CNN?

The number of parameters to train a fully connected network is much larger as compared to a CNN. Also, in most images the nearby pixels are generally related but a fully connected neural network doesn't take this into account, while a CNN takes into account our understanding of the relationship between space and pixels within an image.

## 2.4 MRI (Magnetic Resonance Imaging)

MRI (Magnetic Resonance Imaging) provides good contrast for soft tissues as compared to the other imaging techniques available. Also, there are no known health hazards from temporary exposure to the MR environment. So, MRI is a good technique for accessing brain tumor in human beings. Different imaging modalities can be used for mapping tumor-induced changes, including T1, T1c, T2, and Flair MRI [10]. These MRI modalities are sensitive to the inflammatory and demyelinating changes directly associated with the underlying pathology. 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. Using Flair we can identify the whole tumor structure. In the dataset, for each patient, images from four different sequences namely: T1, T1-contrasted, T2 and FLAIR (Fluid Attenuated Inversion Recovery) are acquired. Each of these sequences exploits the distinct characteristics of the tissues as explained above which results in contrast among the images. We can observe the same from the figures given below:

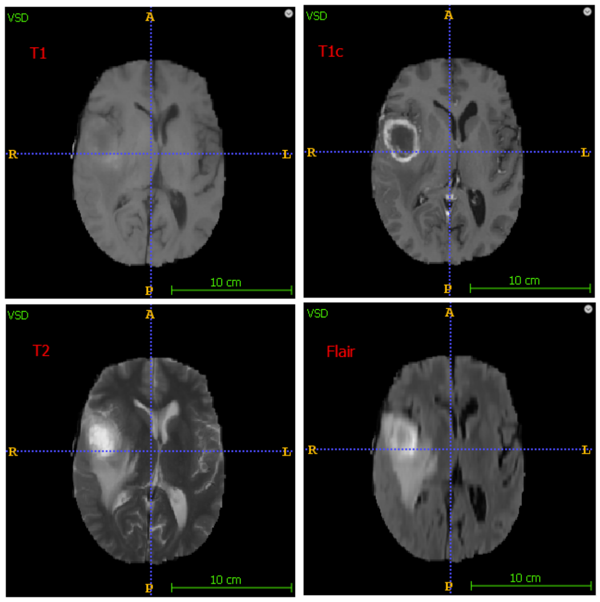


Figure 2.9 MR images

Notice, the difference in the intensities among the four images, all of which are images of the same brain slice taken with different MRI sequences.

### 2.4.1 Tumor structure and representation

The sub-regions of a brain tumor are: necrosis (dead part of the tumor), edema (swelling caused by tumor), enhancing (refers to the part of the tumor which is enhanced in T1c modality), non-enhancing (refers to the part of the tumor which isnot enhanced in T1c modality).

Brain tumors are represented (digitally) using 5 levels (as shown in the figure below). The red part (represented as level 1) is the necrosis, the green part (represented as level 2) is the edema, the blue part (represented as level 3) is the non-enhancing tumor, the yellow part (represented as level 4) is the enhancing tumor and the remaining part of the brain i.e. the normal tissues and the black surrounding area is represented by level 0.

**Complete tumor:** The necrosis, edema, non-enhancing and enhancing part of the tumor as a whole is known as the complete tumor i.e. (Level 1 + Level 2 + Level 3 + Level 4)

**Core tumor:** The necrosis, non-enhancing and the enhancing part of the tumor as a whole is known as the core tumor i.e. (Level 1 + Level 3 + Level 4)

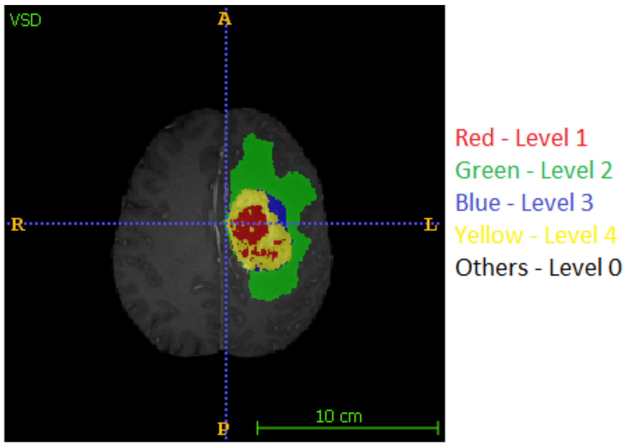


Figure 2.10 Tumor representation

### 2.4.2 MRI sequences

Different MRI sequences exploit distinct characteristics of the tissues which result in contrast between the images, which help for better segmentation. As, we can see in the figure below that the whole tumor is visible clearly in the sequence Flair (i.e. figure A) while the core of the tumor is visible in the sequence T2 (i.e. figure B) and the enhancing tumor structure is visible in the sequence T1c (i.e. figure C). This is why; we need MR images from different sequences so that we can identify/classify the intra-tumoral structure precisely.

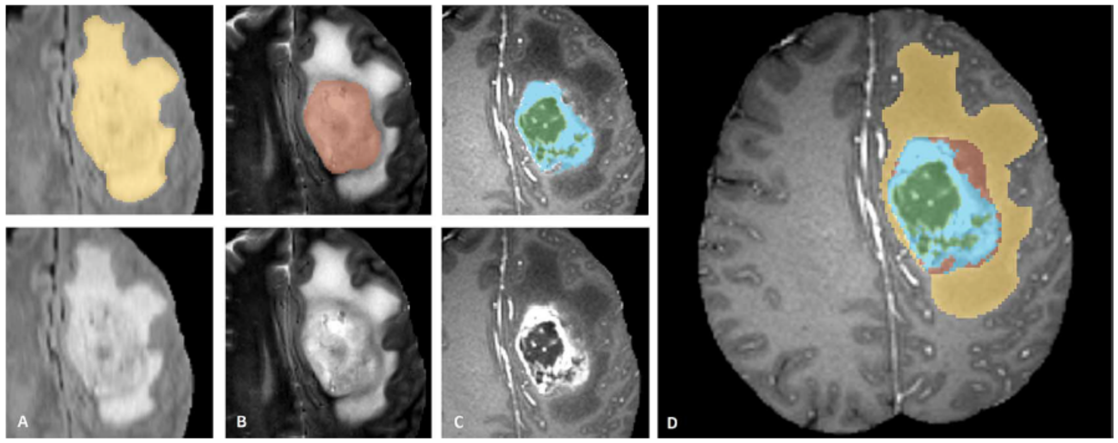


Figure 2.11 Different MRI sequences helping to identify intra-tumoral structure

## 2.5 Imagenet challenge and Deep learning

In, deep learning a hierarchy of increasingly complex features is directly learned from the data. So, the focus is on designing architectures instead of developing hand-crafted features, which may require specialised knowledge. This is one of the major reasons for using deep learning for solving problems as feature extraction may be complicated in some cases.

In Imagenet challenge, for a given image, the task is to produce 5 class labels in decreasing order of confidence and 5 bounding boxes, one for each class label. There are 1000 categories in this challenge. The quality of localisation labelling will be evaluated based on the label that best matches the ground truth label for the image and also the bounding box that overlaps with the ground truth. A training dataset of 1.2 million images along with their ground truth is provided. Testing and validation data consists of 1,50,000 images [20].

Now that we have understood about the Imagenet [5] challenge, let us discuss more about it. For this challenge, deep learning was first used in the year 2012. The error rate of the best method of 2011 was 26%. But, in 2012 a method based on deep learning ranked no.1 in the Imagenet challenge and its error rate was just 16%. The best method of 2012 used 4 GPU's for solving the problem. In 2013, another deep learning method ranked no.1 and the error rate was decreased to 12% which was further decreased to 7% by another deep learning method proposed in 2014. Thus, we can see (refer figure) that the error rate has significantly decreased with deep learning since 2012. So, inspired by this, we are also trying to explore deep learning in the field of medical image processing.

