CONVOLUTIONAL NEURAL NETWORK BASED BRAIN MRI SEGMENTATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BORA BAYDAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2018

Approval of the thesis:

CONVOLUTIONAL NEURAL NETWORK BASED BRAIN MRI SEGMENTATION

submitted by **BORA BAYDAR** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Tolga Çiloğlu _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Gözde Bozdağı Akar _____
Supervisor, **Electrical and Electronics Engineering, METU**

Examining Committee Members:

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Aykut Erdem
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Gökberk Cinbiş
Computer Engineering, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: BORA BAYDAR

Signature :

ABSTRACT

CONVOLUTIONAL NEURAL NETWORK BASED BRAIN MRI SEGMENTATION

BAYDAR, BORA
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Gözde Bozdağı Akar

June 2018, 97 pages

Visualization of the inner parts of human body is crucial in modern medicine and magnetic resonance imaging(MRI) is one of the widely used medical imaging methods. Manual analysis of MRIs, however, wastes the valuable time of experts. Development of an automatic segmentation method for brain MRIs can save time spent by the experts and can avoid human error factor. In this thesis, convolutional neural network (CNN) based methods are applied on brain MRI segmentation problem. The basic architectures used are FCN-8 and U-NET. Performance of different approaches has been analyzed by focusing on structural modifications, upsampling methods, activation functions, loss functions, pre-processing and post-processing methods. For the activation functions, ReLU, LReLU, PReLU and tanh are experimented. Histogram matching, normalization and histogram equalization have been applied for pre-processing. Conditional random fields and a 3 dimensional connected component analysis are separately integrated to the network as post-processors. Results are compared in terms of dice score, sensitivity and specificity.

The experimental results show that the combination of two separate U-Nets has the

best performance. Dilation modules also improve the results when inserted on a shallow network. When combined with additional residual connections, they have also improved the overall results. Inception modules do not provide a remarkable performance improvement. ReLU and PReLU have shown the best performance. No significant difference have been observed between results obtained from different upsampling methods, although bilinear interpolation, despite being non-trainable, has slightly better results.

Keywords: Convolutional, Neural, Network, MRI, Brain, Medical, Image, Segmentation

ÖZ

EVRİŞİMLİ SİNİR AĞLARI TABANLI BEYİN MRI BÖLÜTLEMESİ

BAYDAR, BORA
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Gözde Bozdağı Akar

Haziran 2018 , 97 sayfa

İnsan vücudunun iç kısımlarının görselleştirilmesi, modern tıp için önemlidir. Manyetik rezonans görüntüleme (MRG) en yaygın kullanılan tıbbi görüntüleme yöntemlerinden biridir. Ancak MRG'lerin el ile analizi, uzmanların değerli zamanlarını boşa harcamaktadır. Beyin MRG'leri için otomatik bir bölütleme yönteminin geliştirilmesi, uzmanlar tarafından harcanan zamandan tasarruf sağlayabilir ve insan hatası faktörünü önleyebilir. Bu tezde, evrişimli sinir ağları (ESA) tabanlı yöntemler beyin MRG bölütlemesi problemine uygulanmıştır. Kullanılan temel mimariler FCN-8 ve U-NET'dir. Farklı yaklaşımların performansı, yapısal modifikasyonlar, örnekleme yöntemleri, aktivasyon fonksiyonları, kayıp fonksiyonları, ön işlem ve işleme sonrası yöntemlere odaklanarak analiz edilmiştir. Aktivasyon fonksiyonları için ReLU, LReLU, PReLU ve tanh denenmiştir. Ön işleme için histogram eşleştirme, normalleştirme ve histogram eşitleme uygulanmıştır. Koşullu rastgele alanlar ve 3 boyutlu bağlantılı bileşen analizi, ağa post-işlemciler olarak ayrı ayrı entegre edilmiştir. Sonuçlar, Dice skoru, duyarlılık ve özgüllük açısından karşılaştırılmıştır.

Deneysel sonuçlar iki U-Net'den oluşan kombinasyonun en iyi performansa sahip ol-

duğunu göstermektedir. Genişleme modülleri, sığ bir ağa yerleştirildiğinde sonuçları iyileştirmiştir. Ek artık bağlantılar ile birleştirildiğinde, genel sonuçları da iyileştirdiği gözlenmiştir. Başlangıç modülleri dikkate değer bir performans artışı sağlamamıştır. ReLU ve PReLU en iyi performansı göstermiştir. Farklı üst-örnekleme yöntemlerinden elde edilen sonuçlar arasında anlamlı bir farklılık gözlenmemiştir; ancak, çift doğrusal aradeğerleme, eğitilebilir olmamasına rağmen, biraz daha iyi sonuçlara sahiptir.

Anahtar Kelimeler: Evrişimli, Sinir, Ağları, MRI, MRG, Beyin, Tıbbi, Görüntü, Bölütleme

Daima yanımda olan geniş aileme...

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my supervisor Prof. Gözde Bozdağı Akar. I have had experience in various areas in signal/image processing under her precious guidance. I have learnt so much from her academically, which finally led me to writing this thesis. I must also mention her great kindness to every student. Looking past my last undergraduate semester, it was the most fortunate thing for me to become a member of METU Multimedia Research Group led by her.

Secondly, I would like to thank to Savas Ozkan, who is currently the most experienced member of METU Multimedia Research Group. Without his knowledge and experience on convolutional neural networks, writing this thesis would be much more difficult.

I must also note METU Multimedia Research Group's gratitude to NVIDIA for their Quadro P5000 GPU donation to our laboratory.

I also would like to mention my appreciations to the other members of METU Multimedia Research Group, Ece Selin Boncu, Volkan Okbay and Alican Hasarpa, for their support, motivation and friendship.

I would like to thank Basak Usta for being there for me at both good and bad times. She was very self-giving, motivating, patient and loving.

Finally, I want to thank my parents, Mehmet Fatih Baydar and Gulsun Nese Baydar. I am very proud to be their son.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FNN	Feedforward Neural Network
GPU	Graphics Processing Unit
NN	Neural Network
RNN	Recurrent Neural Network
MRI	Magnetic Resonance Image
ROI	Region of Interest
FLOPS	Floating Point Operations
WT	Whole Tumor
TC	Tumor Core
ET	Enhancing Tumor
ED	Peritumoral Edema
NCR/NET	Necrotic and Non-Enhancing Tumor
HGG	High Grade Glioma
LGG	Low Grade Glioma
MAP	Maximum A Posteriori

CHAPTER 1

INTRODUCTION

Medical images are mostly used for visualization of the inner parts of human body. They are crucial to detect and plan the treatment of severe diseases. One of the most widely used medical imaging methods is magnetic resonance imaging (MRI). In MRI, the patient is placed in a machine that produces a strong magnetic field in order to align the protons with it. Then, by sending a radio-frequency pulse, the alignment is broken. When the radio-frequency pulse ends, the protons realign with the magnetic field. Depending on the tissue, time required for the realignment and energy released during this event differs [9]. By recording this information a 3D image(MRI) is constructed.

Since the introduction of magnetic resonance imaging in 1970s, its utilization has been spreading increasingly. According to OECD data [10], there are more than 100 MRI examinations each year per 1000 person in both developed and developing countries such as United States, France and Turkey. One of the most important applications of magnetic resonance imaging is brain tumor detection. However, since there is no fully automatic detection system for brain tumors, MRIs are analyzed by human experts. This requires time and concentration, which causes human error. In particular, the increase in the number of MRI examinations raises tiredness and affects the performance of experts. Additionally, the valuable time of them is wasted by this operation. For instance, there are 700.000 [11] people living with a primary brain tumor only in the United States. Thus, the development of automatic detection systems would be quite beneficial for health care system.Thanks to the advances in technology, MRIs are digitized and image processing techniques can be applied on them in order to automate segmentation.

Image segmentation is to partition an image into its coherent parts. These partitions after the segmentation have similar characteristics such as color, texture etc. Some of the most widely known image segmentation methods in the literature are clustering based methods, edge based methods, compression based methods, region based methods, graph based methods and pixon based methods. Although image segmentation makes an image easier to analyze, it actually does not convey high-level information. In order to interpret the content of an image, these segments should be classified. Semantic image segmentation combines segmentation with per-pixel classification. In other words, in semantic segmentation, each pixel is assigned a class label and when those pixels are combined, they correspond to segments in the image. We can categorize image segmentation as unsupervised and supervised segmentation.

Unsupervised Segmentation In unsupervised segmentation methods, the inputs are provided but ground truths are not provided by the user for training the algorithm nor (almost) user interaction is needed. So, the algorithm learns by itself or it does not need any supervised learning. For example, k-means clustering learns to differentiate between labels of pixels and tries to find the most matching cluster for each pixel. On the other hand, watershed [12] algorithm treats the image like it was a geographical map and separates the segments using highest points of this map, while edge based segmentation methods try to find edges and assumes each edge is a border between different segments.

Supervised Segmentation Supervised segmentation methods need an input and a user interaction for the segmentation. User needs to either mark a region or it needs to provide the correct segmentation output to the algorithm. For example, in graph cuts [13], user draws lines to different regions that are separated from each other. Similarly, in trainable methods, user provides the ground truths and a model is trained so that it learns to do the segmentation by itself. An example of trainable methods is classification based method, which is the main scope of this work.

Classification Based Supervised Segmentation Image classification is to label an image or a part of an image with the corresponding classes based on its features. The

classical approach for the classification has two main steps. The first one is the feature extraction. In this step, important information related to the class labels are obtained. Edges, corners and color information are only a few examples for these features. The second step is to decide which class those features belong to. This step is carried out by machine learning algorithms. Features are actually high dimensional vectors that reside in a feature space and represent the object/content. Machine learning algorithms strives to separate the feature space into meaningful regions(hyperplanes) that correspond to classes. If the feature vector of an object is in one of these regions, then that object is predicted as the corresponding class. In order to train the machine learning algorithms, both supervised and unsupervised learning can be used. K-means clustering is one of the simplest examples of unsupervised algorithms. It only requires the number of classes from the user; thus, it can also be considered as semi-supervised. Support Vector Machines [14] on the other hand is an example supervised learning algorithm which requires labels of the objects during training.

1.1 Scope of the Thesis

The scope of this thesis is to deeply investigate performance of different approaches for brain tumor segmentation that are based on convolutional neural networks (CNN). Starting from different architectures, this work compares both training and validation results obtained from different upscaling methods, activations, cost functions, post-processing methods and pre-processing methods. For the evaluation, performance metrics taken into account are dice coefficient, sensitivity and specificity. For training and validation, BraTS 2017 dataset is used [15, 16]. The performance of these approaches are also evaluated on another brain MRI dataset.

It is important to note that the scope of the thesis does not cover cascaded network architectures or patch based methods. Only a single architecture type is used for each method. More intuitively, whole MRI slices of different modalities are given to a network that solves a multi-class problem. Methods that crop part of the MRI slices are not involved in the thesis. Similarly, methods that have stacked binary classifier networks to segment each class are not involved.

1.2 Outline of the Thesis

This thesis is outlined as follows: Chapter 2 explains different approaches to brain tumor segmentation proposed in the literature. We briefly state their advantages and disadvantages. Chapter 3 describes several critical concepts about convolutional neural networks while Chapter 4 explains the network architectures used in this work and modifications that are presented. Chapter 5 focuses on post-processing methods and comparison of their results. Implementation details and results are provided in Chapter 6, which also includes the details of the dataset we used. Finally, Chapter 7 makes a conclusion of the work and presents future work.

CHAPTER 2

LITERATURE REVIEW

Obtaining a robust computer aided diagnosis technique for brain tumor segmentation has been a challenging problem for several years since hand labeling brain MRIs to segment anomalies is time consuming and inefficient. There are many methods proposed to solve the problem of brain tumor segmentation. This chapter investigates different approaches by categorizing them into unsupervised and supervised methods. Additionally, pre-processing and post-processing methods will be investigated.