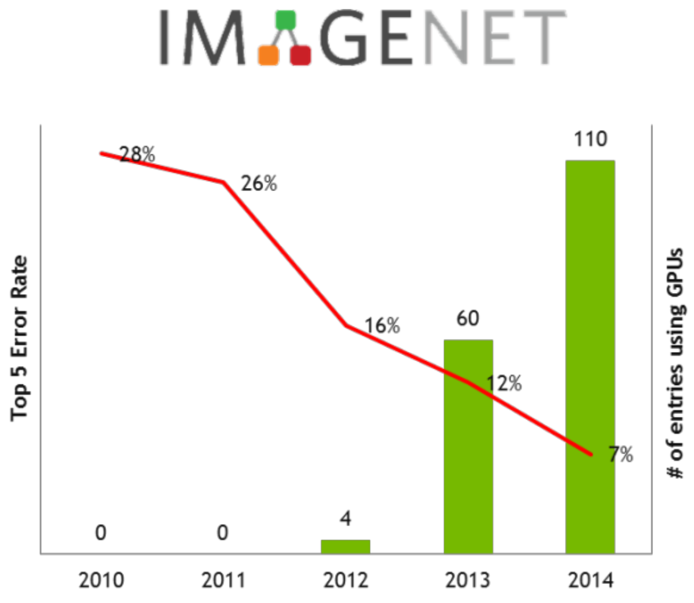


Figure 2.12 Imagenet challenge results

## 2.6 Challenges in tumor classification

Identification of tumor is a very challenging task. The location, shape and the structure of tumor varies significantly from patient to patient which makes the segmentation a very challenging task. In the figure shown below, we have shown some images of the same brain slice from different patients, which clearly reflect the variation of tumor. We can clearly see that the location of tumor is different in all the 8 images/patients shown below. To make it worse, the shape and the intra-tumoral structure is also different for all the 8 patients/images. In fact, there can be more than one region of tumor as can be seen from the images below. This indeed reflects the complexity of automatic segmentation.

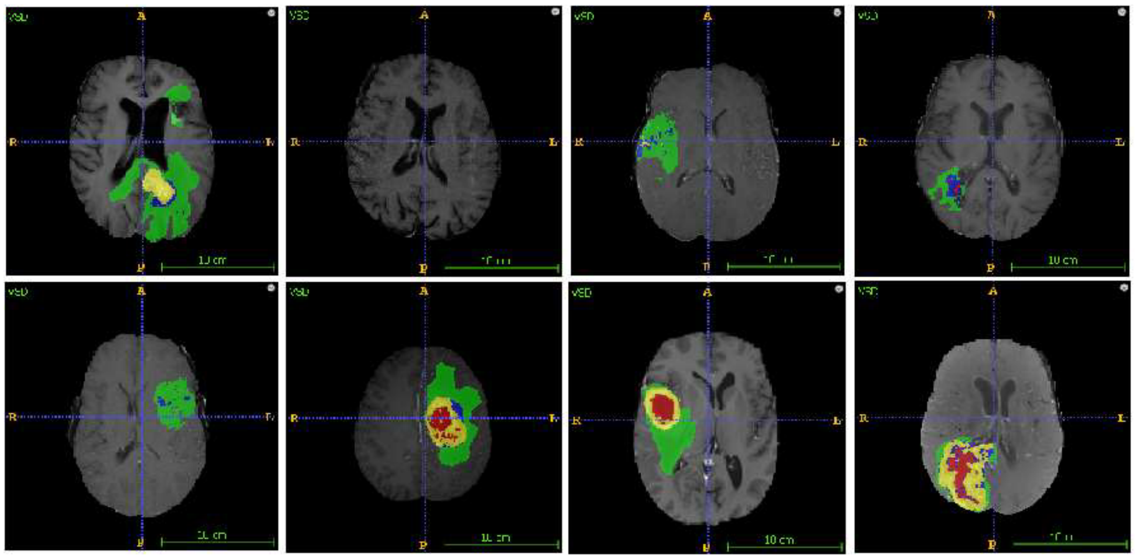


Figure 2.13 Tumor in different patients of the same brain slice

## 2.7 Dice score

The evaluation of the segmentation technique/algorithm is done by calculating the DSC i.e. Dice Similarity Coefficient. The DSC measures the overlap between the manual and the automatic segmentation. DSC is defined as follows [6]:

$$DSC = \frac{2\,TP}{FP + 2\,TP + FN}$$

Here, TP represents true positive, FP represents false positive and FN represents false negative. For comparing different algorithms (segmentation techniques) we measure the dice score in the following cases:

1. Complete tumor (Level 1 + Level 2 + Level 3 + Level 4)
2. Core tumor (Level 1 + Level 3 + Level 4)
3. Enhancing tumor (Level 4)

## 2.8 Dataset

All MRI data are provided by BraTS Challenge 2015 [7], which consists of data for 220 high-grade glioma patients and 54 low-grade glioma patients. Images of four different MRI sequence namely. T1, T1-c, T2 and Flair are provided for each patient along with the professional segmentation as ground truth labels for each case. Each brain scan consists of 155 slices of size 240x240. Each pixel in the MR images represents a $1mm^3$ voxel. MRI's are stored as a 3d array of size 155x240x240. This array is generated by scanning the brain from top to bottom. In the ground truth images, necrosis is represented by integer value 1, edema by 2, non-enhancing tumor by 3, enhancing tumor by 4 and others are represented as 0.

# Chapter 3 Model Design

In this chapter we will discuss about the pre-processing of MR images and the importance of patch extraction. We will also present four different patch extraction methods proposed by us and using the best method we will build a CNN for the HGG patients which will further be used to build another CNN for the LGG patients via transfer learning. Concepts such as balancing classes and real time data augmentation which has helped in improving the accuracies are also discussed here. Finally, we conclude the chapter by showing how segmentation is done for HGG and LGG patients and some post-processing measures.

We have approached the problem of brain tumor segmentation using CNN. We train the CNN on patches, a patch is a sub-image of the original image. The purpose of training the model on patches is due to the fact that the class of any given voxel is highly dependent on the class of its surrounding voxels. The class of the centre pixel of a patch is considered as the class of the patch. Through testing it has been found that the model gives best results when the patch size is taken as 33x33 [11].

## 3.1 Pre-processing

One of the challenging task in dealing with the MRI data is dealing with the artifacts produced either by in homogeneity in the magnetic field or small movements made by the patient during scan time [4]. Often a bias is present across the patients which make the segmentation difficult for an automatic segmenting model.

Since Bias field distortion alters the MR images, therefore the intensity of the same tissue may vary in an image/slice. So, we have applied the N4ITK method [2] to correct this error. However this is not enough for making the intensity of the same type of tissues similar across multiple patients. As a matter of fact, the same tissue of the same patient can be in different intensity when taking the images/MRI at different times. So to make the intensity of the same tissue type more similar we have used the method proposed by Nyul et al. [1] [3].

### 3.1.1 Intensity Normalisation

Nyul et al. [1] [3] has proposed a two step method for normalising the images across multiple patients. The first step is the training step which is executed only once for a given dataset and the second step is the transformation step which is executed for each image in the dataset. The main idea underlying the methods is to deform the image histograms so that they match a mean histogram determined through training. The actual matching is based on certain landmarks identified on the histograms.