2.1 Unsupervised Brain Tumor Segmentation

The main advantage of unsupervised methods is that they do not require a dataset with ground truth labels. These methods use distinct features such as color, spatial position, symmetry etc. In brain tumor segmentation, unsupervised methods are generally used for initial segmentation of the whole tumor region or ROI, rather than detailed segmentation of tumor regions such as tumor core and enhancing tumor. [17] [18] [19] [20] [21] use color information and fuzzy C-means for clustering the data. [22] uses SLIC algorithm which utilizes color and spatial information [23]. However, unsupervised algorithms like fuzzy C-means and SLIC, that use color information and spatial positions are not very successful because the tumor may have a similar color to other parts of brain. Moreover, if the tumor regions is not spatially distant to other parts that share similar pixel intensities, these algorithms have difficulty in distinguishing the anomalies from normal brain tissues.

Another popular feature of the brain images are that they are almost symmetric when the axial view is considered. [17] [22] use symmetry information. The symmetry

axis does not always overlap with the y-axis of the axial image. It may also not be a straight line due to deformations in the brain. For example, [24] and [25] try to find the symmetry axis for further processing. [26] uses symmetry, color and texture information and adopts a decision forest algorithm.

Atlas based methods are also used in the literature [27] [28] [29]. The data collected is segmented by the experts. Then it is combined to form a single atlas image. The input data is registered so that it is aligned with the atlas image. Using this registered image, the input is segmented. However, it requires lots of data, by the fact that brain size and shape may vary according to age, race and environmental factors. Even for such databases they may not give as promising results as supervised methods.

2.2 Supervised Brain Tumor Segmentation

If the ground truth of the dataset is provided, as in BraTS challenge, machine learning algorithms can be used to distinguish between different classes which are brain and tumor parts in these cases. [30] and [31] use a Support Vector Machine to classify the regions according to their intensity and texture features. [32], [22] and [17] use Random Forest for classification. [33] utilizes Markov Random Field to segment the image using spatial and structural information. Since the main objective of this thesis is to compare CNN based methods in various configurations, different approaches on CNNs are explained in more detail in 2.2.1.

2.2.1 Convolutional Neural Network Based Brain Tumor Segmentation

Total number of methods based on CNNs has increased recently and proven their success on brain tumor segmentation. With some minor exceptions, most of the methods proposed so far are based on CNNs. The methods can differ from each other in terms of input dimensions and input size, filter size, network depth and connection paths etc. Although various network architectures have been used for image segmentation such as FCN-8s [34] and DeepMedic [35], most of the medical image segmentation methods, including the brain tumor segmentation, have a U-Net[36] architecture. FCN-8s and U-Net have a downsampling part where the features are found and an upsam-

pling part where the localization is done [34][36]. Thus, they produce an output with the same dimensions as the input. Unlike these, DeepMedic takes a 3D patch of the input and its output is even smaller than the input patch [35]. So while it simultaneously finds features and localizes them on a smaller patch. [37], [38],[39],[40],[41], [42],[43],[44], [36], [45], [46],[47],[48],[47] utilize a U-Net architecture. On the other hand, an FCN-8s like architecture is preferred in some works [49] [50] while some of them adapts DeepMedic on brain tumor segmentation problem [51]. Also, combinations of different architectures such as FCN-8s, U-Net and DeepMedic are applied in some of the proposed methods[52].

[53] uses dilated convolutions which was proposed by [7], stating that dilated convolutions (context modules as named in the paper) are developed for dense prediction problems like semantic segmentation. Increase in the performance of popular networks such as VGG [54], with the addition of context modules, is also presented in the paper. The main benefit of context modules is that they can extract features of different scales using dilated convolutions. Thus, instead of giving images to a network in different scales as in [35], one may improve the performance by adding context modules to the network.

Residual connections (or shortcut connections as named in [55], [56]) plays an important role in the convolutional neural networks since they help avoiding the vanishing gradients problem and makes deeper networks possible by adding the input of previous layer to the response of the next layer [57]. By doing so, it simply introduces the error in the earlier stages so that while propagating the error from output through input layer, the error gradient does not vanish. Moreover, there are methods that use residual connections in brain tumor segmentation problem. For example, [35] indicates promising results, yet the authors improve their previous architecture in [51], stating that addition of residual connections improves the performance of the network for all classes.

[43] uses pre-activation residual blocks instead of post-activation residual blocks. This provides a faster learning rate compared to classical approach in Deep Residual Networks (ResNet [57]) according to [8].

[58] uses inception modules which initially proposed in GoogLeNet [59]. Its advan-

tage is that instead of choosing between 3×3 or 5×5 filter sizes in convolutions, the method uses both of them and allows the network to determine their priorities by itself. However, the network used in [59] is very deep and the training data is very large compared to data used in this work. This inevitably induces overfitting. Thus, the performance may not increase as expected in theory.

[60] proposes a cascaded network that iteratively segments the images for each class. In other words, first it detects the whole tumor for an entire image. Then the region of interest is restricted around this tumor area and the region is fed into another network for the segmentation of tumor core as well as enhancing tumor core.

2.3 Pre-processing and Post-processing

It should be noted that the pre-processing and post-processing play an important role in brain tumor segmentation. For pre-processing, histogram equalization, histogram matching and normalization are quite popular. [38], [37] use histogram equalization. [61] uses histogram matching as proposed in [62] while [63] does histogram matching, choosing the target histogram by averaging histograms of all training data. [49] selects one of the patients to use its histogram as the reference while [64] choses 10 random images to obtain a reference histogram. Although there are not any obvious advantages of these pre-processing methods over one another, while applying pre-processing to normalize the data, one must consider that the assumption of each patient's having a brain tumor is not practical in real life applications. Thus, methods should be adapted in a way that is generalized for both normal brains and brains with anomalies.

The most basic yet successful operation for post-processing is connected component analysis and morphological operations [41] [47] [65]. Conditional Random Fields is also used as post-processing method to regularize the results [66] [67] [68] [41]. [67] states that the Dense Conditional Random Field method increases the performance in all categories. [41] states that the parameter choice is very important and it can easily fail to improve performance.

Another important method that can be count in the post-processing methods is gen-

erative adversarial networks (GAN) [69]. In the image segmentation problem, the segmentation result is given to the discriminator along with the original input images. In this case, input images act as a condition on the segmentation results. Thus, GAN is called conditional adversarial network in this case. [70] proposes a network that generates an output image based on the label input. A similar approach is utilized for the image segmentation problem [71]. However, in this case, the image is given as the input and segmentation labels are generated. [61] and [72] adapt this method for brain tumor segmentation problem. [72] states an improvement of the performance in dice scores with the addition of an adversarial network to classical CNN.

CHAPTER 3

CONVOLUTIONAL NEURAL NETWORKS

In this chapter, convolutional neural networks will be explained in detail. First neural networks, then some specific methods used in CNNs will be explained. We will describe how CNNs are used for classification and segmentation problems, in Section 3.1.5.4 of this chapter. The details of architectures used for pixel-wise classification task are given in Chapter 4.

3.1 Neural Networks

It has been a while since the CPUs have outraced human brain in complex operations per second such as floating point operations. A single average CPU core can compute more than 5 billion floating point operations (5 GFLOPS). For a human, such complex operations may even be impossible to compute. On the other hand, human brain is still considered much powerful than any personal computer. Although human brain can not compute complex mathematical operations, it can do much more than a CPU by combining many simple operations and is still considered more successful in tasks like image classification. In order to achieve such tasks using computers, scientists have tried to imitate the human brain since 1940s and a concept called Artificial Neural Networks has emerged.

3.1.1 Basics of Artificial Neural Networks

Artificial neurons are imitations of biological neurons. A biological neuron, as shown in Figure 3.1, generates an electrical signal. If its dendrites are excited by enough

number of synapses of other neurons, then it produces an electrical signal transmitted through its axon and excites dendrites of other neurons through its synapse. Like biological neurons, artificial neurons are connected to each other and excite each other through these connections. They (or perceptrons in its simpler form) are capable of only calculating a very simple function called neuron activation (Eq. 3.1). An artificial neuron, first produced by [73] and shown in Figure 3.2 which generates a binary output according to the result of this function is called a *perceptron* [74].

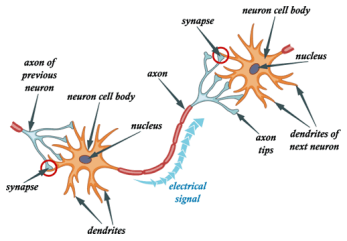


Figure 3.1: A biological neuron is fired when it is excited sufficiently by other neurons connected to it. (Image taken from [1])

$$a = \sum_{j=1}^{N} u_j w_j + \theta \tag{3.1}$$

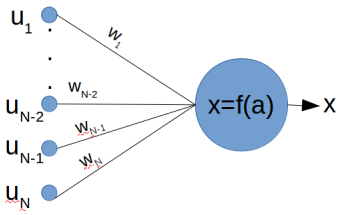


Figure 3.2: An artificial neuron. If $f(a)$ (activation function) is a threshold function, then this neuron is called a perceptron.

An *artificial neural network* is, as the name suggests, a network composed of artificial neuron units that have connections in between.

Feedforward Networks If the connections between neurons are only in a single direction, the network is called a feedforward network. In other words, a neuron can not be connected to itself or to another neuron that is closer to input. Formally,

outputs of $n-1^{th}$ layer are inputs of n^{th} layer while n^{th} layer's outputs are inputs of $n+1^{th}$ layer. Connections between neurons that are in the same layer or that are in nonconsecutive layers are not allowed in feedforward neural networks. An example of feedforward networks is given in Figure 3.3. Moreover, if a feedforward neural network has more than one hidden layers, it is called a *deep neural network*.

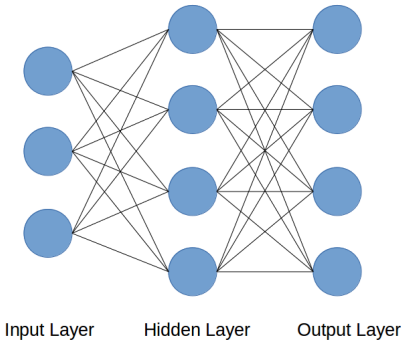


Figure 3.3: A feedforward neural network with one hidden layer.

3.1.2 Multilayer Networks

Multilayer networks are feedforward networks that has at least one or more hidden layers. Combining multiple neurons, we can represent a single layer as shown in Figure 3.4 and its input output relation is given in Equation 3.2. Note each neuron has its own bias that is not shown on the Figure 3.4. In order to make the calculation simpler, input-output relation is represented in matrix form as shown in Equation 3.3. Biases are merged into U and W matrices. When single layers of neurons are stacked, connecting outputs of one to inputs of another, a multilayer network is obtained as shown in Figure 3.3.

$$x_i = f(\sum_{j=1}^{N} u_j w_{ji} + \theta) \tag{3.2}$$

$$X = f(W^T U) \tag{3.3}$$

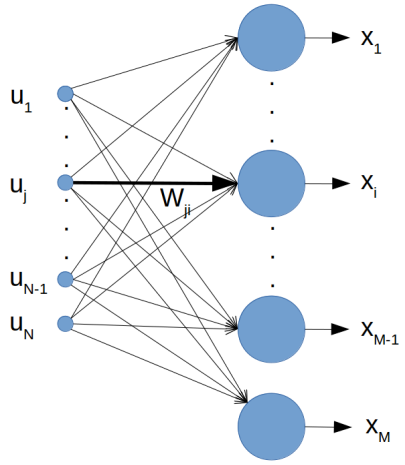


Figure 3.4: A single layer network which is mathematically represented in Equation 3.3.

where

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \quad U = \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}, \quad W = \begin{bmatrix} w_{01} & w_{02} & \dots & w_{0M} \\ w_{11} & w_{12} & \dots & w_{1M} \\ w_{21} & w_{22} & \dots & w_{2M} \\ \dots & \dots & \dots & \dots \\ w_{N1} & w_{32} & \dots & w_{3M} \end{bmatrix}, \quad \theta = \begin{bmatrix} w_{01} \\ w_{02} \\ \vdots \\ w_{0M} \end{bmatrix}$$

Where X is the output vector, U is the input vector, W is the weight matrix where the first row is equal to bias vector θ.