#### 3.1.1.1 Training

In the training phase, the landmarks ($p_{1j}$, $p_{2j}$, $\mu_{1j}$, $\mu_{2j}$,..., $\mu_{9j}$) is obtained from each image and is mapped to the standard scale by mapping the intensities from [$p_{1j}$, $p_{2j}$] onto [$s_1$, $s_2$] linearly. Then for each k, $1 \leq k \leq 9$, the mean $\mu_{ks}$ of the mapped $\mu_{kj}$'s over the training images is computed.

```
Algorithm: Training step

Input: A set of images that is a subset of the training dataset, histogram parameters $pc_1$, $pc_2$ and $s_1$, $s_2$

Output: $\{\mu_{ks}\ 1 \leq k \leq 9\}$

begin

1: For each image

2: Compute the histogram $H_j$ for the image

3: Determine the intensity values $p_{1j}$ and $p_{2j}$ corresponding to $pc_1$ and $pc_2$ and the landmark locations $\mu_{1j,}\ \mu_{2j}$, ....., $\mu_{9j}$ on $H_j$

4: Map [$p_{1j}$, $p_{2j}$] of $H_j$ onto [$s_1$, $s_2$] linearly

5: Find the new mapped landmark locations $\mu'_{1j}$, $\mu'_{2j}$, ....., $\mu'_{9j}$

6: end for

7: Calculate the rounded means $\mu_{1s}$, $\mu_{2s}$, ......, $\mu_{9s}$ of $\mu'_{1j}$, $\mu'_{2j}$, ....., $\mu'_{9j}$ respectively over j = 1, 2, ......, N

end
```

### 3.1.1.2 Transformation

In this phase, for any given image, the actual landmark locations $\mu_{ki}$ obtained from its histogram are matched to $\mu_{ks}$ by doing several separate linear mappings: the first from $[p_{1i}, \mu_{1i}]$ to $[s_1, \mu_{1s}]$; the second from $[\mu_{1i}, \mu_{2i}]$ to $[\mu_{1s}, \mu_{2s}]$,..., and the last from $[\mu_{9i}, p_{2i}]$ to $[\mu_{9s}, s_2]$.

```
Algorithm: Transformation step

Input: Image to transform, $pc_1$, $pc_2$ and $s_1$, $s_2$, $\mu_{1s}$, $\mu_{2s}$, ......, $\mu_{9s}$

Output: The transformed Image

begin

1: Compute the histogram $H_j$ of the image

2: Determine intensity values $p_{1i}$ and $p_{2i}$ corresponding to $pc_1$ and $pc_2$ and the landmark
   locations $\mu_{1i}$, $\mu_{2i}$, ......, $\mu_{9i}$ on $H_j$

3: Map sections of the scale of $H_j$ linearly to the standard scale

4: Map the intensity value of every voxel according to the mapping obtained from step 3 to
   get the output image

end
```

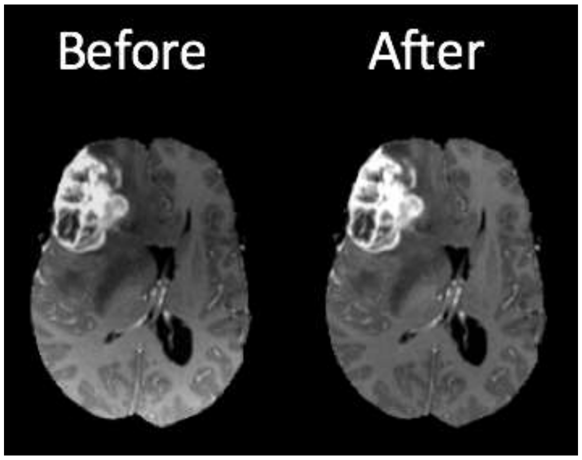


Figure 3.1 Before and after pre-processing

## 3.2 Patch extraction

A patch is a sub-image of the original image. In the figure show below, we have shown an example of a patch of size 33x33 taken from a brain slice image of size 240x240.

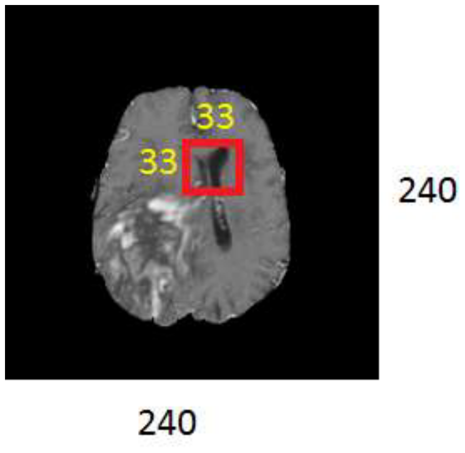


Figure 3.2 Showing a Patch

### 3.2.1 Importance of using patches

Since the size of each slice is 240x240, therefore if we train the CNN on the whole image/slice then the number of parameters to train will be very large and thus we will need a very large amount of data. But, since the dataset is not very large, so, we train our model using patches. Also, the class of any given voxel is highly dependent on the class of its surrounding voxels. So, we are training our model using patches. And by testing different patch sizes we have found that 33x33 works the best for the dataset that we are using.

### 3.2.2 Importance of patch extraction

The segmentation accuracy is greatly influenced by the patches being used for training the model. We have proposed four different algorithms for patch extraction and the segmentation results were significantly different for different algorithms. A comparison of the four methods is shown in the figure below.

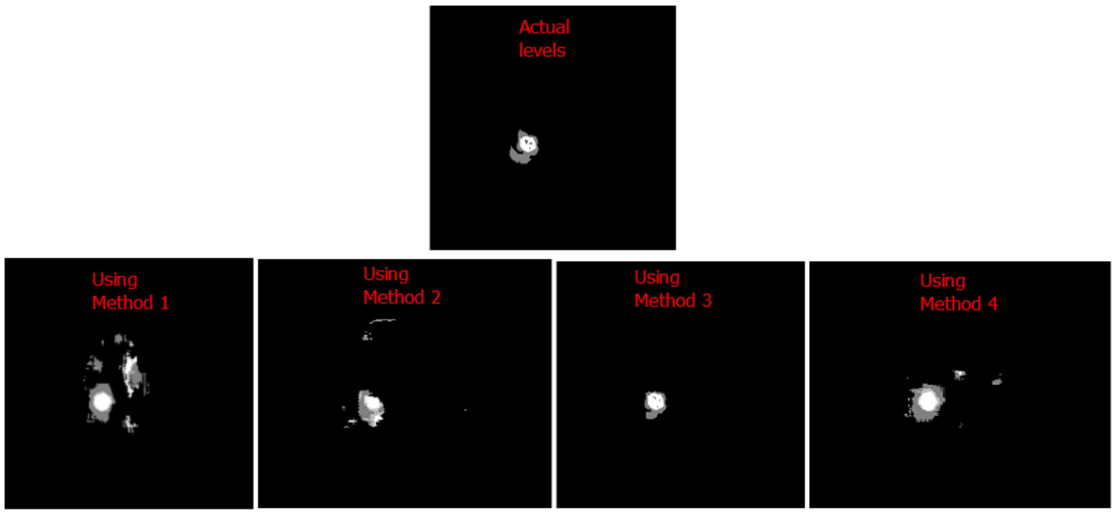


Figure 3.3 Segmentation results using different patch extraction methods

So, we can clearly see that the segmentation of tumor is best using method 3. Based on the segmentation results the algorithms could be ranked in the order (best to worst) as: method 3, method 4, method 2, and method1. So, patch extraction is a very important problem as it greatly influences the results.

### 3.2.3 Proposed Patch extraction methods

The importance of patch extraction has been clearly explained. We have proposed 4 different algorithms for patch extraction which took different time to run and which also gave quite different segmentation accuracies.

After several testing we have found that for extracting background patches the threshold should be 75 percent non zero intensity pixels (i.e. the patch should contain at least 75 percent non zero values) otherwise most of the healthy tissue patches are classified as tumor. Also, when those 75 percent non zero intensity pixels were only from healthy tissue pixels the misclassification error for tumor was found to be maximum. And when we allowed a maximum of 25 percent tumor pixels in the patches for healthy tissues then the accuracy was found to be maximum. And for the tumor patches, the threshold is 75 percent non zero intensity pixels. In case of tumor patches those 75 percent non zero pixels can also include healthy tissue pixels, otherwise the boundary of the tumor region will not be correctly identified/segmented.

This observation has been kept in mind/used while coming up with all the four algorithms/methods. Patches for all the methods are extracted along the plane perpendicular to the axial axis until and unless stated otherwise. Another thing that we kept in mind while proposing these methods was to balance the patches from all the classes as the training will not be good otherwise.

#### 3.2.3.1 Method 1

In this method, we generate patch for each of the voxel (not belonging to the black surrounding area) and the similar patches of the same class are removed.

Patch extraction: Method 1

```
Input: The training dataset
Output: A patch library
begin
1: for each voxel, v in the image
2: extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and append them to get the desired patch of size 4x33x33
3: find the class of the patch
4: check if the no. of non zero intensity pixel is > $threshold_{c,}$
where, $threshold_c$ is the threshold for class c
5: check if the patch is matching with any of the previously extracted patches of the same class
6: if no, add the patch to the patch library along with the class
end
```

Since, the number of patches extracted from this method was huge. So, we incorporated another condition which was to select only the highest entropy patches.

Algorithm: selecting only the best patches

```
begin
1: put all the patches of a class in a list
2: find the entropy of all the patches
3: sort the lists
4: select only the required no. of patches from the sorted lists
end
```

The time required for finding patches for HGG and LGG patients was approximately 300 hrs i.e. 12.5 days using this method. Overfitting occured when training the HGG and LGG models using the patches extracted from this method. Also, the segmentation accuracy for the complete tumor was only about 62% in the HGG patients using this method.

#### 3.2.3.2 Method 2

In this method, we generate patch for each of the voxel (not belonging to the black surrounding area) and if the patch is similar to the previously extracted patch of the same class then it is not included in the patch library.

Patch extraction: Method 2

```
Input: The training dataset
Output: A patch library
begin
1: for each voxel, v in the image
2: extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and
   append them to get the desired patch of size 4x33x33
3: find the class of the patch
4: check if the no. of non zero intensity pixel is > $threshold_{c,}$
   where, $threshold_c$ is the threshold for class c
5: check if the patch is matching with the previously extracted patch of the same class
```

```
6: if no, add the patch to the patch library along with the class
end
```

Since, the number of patches extracted from this method was again huge. So, we incorporated the same condition as in method 1 of selecting the highest entropy patches.

Algorithm: selecting only the best patches

```
begin
1: put all the patches of a class in a list
2: find the entropy of all the patches
3: sort the lists
4: select only the required no. of patches from the sorted lists
end
```

The time required for finding patches for HGG and LGG patients was approximately 60 hrs i.e. 2.5 days using this method. Overfitting occured when training the HGG and LGG models using the patches extracted from this method. Also, the segmentation accuracy for the complete tumor was only about 67% in the HGG patients using this method.

#### 3.2.3.3 Method 3

In this method, once a patch is extracted for a voxel of a slice, a box of size (NxN) is created at that pixel and patches of the same class are not extracted from that box. This helps to prevent overfitting.

Patch extraction: Method 3

```
Input: The training dataset
Output: A patch library
begin
1: for each voxel, v in the image
```

```
2: extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and append them to get the desired patch of size 4x33x33
3: find the class of the patch
4: check if the no. of non zero intensity pixel is > $threshold_c$,
where, $threshold_c$ is the threshold for class c
5: if yes, add that patch to the patch library along with the class and create a box ($N_c$x$N_c$) at v,
where $N_c$ is the box size for class c
6: don't extract any patch of the corresponding class (c) from the box
7: end for
end
```

Since the number of parameters that we have to train is huge, so our target was to extract maximum number of patches so that the training is better. An important concept to keep in mind is that we have to balance the patches from all the classes otherwise the training will not be good. So, 40% of the patches in the patch library for the HGG patients represent normal tissue and the rest is equally divided among the other classes. While for the LGG patch library 50% of the patches represent normal tissue and the rest if equally divided among the other classes.

We started with box size=2 for class 1 and found out the dice score for level 1, but it resulted in an overfitted CNN. So, we increased the box size to 3, 4, etc for class 1 and kept on doing that until we were able to generate sufficient number of patches to train the CNN. After doing this, we found that when box size was taken as 4 the dice score for necrosis (class 1) was maximum and the model does not overfits. Similarly, we found out the optimal box sizes for other classes. Thus, we finally end up with the following box sizes ($N_0$, $N_1$, $N_2$, $N_3$, $N_4$) = (32, 4, 10, 8, 8) as the optimal box sizes. Both the HGG and LGG model when trained using this method and box sizes gave promising results and the problem of overfitting was also resolved.

Note that, since 98% of the voxels belong either to the healthy tissue or the black surrounding area, so the box size for class 0 was maximum and since the number of voxels belonging to class 1 i.e. necrosis region is very low (minimum), so the box size for class 1 was minimum.

From the table below, we can clearly see that the segmentation accuracy varies when we change the box size for the algorithm discussed above which is why finding the optimal box size was also a very challenging task.

Table 3.1 Results using different box size for patch extraction

| | (32, 4, 8, 8, 8) | (32, 4, 10, 8, 8) | (32, 4, 12, 8, 8) |
|---|---|---|---|
| Dice score (in %) for level 2 only | 56 | 70 | 63 |

The time required for finding patches for HGG and LGG patients was approximately 40 hrs i.e. 1.6 days using this method. Overfitting does not occured when training the HGG and LGG models using the patches extracted from this method. Also, the segmentation accuracy for the complete tumor in the HGG patients was about 79% using this method.

#### 3.2.3.4 Method 4

This method is same as the previous method. We just included patches from the other planes as well in this method i.e. from the planes perpendicular to the coronal and the sagittal axis. But, Method 3 has better results as compared to this method.

```
Patch extraction: Method 4

Input: The training dataset

Output: A patch library

begin

1: for each voxel, v in the image

2: extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and
append them to get the desired patch of size 4x33x33

3: find the class of the patch

4: check if the no. of non zero intensity pixel is > $threshold_c$,

where, $threshold_c$ is the threshold for class c
```

5: if yes, add that patch to the patch library along with the class and create a box ($N_c x N_c$) at v, where $N_c$ is the box size for class c

6: don't extract any patch of the corresponding class (c) from the box

7: repeat step 1-6 for the other two planes as well.

8: end for

end

The time required for finding patches for HGG and LGG patients was approximately 120 hrs i.e. 5 days using this method. Overfitting does not occured when training the HGG and LGG models using the patches extracted from this method. Also, the segmentation accuracy for the complete tumor in the HGG patients was about 74% using this method.

## 3.3 CNN for HGG

In the figure below, we have shown the CNN designed for the segmentation of tumor in HGG patients [11]. The filter size is 3x3 for all the layers with a stride of 1x1 for the convolution layer and a stride of 2x2 for the max pooling layers. The first 3 layers of the model are convolution layer followed by a max pooling layer which are followed by another set of 3 convolution and a max pooling layer. Finally, we have 3 fully connected layers at the end of the model.