3.1.3 Optimization with Backpropagation in Neural Networks

Optimization is a crucial part of the neural networks, where the weights of the neurons are updated. Assume that the network has input u, output x and let its target output (or the ground truth) be t. The network should update its weights so that $x = t$. To do that, first a cost function such as mean square error needs to be introduced. Assuming n is the number of samples, mean square error can be shown as Equation 3.4 :

$$\varepsilon = \frac{1}{N} \times \sum_{j=1}^{N} (t_j - x_j)^2 \tag{3.4}$$

Several optimization methods exist to minimize Equation 3.4. For example, Newton's method, which actually is used to find the roots to a function, can be adapted by using the second order derivative. It increases the computational complexity significantly, compared to methods like gradient descent [75, 76] that use first order derivatives. This is why gradient descent is preferred for training most neural networks. Therefore, in this thesis, only gradient descent will be adopted and explained in detail.

3.1.3.1 Gradient Descent

Gradient Descent is an algorithm that uses the gradient of the function to iteratively find a (local) minimum. At each iteration, gradient of the current point is calculated. A step is taken proportional to the negative of the gradient. To understand it easier, the function may be thought as a valley and points in each iteration may be thought as a ball that is going down through this valley. When the ball is on the right wall of the valley, it will go towards left with a speed proportional to the slope. It is vice versa for the left wall. Figure 3.5 visualizes this example.

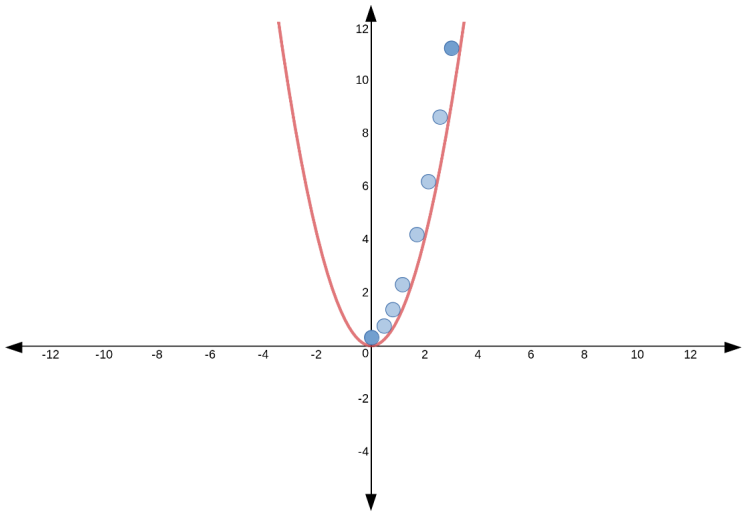


Figure 3.5: A simple example to visualize the gradient descent applied to $t = x^2$. The blue ball travels towards the lowest point of the red valley while its speed changes according to the slope.

When a single neuron is considered, assuming MSE is used as a cost function, the

update of weights at iteration i can be represented as in Equation 3.5.

$$w(i+1) = w(i) + \Delta w(i) \tag{3.5}$$

$$\Delta w(i) = -\mu \nabla \varepsilon(w(i)) \tag{3.6}$$

where μ is the learning rate and $\varepsilon()$ is the cost function with weights $w(i)$. Then, $\nabla \varepsilon(w(i))$ is given in Equation 3.10.

$$\nabla \varepsilon(w(i)) = \frac{\partial \varepsilon}{\partial w(i)} \tag{3.7}$$

$$x_j = w_j(i) u_j \tag{3.8}$$

$$\nabla \varepsilon(w(i)) = \frac{\partial (\frac{1}{n} \times \sum_{j=1}^{n} (t_j - w_j(i) u_j)^2)}{\partial w(i)} \tag{3.9}$$

$$\nabla \varepsilon(w(i)) = -\frac{2}{n} \times \sum_{j=1}^{n} (t_j - x_j) u_j \tag{3.10}$$

There are three approaches to apply gradient descent which are gradient descent by a single sample, full gradient(batch gradient) descent and mini batch gradient descent.

Gradient Descent by a Single Sample In this approach, the update function in Equation 3.6 is applied after each sample is fed into the network. After new weights are calculated, another sample is fed into network and error is calculated to update the weights again. The downside of stochastic gradient is it can be easily trapped in local minimum. Moreover, unstable results can be obtained since weights are updated by the response of a single sample rather than overall sample responses.

Full Gradient Descent In full gradient descent method, the network is fed with all training samples. After calculation of error for each sample, the network is updated once. Although it helps the model to escape the local minimums and the convergence is more stable compared to gradient descent by a single sample, the training time is generally very long since the weights are updated only once after all of the training

dataset is applied to the network. Additionally, feeding the whole dataset may not be possible due to memory constraints.

Mini Batch Gradient Descent In mini batch gradient descent method, the network is fed with a group of samples from the training dataset. The number of samples in the mini batch is predefined. This method tries to take advantage of both gradient descent by a single sample and full gradient descent by simply fusing them. The training time takes longer than gradient descent by a single sample but it can escape local minimums. Therefore, it is the most widely used method among the three. The batch size is chosen according to the hardware capacity, dataset size and experimental results.

When the gradient descent approaches to the minimum, it may start to oscillate in some dimensions as shown in Figure 3.6 since gradient direction of these dimensions may change after each iteration due to the noise exhibited from samples [77].

Momentum By adding a momentum parameter, oscillations stated above may be reduced. Improving the classical gradient descent Equation 3.6, the update with momentum becomes Equation 3.11 [78],[79].

$$\Delta w(i) = \delta \Delta w(i-1) - \mu \nabla \varepsilon(w(i)) \tag{3.11}$$

where δ is the momentum parameter.

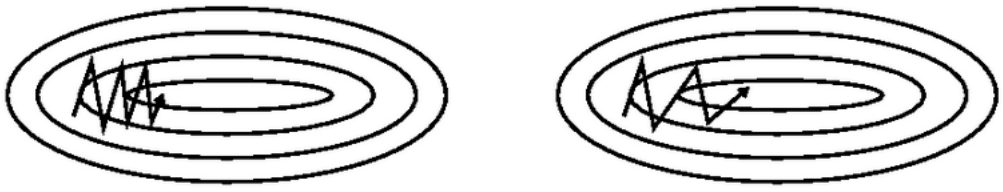


Figure 3.6: Gradient descent without momentum on left, gradient descent with momentum on right.(Image taken from [2])

Adam Optimizer Adam [80] is an improvement to stochastic gradient descent. By combining advantages of AdaGrad [81] (successful at problems with sparse gradients) and RMSProp [82] (successful at noise problems) algorithms, Adam optimizer is claimed to achieve better results. In machine learning, decreasing the learning rate as the iterations increase, is a common approach. Adam handles this by itself. It also assigns a different learning rate for each trained parameter. Additionally, Adam utilizes a moving average of the first and second momentums for updates. Another important feature of Adam is that prior to parameter update, it introduces a bias correction. This avoids large step sizes, and thus, divergence [80].

3.1.3.2 Backpropagation

In a neural network, there can be several neurons that have many weights. Thus, the optimization becomes more complex due to high number of unknowns. By using the chain rule, error is propagated from output through input [83], [78]. This process is called *backpropagation*.

Using backpropagation, an equation is found for each weight so that there are same number of equations and unknowns. This is done by propagating error through input. Then, the derivative of those propagated errors are calculated according to the weights correspondingly. The backpropagation on a multilayer network is performed by applying the chain rule to find the change in output x_k with respect to weight of j^{th} input of i^{th} neuron in L^{th} layer $w_{L,j,i}$. Figure 3.7 shows the structure so that the chain rule will be applied starting from the output layer. Using Equation 3.3, we can represent a single output x_k in terms of the weights in the last layer and outputs of the previous layer as shown in Equation 3.12.

$$x_k = f(W_{(L+1),k}^T x_L) \tag{3.12}$$

Where $f()$ is an activation function. We can also represent the output of a neuron in the hidden layer as in Equation 3.13.

$$x_{L,i} = f(W_{L,i}^T u) \tag{3.13}$$

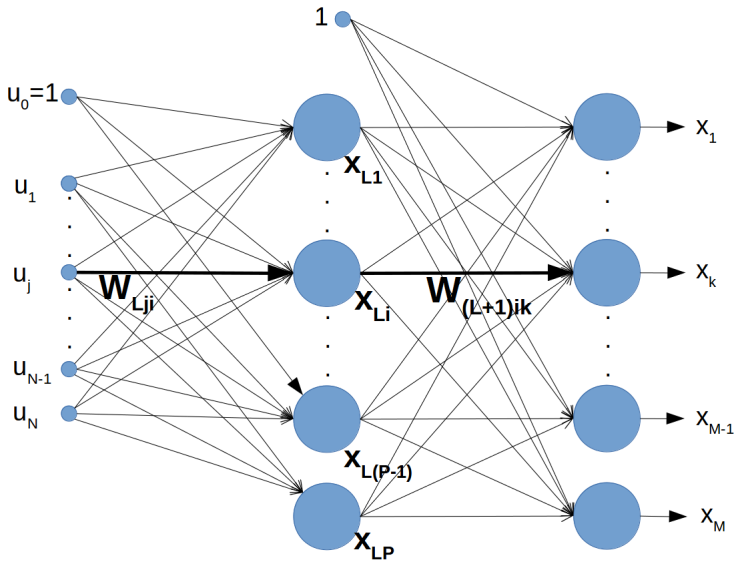


Figure 3.7: Backpropagation of x_k for $w_{L,j,i}$

In order to apply gradient descent to $w_{L,j,i}$, $\partial\varepsilon/\partial w_{L,j,i}$ is needed. ε is the MSE as given in Equation 3.4. By applying chain rule (Equation 3.16), it can be found.

$$\frac{\partial\varepsilon}{\partial w_{L,j,i}} = \frac{\partial\varepsilon}{\partial x_k}\frac{\partial x_k}{\partial w_{L,i,j}} \tag{3.14}$$

$$\frac{\partial x_k}{\partial w_{L,j,i}} = \frac{\partial x_k}{\partial x_{L,i}}\frac{\partial x_{L,i}}{\partial w_{L,i,j}} \tag{3.15}$$

$$\frac{\partial\varepsilon}{\partial w_{L,j,i}} = \frac{\partial\varepsilon}{\partial x_k}\frac{\partial x_k}{\partial x_{L,i}}\frac{\partial x_{L,i}}{\partial w_{L,i,j}} \tag{3.16}$$

Which, using Equations 3.12 and 3.13, becomes Equation 3.17

$$\frac{\partial\varepsilon}{\partial w_{L,j,i}} = -\frac{2}{M} \times (t_k - x_k) f'(W^T_{(L+1),k} x_L) w_{(L+1),i,k} f'(W^T_{L,i} u) u_j \tag{3.17}$$

However, when propagated backward, $w_{L,j,i}$ is affected by all of the weights $w_{(L+1),j,k}$ where k ranges from 1 to M. Thus, by addition of these partial derivatives, we obtain Equation 3.18

$$\frac{\partial\varepsilon}{\partial w_{L,j,i}} = -\frac{2}{M} \times \sum_{k=1}^{M} (t_k - x_k) f'(W^T_{(L+1),k} x_L)(w_{(L+1),i,k}) f'(W^T_{L,i} u)(u_j) \tag{3.18}$$

Solution of Equation 3.18 is simple if the calculation of derivative of the activation function $f()$ is easy. This is why functions like rectified linear function is preferred as

activation functions in neural networks. These will be explained in detail in Section 3.1.5.5.