Table 3.2 CNN model for HGG patients

| | Type | Filter size | Stride | # filters | FC units | Input |
|---|---|---|---|---|---|---|
| Layer 1 | Conv. | 3x3 | 1x1 | 64 | - | 4x33x33 |
| Layer 2 | Conv. | 3x3 | 1x1 | 64 | - | 64x33x33 |
| Layer 3 | Conv. | 3x3 | 1x1 | 64 | - | 64x33x33 |
| Layer 4 | Max-pool. | 3x3 | 2x2 | - | - | 64x33x33 |
| Layer 5 | Conv. | 3x3 | 1x1 | 128 | - | 64x16x16 |
| Layer 6 | Conv. | 3x3 | 1x1 | 128 | - | 128x16x16 |
| Layer 7 | Conv. | 3x3 | 1x1 | 128 | - | 128x16x16 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Layer 8 | Max-pool. | 3x3 | 2x2 | - | - | 128x16x16 |
| Layer 9 | FC | - | - | - | 256 | 6272 |
| Layer 10 | FC | - | - | - | 256 | 256 |
| Layer 11 | FC | - | - | - | 5 | 256 |

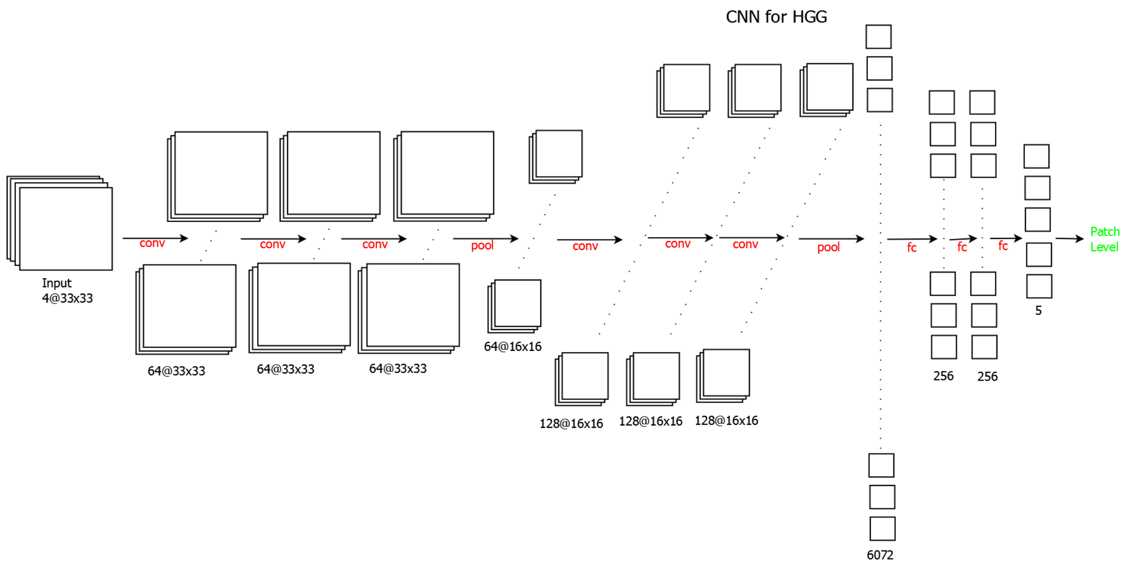


Figure 3.4 CNN for HGG patients

## 3.3.1 Balancing classes and data augmentation

**Balancing classes:** Another important factor in patch selection is to make sure that the classes of the input data are balanced. But, since approximately 98% of the voxels belongs to the tumor classes, therefore classes of the input data is not balanced for any of the algorithm discussed above. So, we have selected all the patches from the underrepresented classes and have randomly selected from the others.

Since, the number of parametes that we have to train for the HGG model is 21,18,213 for the classification of 89,28,000 voxels, therefore, we have to increase the size of the training dataset for better classification. For this, we have done run time augmentation.

**Augmentation:** Data augmentation is a common procedure in the context of CNN, when the training data set is small and the number of parameters to train is significantly large. Augmentation is basically, a technique to get around a lack of data in the dataset. Data augmentation can increase the size of training set by 10-fold or more. If the dataset is already very large then this technique may not be ineffective

We are doing rotation and flipping randomly in the training dataset at run time to generate new dataset. We randomly select some samples (t) from the batch (size=128) and randomly apply one of the three operations (rotation, horizontal flip and vertical flip) or a random combination of the three operations to get 't' modified samples which are then merged with the remaining (128-t) samples to get a new batch of 128 samples which is used to train the model. Also, the class of the samples (t) which are processed is kept the same as it was before applying any operation.

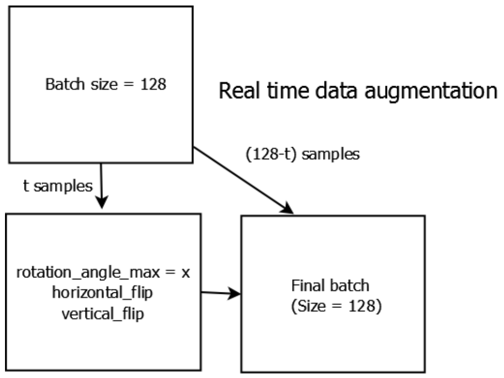


Figure 3.5 Real time data augmentation

## 3.4 Transfer learning

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. It is basically, machine learning with an additional source of information (from one or more related tasks) apart from the standard training data (refer the figure shown below).

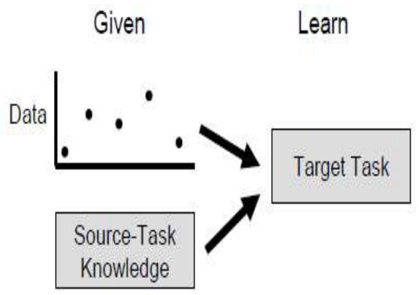


Figure 3.6 Transfer learning

### 3.4.1 Benefits of Transfer learning

Let us try to understand this with the help of an example. Most related objects are similar when viewed at a low resolution. E.g. low resolution images of most 4 legged farm animals have the same general shape. Thus, knowledge learned at low resolution may apply to all these animals. At higher resolution, details begin to emerge that differentiate them. [15]

In case of tumor also, at low resolution both the higher grade and lower grade tumors are similar. Also, since the  data available for LGG patients is very less i.e. data of only 54 patients. So, we are using transfer learning for the segmentation of tumor in LGG patients.

### 3.4.2 LGG model based on Transfer Learning

The CNN model for the LGG patients is trained in the following way. Firstly, the last 5 layers of the HGG model are removed and the 6th layer of the HGG model is made as the new output layer. Then the best learned weights that have already been learned for the segmentation of tumor in HGG patients are loaded. All the training patches from the LGG patients are passed through the model explained above and a vector of size (128, 16, 16) is generated and stored for each patch. Once we have extracted vectors for all the training patches of the LGG patients, we will then train the CNN for the LGG patients using those vectors.

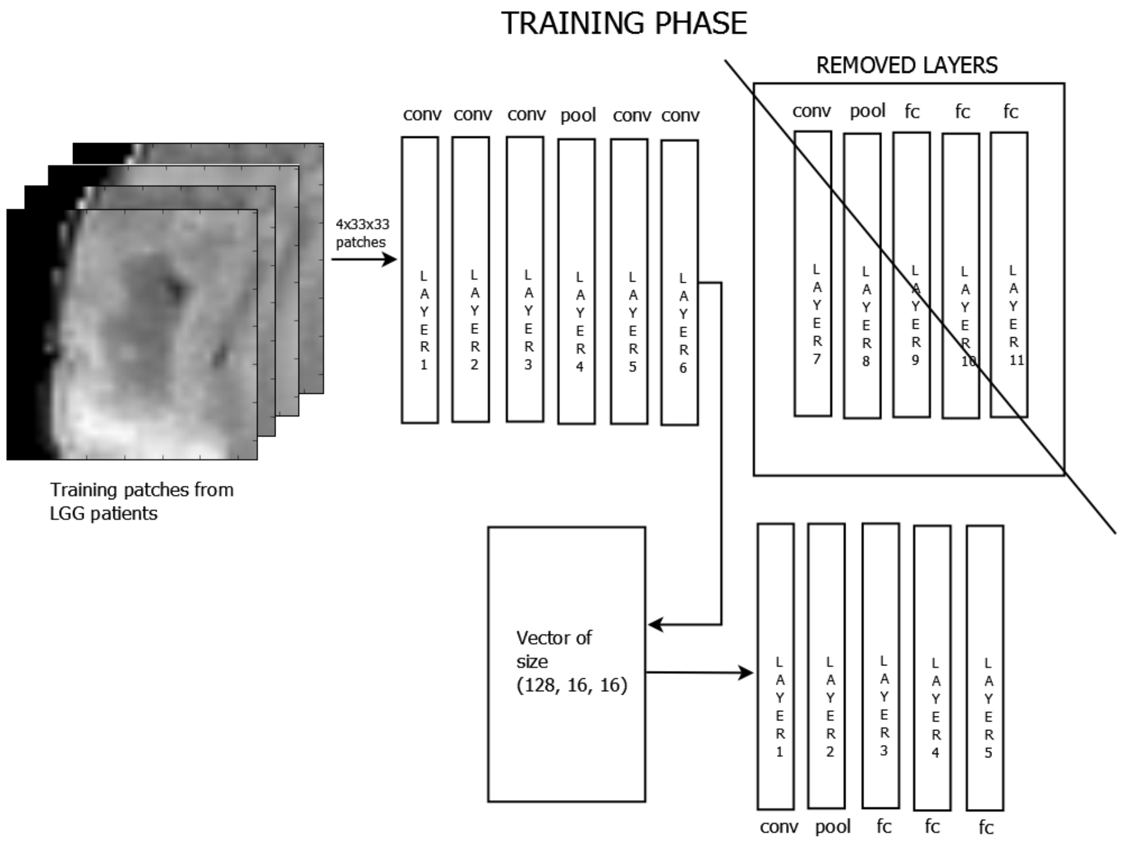


Figure 3.7 Model for LGG using Transfer learning

Table 3.3 CNN model for LGG patients

| | Type | Filter size | Stride | # filters | FC units | Input |
|---|---|---|---|---|---|---|
| Layer 1 | Conv. | 3x3 | 1x1 | 128 | - | 128x16x16 |
| Layer 2 | Max-pool. | 3x3 | 2x2 | - | - | 128x16x16 |
| Layer 3 | FC | - | - | - | 256 | 6272 |
| Layer 4 | FC | - | - | - | 256 | 256 |
| Layer 5 | FC | - | - | - | 5 | 256 |

### 3.4.3 CNN parameters

We have tried training the CNN for both HGG and LGG with different optimizers such as Adam, Adadelta, RMSprop, Nadam, SGD but the best result was given by SGD (Stochastic Grdient Descent). We have trained both the HGG and the LGG model for 20 epochs with initial learning rate as 0.003 and final learning rate as 0.00003. The learning rate has been linearly reduced. A momentum of 0.9 is being used. The batch size that we have used for training the is 128 in both HGG and LGG model. A dropout of 0.1 is used in case of HGG and 0.5 is used in case of LGG model. LeakyRelu ($f(x) = \max(0, x) + \alpha \min(0, x)$) has been used as the activation function with alpha=0.333 in all the layers in both HGG and LGG model, but in the last fully connected layer we have used softmax activation function. The parameters of the CNN have been initialised with Xavier uniform initialisation [12]. In the convolution layers the feature maps are padded before convolution for both HGG and LGG model so that the resulting feature map could maintain the same dimension.

## 3.5 Segmentation

Segmentation is the process of partitioning an image into multiple regions or segments. It is used to simplify the image representation and thus helps to locate the object of interest within the image. Precisely, segmentation is the process of assigning a label to each pixel/voxel in an image such that pixels/voxels of the same label share certain characteristics.

In the output of our models i.e. in the segmented image, necrosis is represented by integer value 1, edema by 2, non-enhancing tumor by 3, enhancing tumor by 4 and others are represented as 0.

For doing brain tumor segmentation, first of all the input image is pre-processed in the similar way as the training data was pre-processed and then the patch for each voxel is extracted and their classes are predicted through the model to get the segmented image.

### 3.5.1 Segmentation in HGG

Tumor segmentation in the HGG patients is done in the following way. For each voxel in the MRI of the HGG patients, patch of size 33x33 is extracted from all the 4 sequences i.e. T1, T1c, T2 and Flair and are appended to get a new patch. Then, this new patch is passed through the HGG model to get the prediction of the class of the patch which is then stored in a 3d vector of size 155x240x240 at the corresponding location of the voxel.

```
Segmentation algorithm for HGG model
Input: MRI of a patient
Output: Segmented image (of size155x240x240)
begin
1: pre-process the input image in the same way as the training data was pre-processed
2: for each voxel (v) in the image
3: extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and
   append them to get the desired patch of size 4x33x33
4: predict the class of the patch using the model (CNN) constructed for the HGG patients
5: store the class of the patch at the location of v in the segmented image
6: end for
end
```

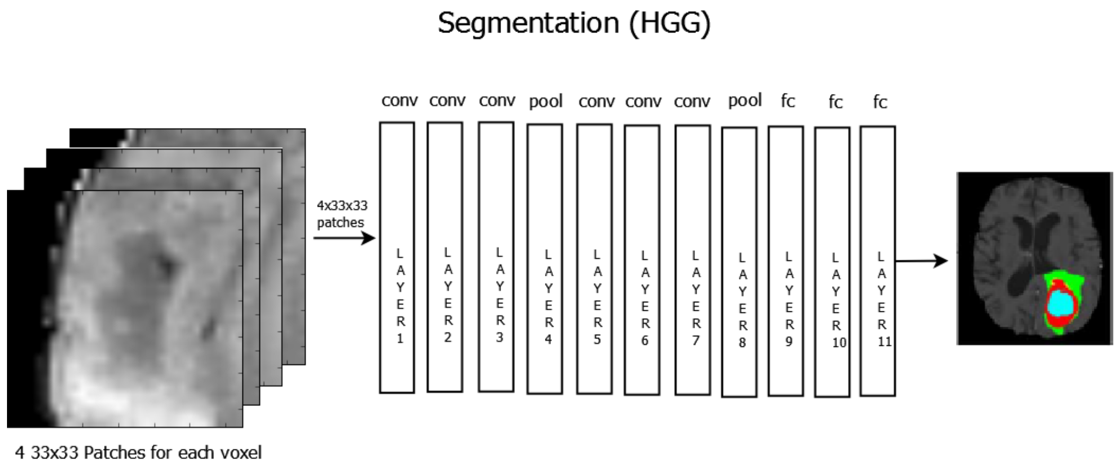


Figure 3.8 Segmentation for HGG patients

### 3.5.2 Segmentation in LGG

The segmentation of tumor in the LGG patients is done in the following way. Firstly, the last 5 layers of the HGG model are removed and the 6th layer of the HGG model is made as the new output layer. Then the best learned weights that have already been learned for the segmentation of tumor in HGG patients are loaded. Now, for each voxel in the MRI of the LGG patients, patch of size 33x33 is extracted from all the 4 sequences i.e. T1, T1c, T2 and Flair and are appended to get a new patch. Then this new patch is passed through the model explained above to get an output vector of size 128x16x16 which is again passed through the five layers of the LGG model and the level of the patch is generated and stored in a 3d array of size 155x240x240 at the corresponding location of the voxel.

```
Segmentation algorithm for LGG model

Input: MRI of a patient
Output: Segmented image (of size155x240x240)
begin
1: pre-process the input image in the same way as the training data was pre-processed
2: for each voxel (v) in the image
3:     extract patch of size 33x33 with v at the centre of the patch from each MRI sequence and
       append them to get the desired patch of size 4x33x33
4:     pass the patch through the first 6 layers of the HGG model to get a vector of size
       128x16x16
5:     pass this vector through the 5 layers of the LGG model to get the class of the patch
5:     store the class of the patch at the location of v in the segmented image
6: end for
end
```