3.1.3.3 Loss Functions

As seen from the Eq. 3.6 to Eq. 3.18, the error function used for the optimization affects the performance importantly. Selection of a loss function depends on the task. For example, the mean square error (Eq. 3.4) or mean absolute error can be used for regressive problems, which aim to reach a continuous variable. Similarly, for classification problems, where the target value is discrete(1 if belongs to that class, 0 otherwise), error functions such as likelihood and cross entropy loss are used. More specifically, a differentiable version of dice score can be adopted for image segmentation problem. Because of the scope of this thesis, loss functions that can be used for the segmentation problem is given in detail.

Cross Entropy Loss For image classification and image segmentation, most widely used cost function utilizes cross entropy function (Eq. 3.19) which, similar to mean square error, measures the distribution characteristics of two representations.

$$D(S, L) = -\sum_{i}^{C} L_i \log S_i \tag{3.19}$$

where C is the number of classes, S is the prediction (output of the softmax) and L is the ground truth label. For a segmentation case, the cross entropy is calculated for each pixel and reduced to a scalar. In order to turn this into a cost function, one must consider both class members where $L_i = 1$ and non-class members $L_i = 0$. Thus, the equation becomes 3.20.

$$D_{loss}(S, L) = -\sum_{i}^{C} L_i \log S_i - \sum_{i}^{C} (1 - L_i) \log (1 - S_i) \tag{3.20}$$

For the image segmentation problem, reduction can be done by operations such as summation or averaging of each pixel.

Weighted Cross Entropy In order to solve the class imbalance problem, one can use Eq. 3.21 where W_i corresponds to the weight that belongs to i^{th} class. W_i values can be chosen inversely proportional to number of samples in i^{th} class.

$$D_{loss}(S, L) = -\sum_{i}^{C} W_i L_i \log S_i - \sum_{i}^{C} (1 - L_i) \log (1 - S_i) \tag{3.21}$$

Dice Loss Another loss function used for image segmentation is the Dice Loss Function (Eq. 3.22). It is similar to dice coefficient which is explained in Section 6.3.1. The only difference is that instead of using sparse results, softmax results are used to calculate the dice coefficient so that it becomes differentiable. Note that two of the top performing methods proposed in BraTS 2017 challenge use dice loss [52, 43].

$$DiceLoss = -\frac{2}{|C|} \sum_{i}^{C} \frac{\sum_k S_i^k L_i^k}{\sum_k S_i^k + \sum_k L_i^k} \tag{3.22}$$

Where C is the number of classes and k is every pixel in the image.

Another loss adapted in this work is adversarial loss which comes from adversarial networks. The details of the adversarial loss is given in Section 3.1.5.8.

3.1.4 A Brief History of Convolutional Neural Networks

Artificial neural networks are inspired by biological neural networks. Likewise, convolutional neural networks are inspired by the visual system of animals. In the paper [84], Fukushima explains a multilayered artificial neural network architecture he designed, called Neocognitron. In the proposed design, there are two types of cells, namely S-cells and C-cells, which are similar to "simple cells" and "complex cells" both of which were cell types proposed by Hubel and Wiesel in [85] to explain a cat's visual system. In Neocognitron architecture, first, basic features are locally found by S-cells. Then those local features are combined by C-cells and more complex features are obtained. By doing so, his architecture achieved recognizing different patterns independent of their spatial shift. This architecture can be an important inspiration for convolutional neural networks.

Although CNNs have been used in the literature for a while [83], they were not used widely until the success of [3] because of two main reasons. The first reason was

Table 3.1: Comparison between a high end personal computer GPU, a high end workstation GPU, a high end personal computer CPU and a high end workstation CPU

	GTX 1060	Quadro P6000	Intel i7 7700k	Intel Xeon E5-2699 v4
Number of Cores	1280	3840	8	22
Clock Speed	1507 MHz	1607 MHz	4.2 GHz	2.2 GHz
Maximum FLOPS	4375 GFLOPS	12000 GFLOPS	150 GFLOPS	634 GFLOPS

the lack of data to train the network. Second was that their training time was so long due to the lack of process parallelization. With the increase in the number of photographs uploaded on the Internet everyday, researchers have overcome the first problem. The second problem was solved by the advances in the GPU technology. Human brain has an average number of $8.6 * 10^{10}$ neuron cells and its power comes from being massively parallel. Being such parallelizable, the architecture of GPUs is more convenient to human brain than that of CPUs. There are many simple cores in a GPU. In order to compare common products of both at the time of writing, an *Intel i7 7700k* has a clockspeed of 4.2 GHz while an *NVIDIA GTX 1060* has a clock speed of 1500Mhz. On the other hand, *GTX 1060* has 1280 cores while *i7* has 4 cores. As similar to human brain, higher number of weaker cores are more suitable for neural networks. A simple comparison between GPUs and CPUs can be observed in Table 3.1.

3.1.5 Layers of CNNs

CNNs share some general characteristics of neural networks, such as optimization methods, loss functions etc. This section focuses on non general characteristics of CNNs. AlexNet [3], a state of the art image classification network, is given(Figure 3.8) in order to show convolution layers, pooling layers and fully connected layers. Since

it is a classification network, it does not have upsampling layers. Also the fully connected(dense) layers at the end become $1x1$ convolutional layers in the segmentation case. More details are given in Section 3.1.5.4.

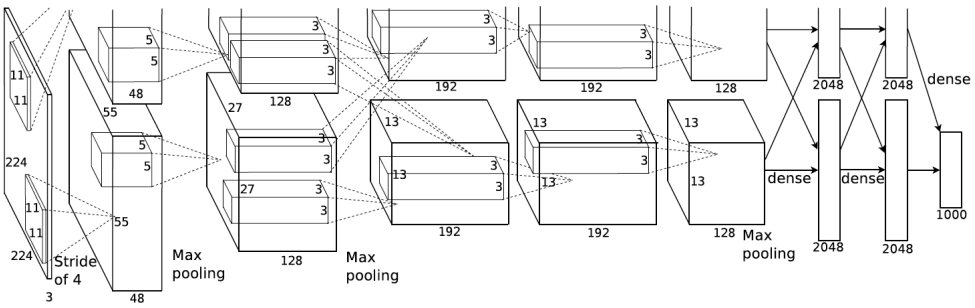


Figure 3.8: AlexNet (Image taken from [3])

Each box corresponds to features obtained by convolutions. Dashed lines show convolution layers. Max pooling and stride is written below if applied. At the end there are 3 fully connected(dense) layers shown by solid lines.

3.1.5.1 Convolution Layers

In convolutional neural networks, the most important layer type, which also gives the architecture its name, is convolution layers. The convolution layer actually corresponds to a special type of feedforward network layer, where each pixel p_{ij} corresponds to an input u_j and each value w_{ij} of the convolution filter corresponds to a weight w_j. Assuming a filter size of 3×3 and an input size of $d \times n$, there are $(d-2) \times (n-2)$ neurons with $(d-2) \times (n-2)$ number of outputs. Unlike a traditional network, which would have $(d-2) \times (n-2)$ number of different weight matrices, there is only one weight matrix that is shared among all neurons in a convolution layer. This is the most important characteristic of a convolutional neural network. What makes shared weights possible is that the features do not change according to spatial positions.

The transfer function Equation 3.1 corresponds to a convolution in a CNN. A simple 2D convolution with 3×3 weight matrix is formulated in Equation 3.24 (Note that

this equation does not take the symmetric of the filter unlike the general 2D discrete convolution formula in Equation 3.23). Basically, one may refer to a convolution filter as an artificial neuron given in Figure 3.10. This operation is applied on whole image by sliding the filter as shown in Figure 3.9.

$$C_{ij} = \sum_{k=-1}^{1} \sum_{l=-1}^{1} A_{(i-k)(j-l)} B_{(k)(l)} \tag{3.23}$$

where $1 < i < D$ and $1 < j < N$ and dimensions of matrix A is $D \times N$

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \dots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \dots & p_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ p_{d1} & p_{d2} & p_{d3} & \dots & p_{dn} \end{bmatrix} \qquad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

$$R = \begin{bmatrix} r_{22} & r_{23} & r_{24} & \dots & r_{2(n-1)} \\ r_{32} & r_{33} & r_{34} & \dots & r_{3(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ r_{(d-1)2} & r_{(d-1)3} & r_{(d-1)4} & \dots & r_{(d-1)(n-1)} \end{bmatrix}$$

$$r_{ij} = \sum_{k=-1}^{1} \sum_{l=-1}^{1} p_{(i+k)(j+l)} w_{(2+k)(2+l)} \tag{3.24}$$

where $1 < i < d$ and $1 < j < n$

Note that the Equation 3.24 represents a convolution of a 1 channel only. If there are multiple channels, then the filter W becomes a 3D matrix with the same number of channels.

In other words, convolution layers consist of filters that will be applied on each channel of the input. Since applying a filter on an image is basically a convolution, these layers are named convolution layers. By applying filters, features of an image that correspond to each filter is found. Similar to the architecture proposed in [84], as the layers go deeper, the features become more complex. For example, assume that in the first layer, vertical and horizontal edges of an image are found. In the second layer, the outputs of the first layer, which correspond to vertical and horizontal edges of the

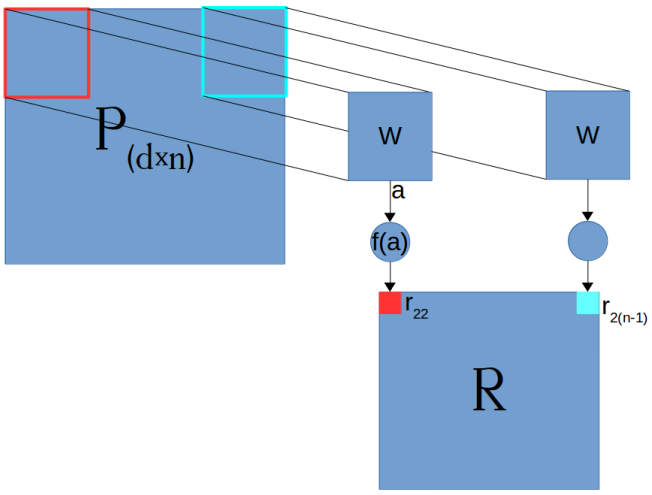


Figure 3.9: Convolution is applied on the entire image.

The red rectangle in P represents 3×3 pixels on the upper left corner. The result is the red rectangle in R and it is actually r_{22} in Figure 3.10 .

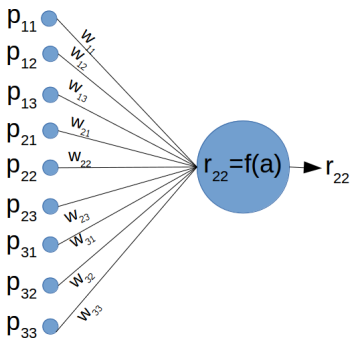


Figure 3.10: A neuron representing the filter W applied on pixels around p_{22}.

image, are used to find the corners in the image. Of course in real applications the features get much more complex in the deeper layers of a network.

3.1.5.2 Pooling Layers

Pooling layers are usually used to reduce the dimensions of the output of a convolution layer. These layers are needed because number of features obtained from a convolution layer can be undesirably high. For example, assuming zero padding on the edges so that the convolution result has the same size with the image, an RGB

image of size $H \times W \times C$ would have $H \times W \times (number\,of\,filters\,in\,the\,layer)$ features. Training on such high number of features would result on problems like overfitting. Therefore, dimension reduction is needed. There are many methods that can be used to reduce the dimensions, such as average pooling and max pooling. The most preferred one is max pooling because it extracts the feature that is dominant in that local region where the pooling is applied. This is particularly critical since this module allows to propagate the error with the highest responses so that it decreases the chance of vanishing gradient problem which might be frequently observed for the averaging operations. Figure 3.11 shows max pooling and average pooling applied on 2×2 windows with stride 2.