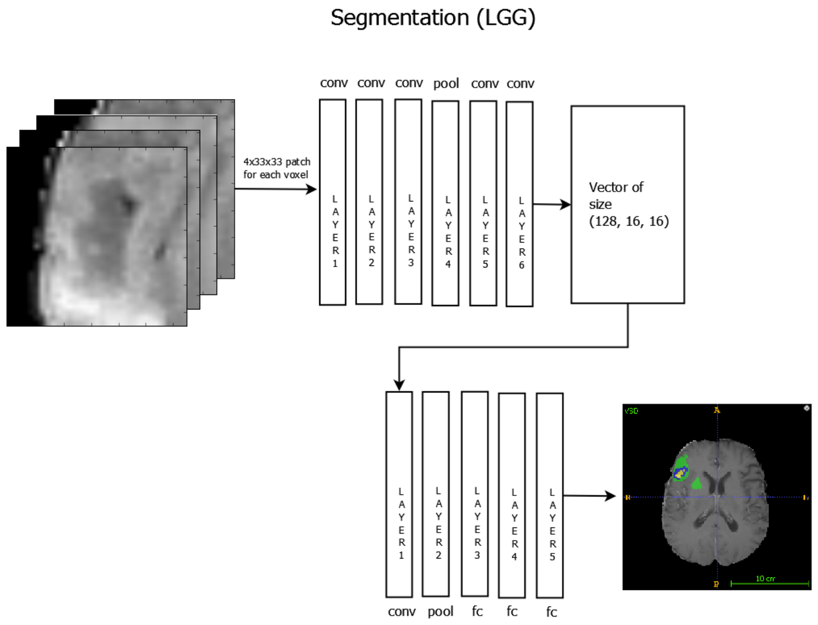


Figure 3.9 Segmentation for LGG patients

## 3.6 Post processing

Some small clusters may be erroneously classified as tumor. So, to deal with that we have used volumetric constraints to remove such errors. The $T_{vol}$ for HGG patients is 10,000 while the $T_{vol}$ for the LGG patients is 3,000. We have tried using erosion as a post-processing measure to get better boundary for the tumor but it has been found to be ineffective. In fact, the segmentation accuracies decreased after applying erosion over the segmented images.

# Chapter 4 Results

In this chapter, we will be showing segmentation accuracies of some of the benchmark algorithms. We will also compare our results with the deep learning method proposed by Pereira et al. [11] which is the present best algorithm in brain tumor segmentation to the best of our knowledge.

The model has been implemented in python with Theano [8][9] as the backend. Dell precision Tower 7810 has been used for running the code of all the algorithms discussed in this report.

## 4.1 Ground truth vs. Automatic Segmentation

Below are some images showing the difference between automatic segmentation and the actual ground truth values. From the images shown below we can clearly see that the proposed model for LGG patients could detect the location, shape, size and the intra-tumoral structure quite precisely. Although, the segmentation of tumor in LGG patients is considered as a very difficult task due to the lack of data, but our model for LGG patients based on transfer learning is performing quite well as can be seen from the images below.

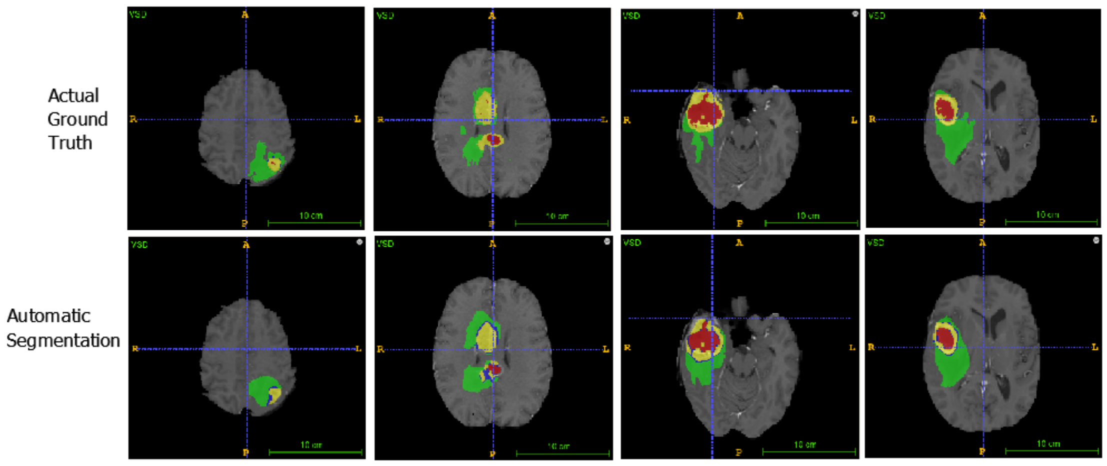


Figure 4.1 Ground truth vs. Automatic segmentation for HGG patients

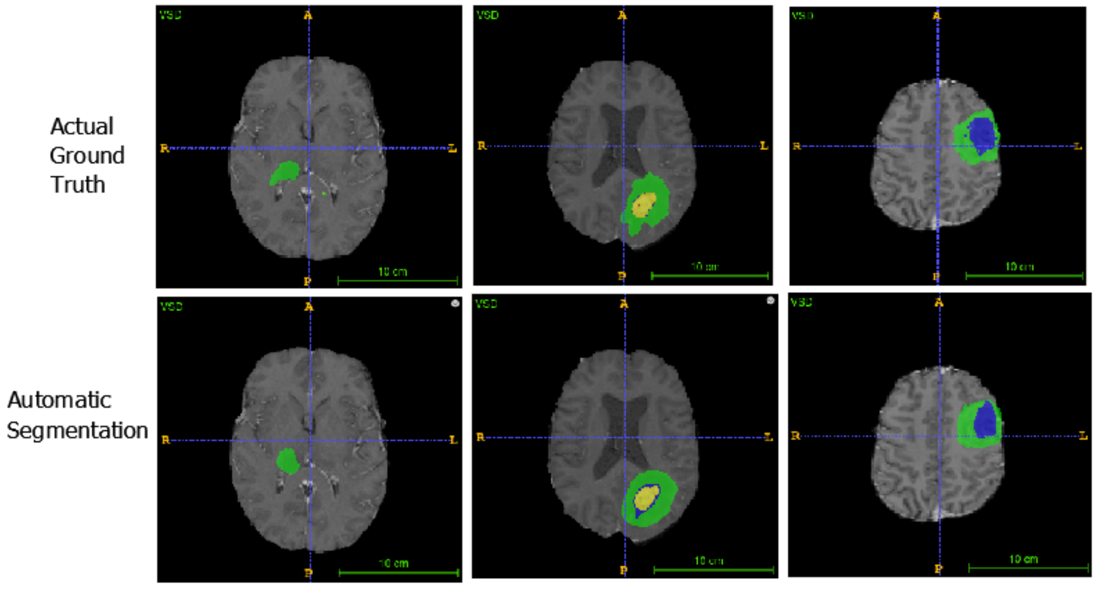
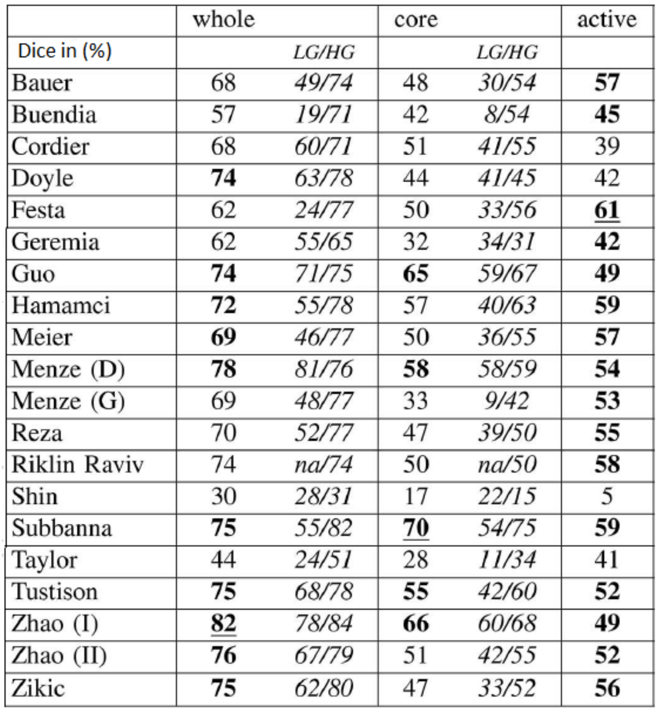