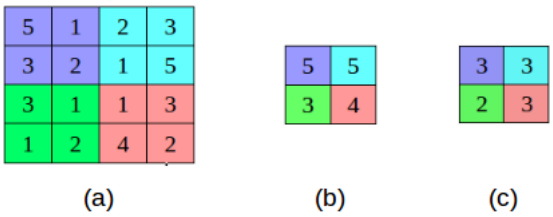


Figure 3.11: (a) Image pixel values. (b) Result of max pooling for each colored region. (c) Result of average pooling for each colored region.

3.1.5.3 Upscaling Layer

Not always downsampling is needed in neural networks. Upsampling is also needed in neural networks for tasks like pixel-wise classification, or semantic image segmentation. After reducing the spatial dimensions as in classical classification task, a CNN built for pixelwise classification needs to construct an output with the same spatial dimensions of input. Thus, upsampling layers are used to increase the dimension of the deep features. There are many methods used as an upsampling layer in CNNs. Bilinear interpolation, deconvolution and subpixel upsampling are some of the important ones.

Bilinear Interpolation Bilinear interpolation is the simplest method to upscale a 2D image. It is an extension of linear interpolation. In linear interpolation, the effect of a known point, on the value of the target point is inversely proportional to the

distance between them. The equation of the linear interpolation is given in Eqn. 3.25. It actually finds a point on a line, given one of its coordinates and the equation of the line.

$$y_3 = y_1 + \frac{(x_3 - x_1) * (y_2 - y_1)}{(x_2 - x_1)} \tag{3.25}$$

Bilinear interpolation extends this equation to 3D space, considering the 2D coordinates and the pixel values. It can be seen in Figure 3.12

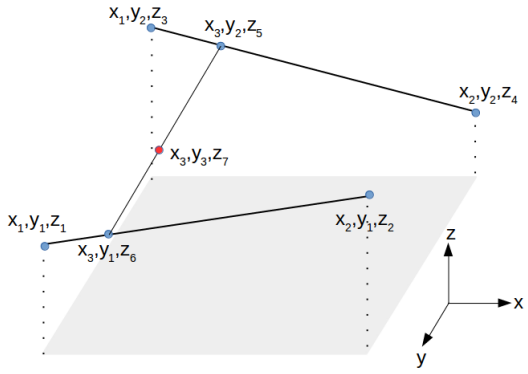


Figure 3.12: Bilinear interpolation

Deconvolution In signal processing, deconvolution is the inverse function of convolution operation. Let $f()$ be the convolution function, I be the input and Y be the output. The inverse function $f^{-1}()$ is the deconvolution which gives the output I if its input is Y. However, the deconvolution term is not used for the actual deconvolution operation in the neural networks. It actually corresponds to convolution operation which is used for upsampling. It is sometimes called as transposed convolution or backward convolution too. [34] states that upsampling with a factor f corresponds to convolution applied with a fractional stride $\frac{1}{f}$. Bilinear interpolation is also a deconvolution, but with constant weights. The advantage of using a $deconvolution$ layer for upsampling is that it has the learning capability, unlike methods like bilinear interpolation. The deconvolution operation (as used in neural networks) applied to obtain an upscaled output can be illustrated as shown in Figure 3.13.

[86] proposes a network in which the deconvolution layers are preceded by unpool-

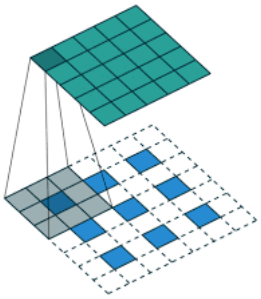


Figure 3.13: Deconvolution applied on a 3×3 image (blue squares) padded with zeros (white squares) to obtain a 5×5 image (green squares) (Image taken from [4])

ing layers. Therefore, instead of padding zeros around each pixel, the convolution operation is applied on unpooled images. These unpooled images are obtained by memorizing the pooling positions that corresponds to the same level or same size.

Sub-pixel Upsampling Sub-pixel convolutional layers are proposed in [5] for super-resolution. The architecture consists of convolution layers to extract features and a sub-pixel convolutional layer to obtain an image with bigger size. The architecture can be seen in Figure 3.14 taken from [5]. The sub-pixel convolution layer actually takes the features (or channels) and pads them in a specified order to obtain an up-scaled image. For example, assume there are 4 channels, pixels of which are named $p_{1(x,y)}$, $p_{2(x,y)}$, $p_{3(x,y)}$, $p_{4(x,y)}$ respectively. w corresponds to width and h corresponds to height of these feature channels. Then the result of sub-pixel operation becomes R as shown below.

$$R = \begin{bmatrix} p_{1(1,1)} & p_{2(1,1)} & \cdots & p_{1(w,1)} & p_{2(w,1)} \\ p_{3(1,1)} & p_{4(1,1)} & \cdots & p_{3(w,1)} & p_{4(w,1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{1(1,h)} & p_{2(1,h)} & \cdots & p_{1(w,h)} & p_{2(w,h)} \\ p_{3(1,h)} & p_{4(1,h)} & \cdots & p_{3(w,h)} & p_{4(w,h)} \end{bmatrix}$$

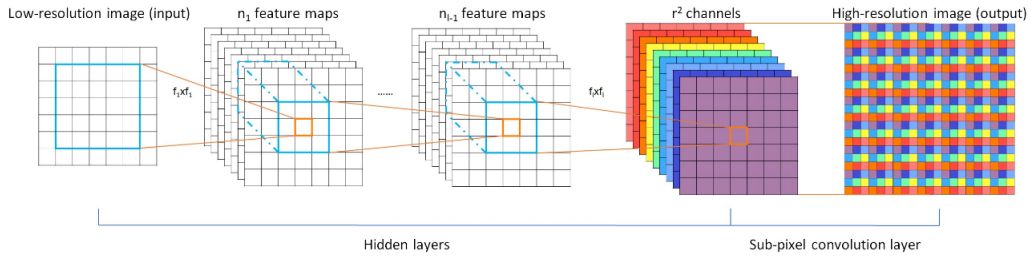


Figure 3.14: Visualization of sub-pixel upsampling. (Image taken from [5])

3.1.5.4 Fully Connected Layers

Fully connected layers use the feature information to reach a final class label. Nodes of fully connected layers are connected to each output of the preceding layer and their outputs are connected to each node of the following layer. In the last layer, there are n number of nodes, where n equals to the number of classes 3.8. The nodes in the last layer of the fully connected layers generally have a special activation function that will generate a probabilistic result for each label. This activation function is generally a softmax function, or a sigmoid function in binary classification case.

Dense Layers in Fully Convolutional Networks If the aim of the network is to perform segmentation, it is a fully convolutional network. The dense layer at becomes a 1×1 convolution layer with a different activation function. In the classical classification problem, the output layer is a vector with a single dimension equal to the number of classes. In the pixelwise classification problem, on the other hand, the output is a 3D matrix. Its width and height is equal to those of input image while its depth is equal to the number of classes. So, one might say that there is an output vector for each pixel in the image, resulting in an output of dimension $Width \times Height \times NumberOfClasses$. Particularly, one can represent a fully connected layer in a classifier by 1×1 convolutions applied to single pixel images. In this case, the output's dimensions are $1 \times 1 \times NumberOfClasses$.

3.1.5.5 Activation Functions

After the input of an artificial neuron is multiplied by the weights and summed to a scalar, the result is fed into an activation function $f()$ (Figure 3.2). In this section, different activation functions will be given and their advantages and disadvantages will be discussed.

Softmax Softmax function (Equation 3.26) is usually used in the output layer of multilayer neural networks that are built to do classification task. It actually calculates the probability of each class. Since we are speaking of probabilities, the sum of all softmax results in an output layer equals to 1. Instead of making a strict choice between classes, softmax estimates which class the input more likely belongs to. Assuming there are M classes, softmax can be represented as:

$$softmax(x_k) = \frac{e^{x_k}}{\sum_{i=1}^{M} e^{x_i}} \tag{3.26}$$

Sigmoid Sigmoid function is also limited between 0 and 1.Its main disadvantage is that its gradients start to vanish for values that are too large or too small. In other words, the training gets slower as the values move away from 0 and saturates [3].

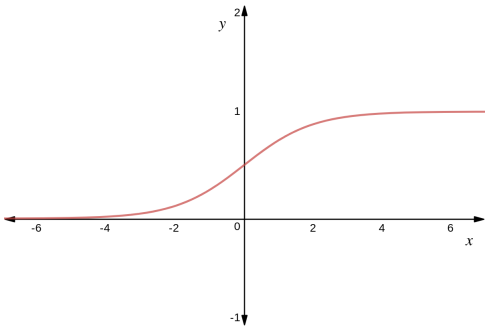


Figure 3.15: Sigmoid function

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.27}$$

Tanh tanh function (Eqn. 3.28, Figure 3.16) is actually very similar to sigmoid function but instead of being limited between 0 and 1, it is limited between -1 and 1.

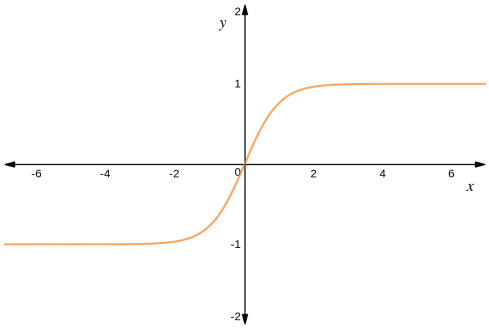


Figure 3.16: tanh function

$$tanh(x) = 2sigmoid(x) - 1 = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{3.28}$$

Rectified Linear Until publishing of [3], it was believed that differentiable, non linear, symmetric functions were better to use in neural networks. However, [3] has shown that rectified linear units(ReLU) makes training a neural network computationally more efficient and faster. Due to its being differentiable everywhere except 0, it is easy to calculate the gradient descent (Eqn. 3.29) for ReLUs. At 0, on the other hand, its derivative is accepted and defined by the user as either 0 or 1. Additionally, since the function does not saturate, the gradients do not vanish for very large values. Therefore, the training is faster.

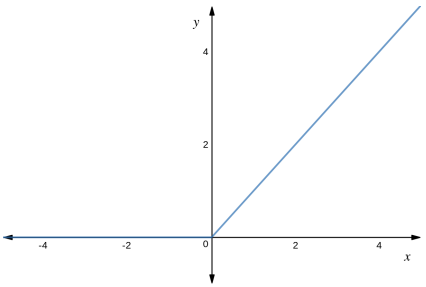


Figure 3.17: ReLU function

$$ReLU = max(0, x) \tag{3.29}$$

Leaky Rectified Linear Rectified linear units somewhat ignores the negative input values. Any negative valued input results in a 0 output and the gradient is 0. In order

to keep the constant gradient and to avoid the dying problem, leaky rectified linear units (LReLU) are used. In LReLUs, the function becomes Eq. 3.30

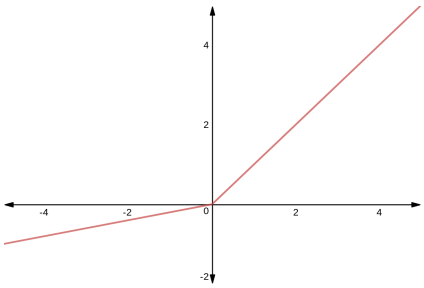


Figure 3.18: LReLU function as given in Eq. 3.30(where α is 0.2)

$$LeakyReLU = max(\alpha x, x) \tag{3.30}$$

where α is a positive constant smaller than 1. Note that if α is trainable value, the function is called *parametric rectified linear unit (PReLU)*.

3.1.5.6 Dropout

Dropout is training a neural network by ignoring some of the neurons at certain iterations Figure 3.19. This helps prevent the overfitting [6]. The effect of ignored neurons on other neurons is reduced. Thus, non ignored neurons learn better.