Figure 4.2 Ground truth vs. Automatic segmentation for LGG patients

## 4.2 Comparative study with existing literature

Results of some benchmark methods on BraTS 2013 dataset is shown below. From the figure below it can be clearly seen that no single algorithm performed the best in all the three cases of evaluation i.e. whole, core and the active tumor. The algorithm proposed by Zhao (I) performed the best in “whole tumor segmentation” whereas “core tumor segmentation” was best performed by the algorithm proposed by Subbanna and the algorithm proposed by Festa outperformed all the algorithms in “active tumor segmentation”.



| | whole | | core | | active |
|---|---|---|---|---|---|
| Dice in (%) | | LG/HG | | LG/HG | |
| Bauer | 68 | 49/74 | 48 | 30/54 | **57** |
| Buendia | 57 | 19/71 | 42 | 8/54 | **45** |
| Cordier | 68 | 60/71 | 51 | 41/55 | 39 |
| Doyle | **74** | 63/78 | 44 | 41/45 | 42 |
| Festa | 62 | 24/77 | 50 | 33/56 | **61** |
| Geremia | 62 | 55/65 | 32 | 34/31 | **42** |
| Guo | **74** | 71/75 | **65** | 59/67 | **49** |
| Hamamci | **72** | 55/78 | 57 | 40/63 | **59** |
| Meier | **69** | 46/77 | 50 | 36/55 | **57** |
| Menze (D) | **78** | 81/76 | **58** | 58/59 | **54** |
| Menze (G) | 69 | 48/77 | 33 | 9/42 | **53** |
| Reza | 70 | 52/77 | 47 | 39/50 | **55** |
| Riklin Raviv | 74 | na/74 | 50 | na/50 | **58** |
| Shin | 30 | 28/31 | 17 | 22/15 | 5 |
| Subbanna | **75** | 55/82 | **70** | 54/75 | **59** |
| Taylor | 44 | 24/51 | 28 | 11/34 | 41 |
| Tustison | **75** | 68/78 | **55** | 42/60 | **52** |
| Zhao (I) | **82** | 78/84 | **66** | 60/68 | **49** |
| Zhao (II) | **76** | 67/79 | 51 | 42/55 | **52** |
| Zikic | **75** | 62/80 | 47 | 33/52 | **56** |

Figure 4.3 BraTS 2013 results [10]

A comparison of the proposed model with the deep learning method of Pereira et al. [11] is shown in the table below.

Table 4.1 Comparison of results

| | whole | | core | | active | parameters | parameters |
|---|---|---|---|---|---|---|---|
| Dice in (%) | | LG/HG | | LG/HG | HG | (HGG) | (LGG) |
| Pereira et al. | 84 | 65/88 | 72 | 53/76 | 73 | 21,18,213 | 19,33,701 |
| Proposed | 77 | 74/79 | 74 | 62/80 | 75 | 21,18,213 | 18,20,549 |

We can clearly see (from the table above) that the number of parameters in the method proposed by Pereira et al. [11] is larger as compared to that of us. Also, the number of patches used by them for training the model for LGG patients is around 3,35,000 whereas we have used around 1,60,000 patches for training the CNN for LGG patients. Thus, the training time of our algorithm is significantly less as compared to them. Also, we are getting better results in all the cases in LGG patients as compared to them. In case of HGG patients, our segmentation results in core and active tumor is a little better as compared to them but we are significantly lacking behind in complete tumor segmentation.

# Chapter 5 Conclusion and Future Work

In this Project, we have looked at the problem of automatically segmenting tumor from the MR images and we have come up with four different algorithms for extracting patches which can be used to train CNN's to do the automatic segmentation. We also came up with two different CNN's, one for the HGG and the other for the LGG patients to do the automatic segmentation in a reasonable amount of time with high accuracy. We have also shown how transfer learning has helped in increasing the segmentation accuracy in case of LGG patients.

While the proposed model yields promising results, an application such as brain tumor segmentation leaves no room for errors. In a surgical setting it is essential to remove as much of the tumor mass as possible without damaging the surrounding healthy tissues. So, in the future, the work could be extended by developing more advanced patch extraction algorithms which will help in increasing the segmentation accuracy. Till now we have looked into extracting and training using 2D patches, the work could also be extended by designing models which could be trained on 3D patches and thus trying to come up with good algorithms for extracting 3D patches.

# Bibliography

[1] L. G. Nyul, J.K. Udupa, and X. Zhang, “New variants of a method of MRI scale standardization,” IEEE Trans. Med. Imag., vol. 19, no. 2, pp. 143-150, Feb. 2000

[2] N. J. Tustison et al., “N4ITK: Improved n3 bias correction,” IEEE Trans. Med. Imag., vol 29, no. 6, pp. 1310-1320, Jun. 2010

[3] L. Nyul and J. Udupa, “On standardizing the MR image intensity scale,” Magn. Reson. Med., vol. 42, no. 6, pp. 1072-1081, 1999

[4] M.Shah et al., “Evaluating intensity normalization on MRIs of human brain with multiple sclerosis,” Med. Image Anal., vol. 15, no. 2, pp. 267-282, 2011

[5] A. Krizhevsky, I.Sutskever, and G. E. Hinton, “Imagenet classification with deep convolution neural networks,” in Adv. Neural Inform. Process. Syst., 2012, pp. 1097-1105

[6] L. R. Dice, “Measures of the amount of ecologic association between species,” Ecology, vol. 26, no. 3, pp. 297-302, 1945

[7] BraTS 2015 dataset [Online]. Available: https://www.smir.ch/BRATS/Start2015

[8] F. Bastien et al., “Theano: New features and speed improvements,” in Deep Learn. Unsupervised Feature Learning NIPS 2012 Workshop, 2012

[9] J. Bergstra et al., “Theano: A CPU and GPU math expression compiler,” in Proc. Python Sci. Comput. Conf. (SciPy), Jun, 2010

[10] Bjoern H. Menze et al., “The multimodal brain tumor image segmentation benchmark (BRATS)”, IEEE Trans. Med. Imag., vol. 34, no. 10, Oct 2015

[11] Sergio Pereira, Adriano Pinto, Victor Alves and Carlos A.Silva., “Brain tumor segmentation using convolution neural networks in MRI images”, IEEE Trans. Med. Imag., vol. 35, no. 5, May 2016

[12] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proc. Int. Conf. Artif. Intell. Stat., 2010, pp. 249-256

[13] S. Bauer et al., “A survey of MRI-based medical image analysis for brain tuor studies,” Phys. Med. Biol., vol. 58, no. 13, pp. 97-129, 2013

[14] D. N. Louis et al., “The 2007 who classification of tumors of the central nervous system,” Acta Neuropathologica, vol. 114, no. 2, pp. 97-109, 2007

[15] Eric Eaton, Marie desJardins, “Knowledge transfer with a multiresolution esemble of classifiers,” ICML-06, Workshop on structural knowledge transfer for machine learning, June, Pittsburg

[16] Nima Tajbakhsh et al, "Convolution neural networks for medical image analysis: full training or fine tuning?," IEEE Trans. Med. Imag., vol. 35, no. 5, May 2016

[17] D. Zikic et al., "Segmentation of brain tumor tissues with convolution neural networks," MICCAI Multimodal Brain Tumor Segmentation Challenge (BraTS), pp. 36-39, 2014

[18] G. Urban et al., "Multi-modal brain tumor segmentation using deep convolution neural networks," MICCAI Multimodal Brain Tumor Segmentation Challenge (BraTS), pp. 1-5, 2014

[19] M. Havaei et al., Brain tumor segmentation with deep neural networks 2015 [Online]. Available: http://arxiv.org/abs/1505.03540, ArXiv: 1505.03540v1

[20] Imagenet challenge [Online]. Available: http://www.image-net.org/challenges/LSVRC/2017/index.php#loc