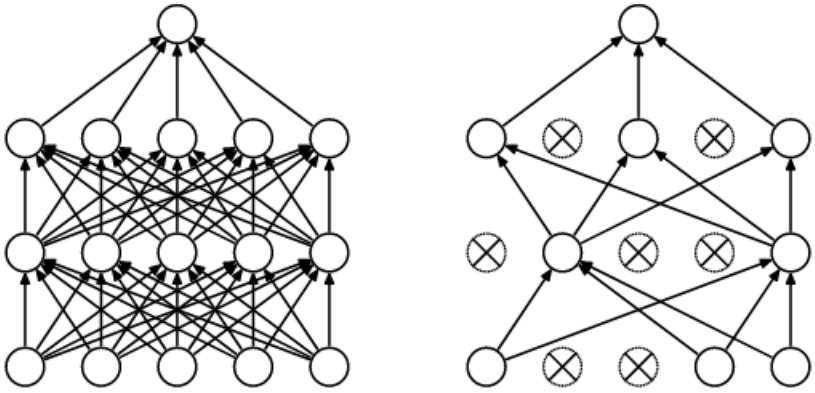


Figure 3.19: Left: A standard neural network. Right: A network with dropout. Crossed neurons are ignored for current iteration. (Image taken from [6])

3.1.5.7 Batch Normalization

Since the datasets used in deep learning are huge, the input of network varies significantly. Batch normalization is, as the name suggests, normalizing the data in the mini batch and making the normalization part of the network, instead of normalizing the whole dataset as a pre-processing method. Thus, in tasks like image classification, it increases learning speed [87]. Additionally, [87] states that the need for dropout layers may be excluded since batch normalization acts as a regularizer.

3.1.5.8 GAN:Generative Adversarial Networks

Generative adversarial networks were proposed in [69] for generating images, that are similar to the ones in the dataset, from random inputs. It consists of two CNNs, namely, generator and discriminator. The generator is trained to produce a result that is alike one of the samples in the provided dataset. It can also be said that it tries to convince the discriminator so that it confuses the fake data to be the real data. The discriminator, on the other hand, tries to distinguish between real samples and artificially generated results. This game between discriminator and generator can be represented as in Eqn. 3.31 In the segmentation problem, instead of a random matrix, generator takes an image as the input and produces a segmentation result while the discriminator takes both the ground truth label and generator's result. Discriminator then distinguishes the segmentation result from the ground truth. Each of the networks are trained one after the other at each step, keeping the parameters of the other network constant.

$$\min_G \max_D L(G,D) = \mathbf{E}_{x\sim p_{GT}(x)}[log(D(x))] + \mathbf{E}_{z\sim p_{input}(z)}[log(1-D(G(z)))] \quad (3.31)$$

where z is the MRIs (inputs) and x is the ground truths. So, while G tries to minimize the second term to fool D, D tries to maximize both terms to discriminate between real and fake data.

The loss function of the generator is updated by the addition of $\alpha\mathbf{E}_{z\sim p_{input}(z)}[log(1 - D(G(z)))]$ where α is small enough so that the actual loss of the generator keeps its significance. This addition of adversarial loss to the segmentation loss(cross entropy or dice loss) acts as a regularization.

The general network structure is given in Figure 3.20. The generator part is already explained in the previous chapters, while discriminator can be seen in Figure 3.21. Note that the discriminator needs to produce 1 if the ground truth is fed into the network. Otherwise, it needs to produce a 0. Hence, discriminator is actually a binary classifier. It has two fully connected layers at the end. Output of the last layer is fed to a sigmoid function to generate a result between 1 and 0.

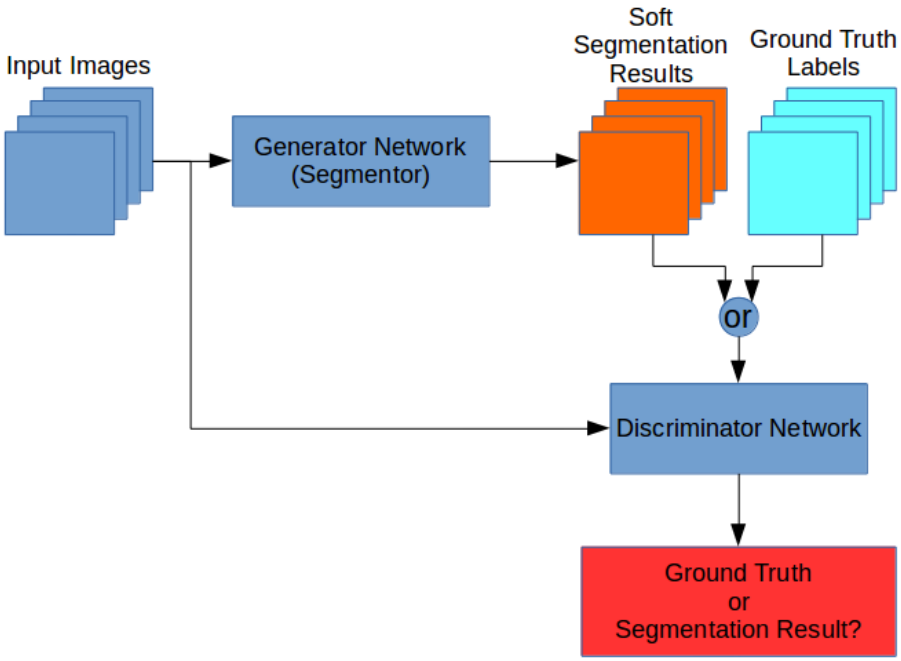


Figure 3.20: General structure of GAN used for segmentation problem. The generator is the segmentation network explained in previous chapters. Discriminator network is given in 3.21

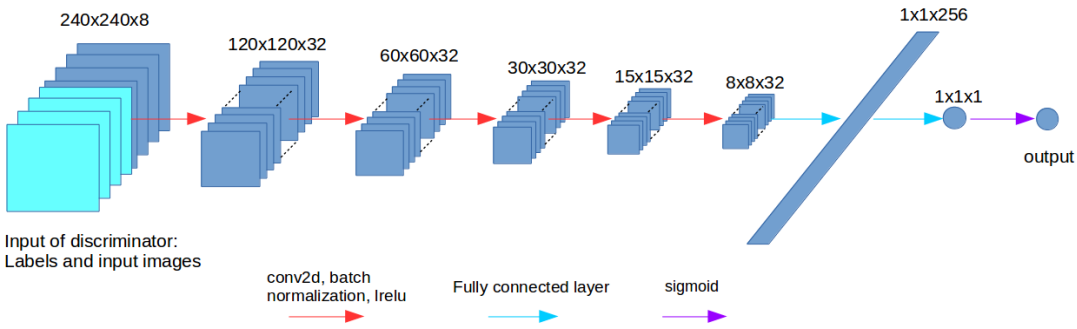


Figure 3.21: Discriminator network architecture used in GAN

CHAPTER 4

CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES USED FOR BRAIN MRI SEGMENTATION

In this chapter, different network architectures, namely, FCN-8s and U-Net will be explained. Additionally, some modifications that improve the performance of these networks are given. The performances of them will be given in Chapter 6. Note that another popular segmentation architecture, DeepMedic [35], is not used due to its patch based approach.

4.1 FCN-8s

FCN-8s is a Fully Convolutional Neural Network proposed in [34] for semantic segmentation. Except the last layers, it is not different than a classical classification CNN. The aim of the last layer is to combine features of different scales to localize the complex features. While the original paper has 3 different networks -FCN-32s, FCN-16s and FCN-8s- FCN-8s (Figure 4.1) is proven to have the best performance among them. Hence, it is compared with U-Net architecture in this thesis. It would not be wrong to say that U-Net is an extended, deeper version of FCN-8s.

4.2 U-Net

U-Net is a Fully Convolutional Neural Network proposed in [36], of which main aim is medical image segmentation. It has n number of convolution layers followed by pooling layers and n number of deconvolution layers to reconstruct a segmentation output that has the same dimensions with the input. It gradually downsamples the

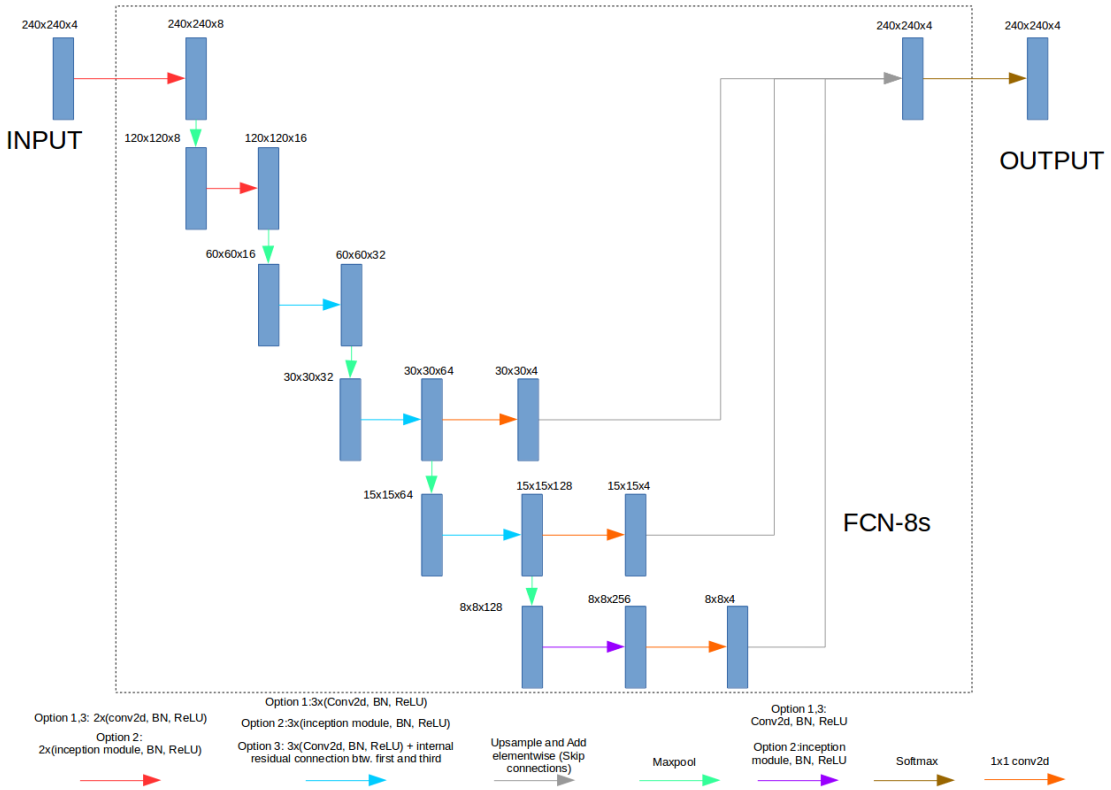


Figure 4.1: FCN-8s architecture used for brain tumor segmentation in this work

input and finds its deep features. Then it upsamples those features by deconvolution and concatenates these upsampling results channel-wise with the features that has the same dimensions. An example architecture to U-Net can be seen in Figure 4.2.

The deeper features have a global meaning but they lack the local information needed for pixel level classification. Re-introducing the outputs of shallow layers by concatenation provides the local information. Additionally, they are similar to skip connections with an only difference that is skip connections are connected as additions instead of concatenations. Lets consider Figure 4.2. We can say that the features in the bottom of the U-shape are complex and represent global information. By introducing the simpler features obtained in the encoder to the decoder, complex features are combined with local information. In other words, the encoder is first used to find complex global features, then the information in decoder is combined with these features for localization. At the top of the decoder, 120 features are obtained with the same spatial dimensions of the input, i.e. each pixel has 120 features of its own.

Note that the number of layers in a U-Net architecture may vary according to the input dimension and complexity of the task. [55]

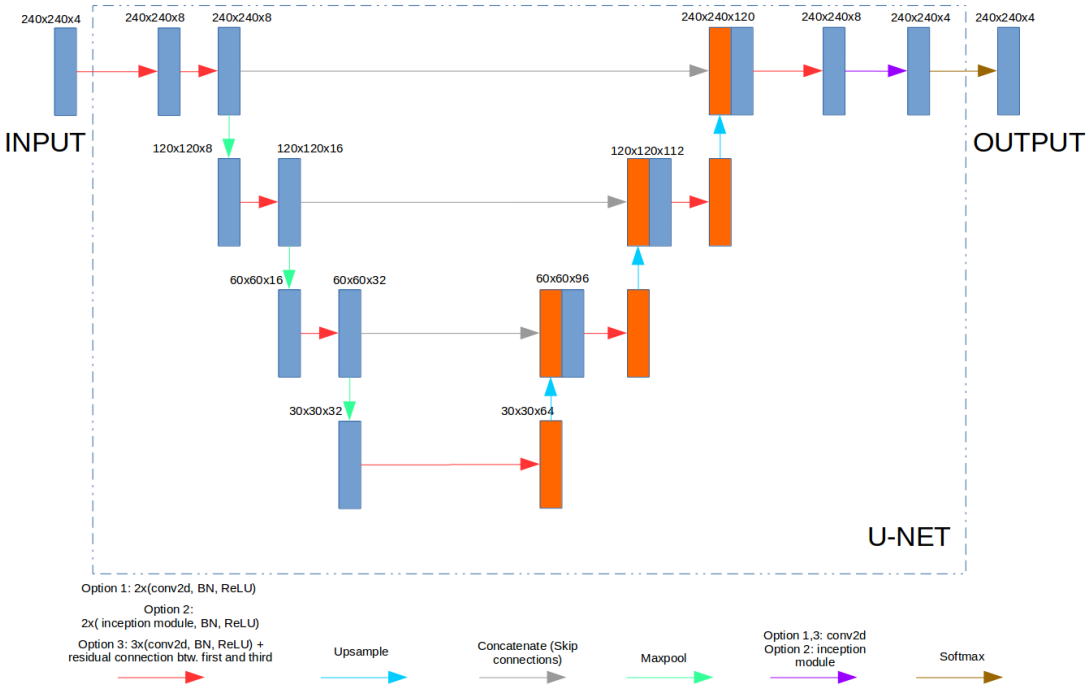


Figure 4.2: U-Net architecture used for brain tumor segmentation

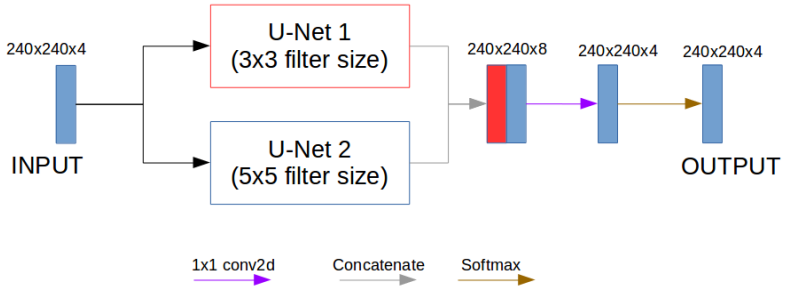


Figure 4.3: Multi-scale U-Net (with 3x3 and 5x5 filter sizes)

There are many modifications that can affect the performance of the network. These include new connections and connection paths, filter size, input dimension, training methods etc. Section 4.3 explains some of these modifications in detail. This section, on the other hand, focuses on how these modifications are applied on U-Net for brain tumor segmentation problem and how these modifications affect the performance.

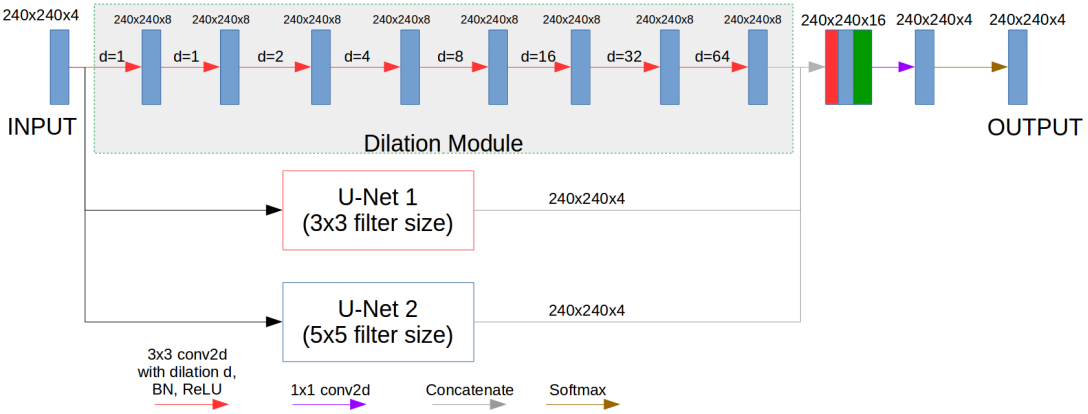


Figure 4.4: Multi-scale U-Net (with 3x3 and 5x5 filter sizes) and a dilation module

4.3 Proposed Modifications

Although the CNN based semantic segmentation methods has been proven to perform better than classical approaches, they still need some melioration. In this section, improvement methods that can be applied to CNN based segmentation problem are explained.

4.3.1 Inception Modules

Inception modules can be thought as an improvement to the convolution layers by making an architectural modification. In a classical convolutional layer, the filter size is chosen by the user while building the network. However, in inception modules, there are multiple filter sizes as shown in Figure 4.5. Thus, instead of choosing the filter size of the corresponding layer, the user lets the network decide the priorities of the filter sizes [59].

4.3.2 Dilated Convolutions

By inserting zeros between the elements of a filter, dilated convolutions are obtained. For example, if 2 zeros are inserted, then the dilation is 2. Red dots in Figure 4.6 shows the dilated convolution filters with dilations $1, 2, 3$ respectively. The blue squares on the other hand shows the receptive field of each filter, applied on the output

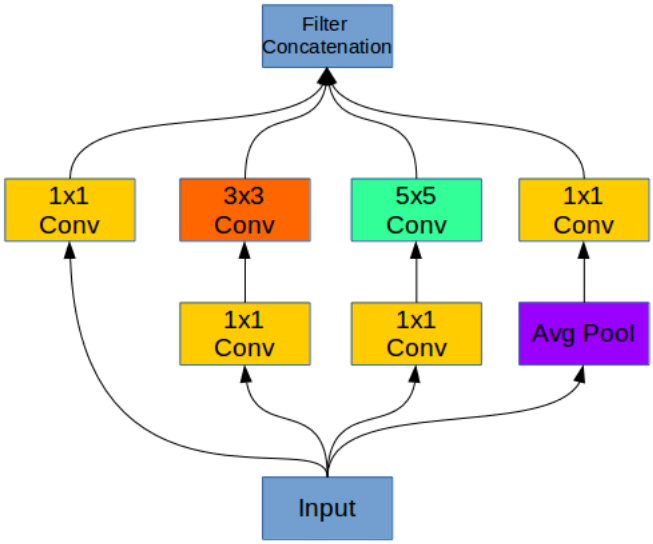


Figure 4.5: An Example of Inception Module

of the previous convolution. The main advantage of the method is that when dilation is increased in each layer, the receptive field increases exponentially without resolution or coverage loss [7]. This approach has been used by [88] in the brain MRI segmentation problem. This work utilizes a dilation module with less features (8) for each layer than the one used in [88] (32). (Figure 4.4).

Although it seems like a simple application, it requires some fine tunings such as initialization of the network parameters. In [7], it is suggested that the network should be initialized either using a form of identity initialization or by using the parameters of the original network without dilated convolutions.

4.3.3 Residual Connections

Residual connections (Figure 4.7 (a)) are shortcuts that link the responses of previous layers to subsequent layers. Their main aim is to avoid the vanishing gradient problem by providing a shortcut connection for the error during backpropagation. Proposed in [57], residual connections make deeper networks possible by also preventing overfitting to some extent. Moreover, they make the optimization and training easier.

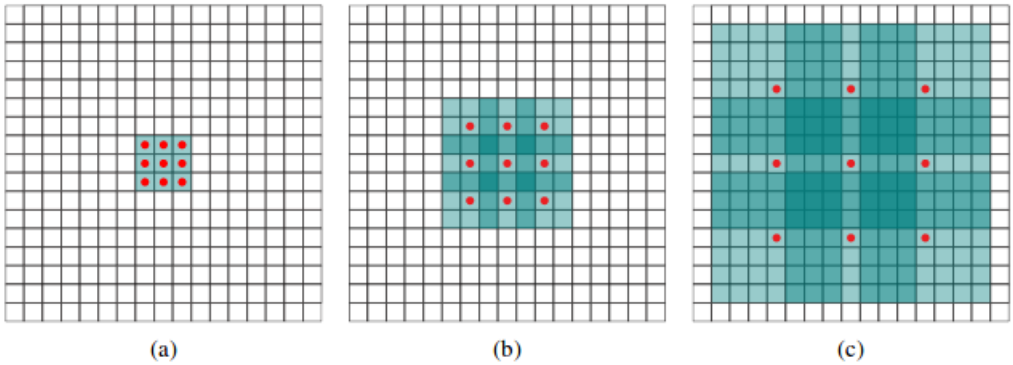


Figure 4.6: The blue areas show the receptive field of each layer (a) Normal convolution (b) Dilated convolution with dilation 2 (c) Dilated convolution with dilation 4 (Image taken from [7])

4.3.3.1 Pre-activation Residual Block

Pre-activation residual block is a modified version of the original residual connections proposed in [57]. The only difference is that instead of connecting the input prior to batch normalization to the output of the second next filter, it connects the input of the filter to the output of the second next batch normalization [8]. The difference can be seen in Figure 4.7.

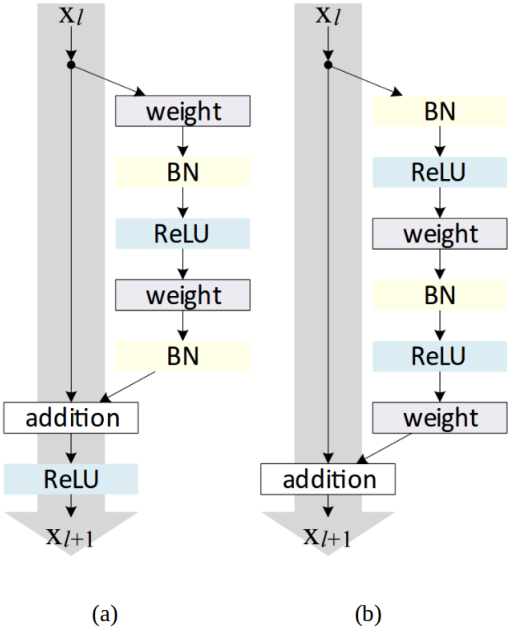


Figure 4.7: (a) Original post-activation residual block (b) Pre-activation block (Image taken from [8])

CHAPTER 5

PRE-PROCESSING AND POST-PROCESSING METHODS

In order to provide generalization for datasets with different data ranges and different characteristics, pre-processing is applied.

5.1 Pre-Processing Methods

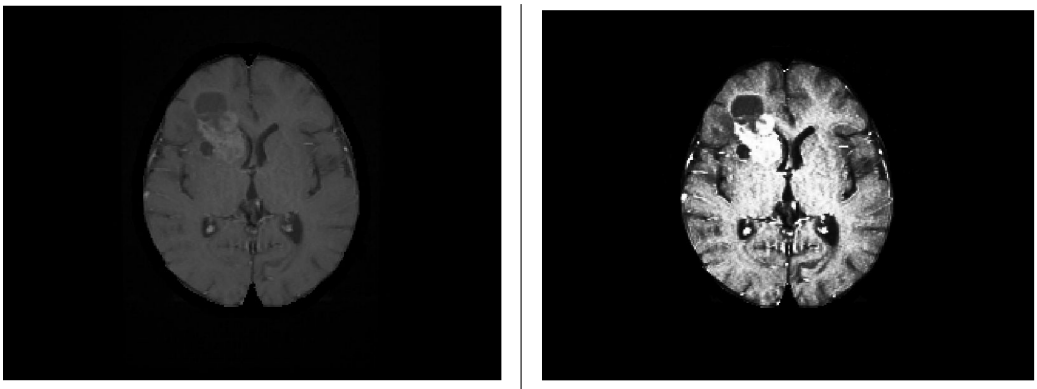
Pre-processing is a crucial step not only for image/signal processing but all kinds of data. Given a dataset, pre-processing standardizes and makes it more meaningful for an algorithm. There are many pre-processing methods in the literature; however, only few of them, namely, normalization, histogram matching and histogram equalization can be applied on brain MRIs.

5.1.0.1 Normalization

Normalization is a widely used pre-processing step which aims to scale the data between a given maximum and a minimum. It can be both linear (Eqn. 5.1) and non-linear (Eqn. 5.2). An example of linear normalization is given in Figure 5.1.

$$x_{normalized} = (desiredMax - desiredMin)\frac{(x - min(x))}{max(x) - min(x)} + desiredMin \tag{5.1}$$

$$x_{normalized} = \frac{desiredMax - desiredMin}{1 + \exp(-\frac{x-\beta}{\alpha})} + desiredMin \tag{5.2}$$



(a) Original Image (b) Normalized Image

Figure 5.1: An example image and normalization applied on it.

5.1.0.2 Histogram Matching

In order to standardize the images, their histograms may be matched to the histogram of a target image. In brain tumor segmentation problem, this approach may be problematic since not all patients have a brain tumor. Thus, even if the brain image does not have a tumor region, the pixel intensities that are supposed to be specific to the tumor region may arise after the histogram matching is applied.

5.1.0.3 Histogram Equalization

The aim of this method is to have a more uniform histogram so that the contrast of the image increases. However, this also affects the data. For example, background data becomes less distinguishable. In brain tumor segmentation problem, it should be applied with a foreground mask in order to prevent such artifacts. Figures 5.2 and 5.3 shows examples of histogram equalization without and with a foreground mask respectively. A common method to produce an equalized histogram is to use a mapping function that obtains the most possible linear ramp shaped cumulative distribution function.

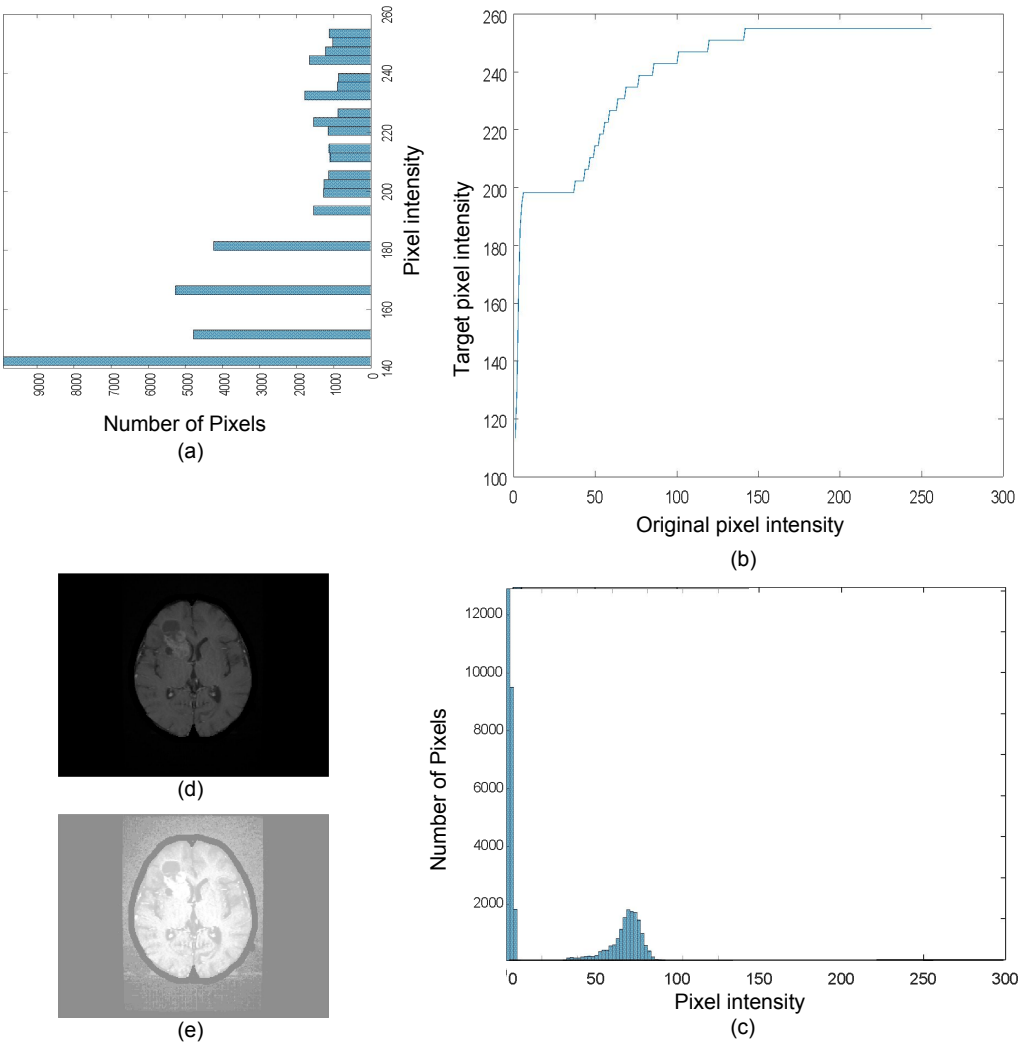


Figure 5.2: Histogram equalization applied **without** a foreground mask.

(a) is the resulting histogram (b) is the mapping function to map pixel intensities in the original image (c) histogram of the original image (d) original image (e) histogram equalization applied image. Note that in order to obtain an equalized histogram that has an almost linear cumulative distribution function, all of the zero pixels are mapped to 140. Thus, the resulting image (e) loses a lot of information.

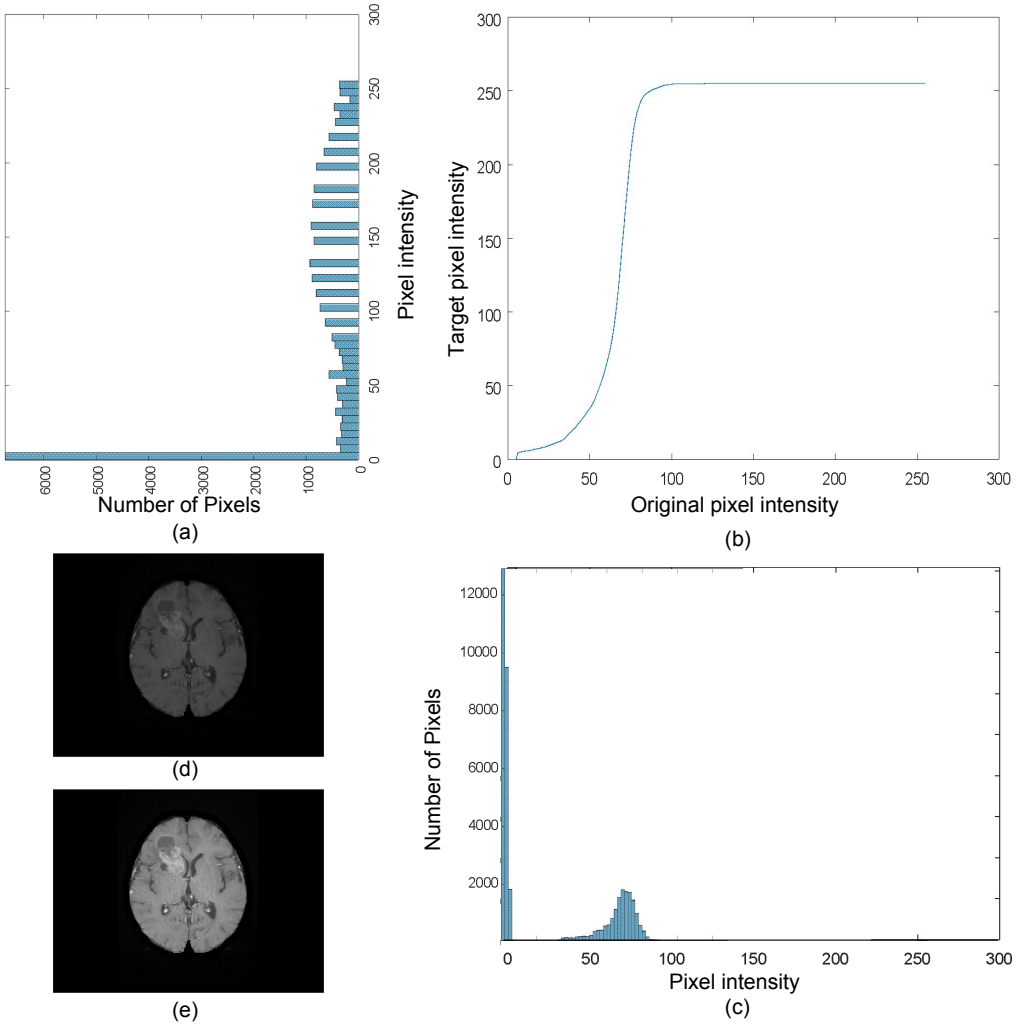


Figure 5.3: Histogram equalization applied **with** a foreground mask.

(a) is the resulting histogram (b) is the mapping function to map pixel intensities in the original image (c) histogram of the original image (d) original image (e) histogram equalization applied image. In order not to lose any information as in Figure 5.2, values less than 6 are ignored and the histogram obtained is distributed more equally except for the background values that have intensities less than the masking threshold.

5.2 Post-Processing Methods

In order to improve the performance, we have adapted some post processing methods. Since our problem is a multi-class problem, application of deterministic approaches like opening(erosion and dilation sequentially) is not practical. Thus, we have preferred two probabilistic methods. The first one is conditional random fields that can filter the output according to input pixel values and spatial locations.

Additionally, we have noticed that our algorithm may fail to distinguish between other anomalies and tumor tissues (Figure 5.4) due to similar characteristics. Thus, by assuming that the whole tumor (combination of tumor and tumor related tissues) is a single structure and only one tumor can exist in a brain, we have applied a maximum 3D connected region selection to filter out smaller structures. It has shown a significant improvement in the performance.

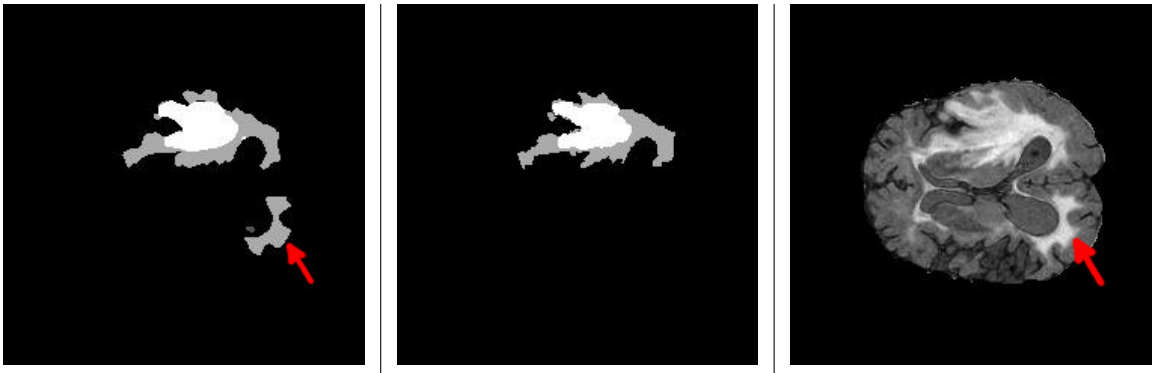
5.2.1 Conditional Random Fields

Conditional random fields [89] are actually a type of Markov random fields. Thus, prior to details of CRF used in this work, MRF is explained below.

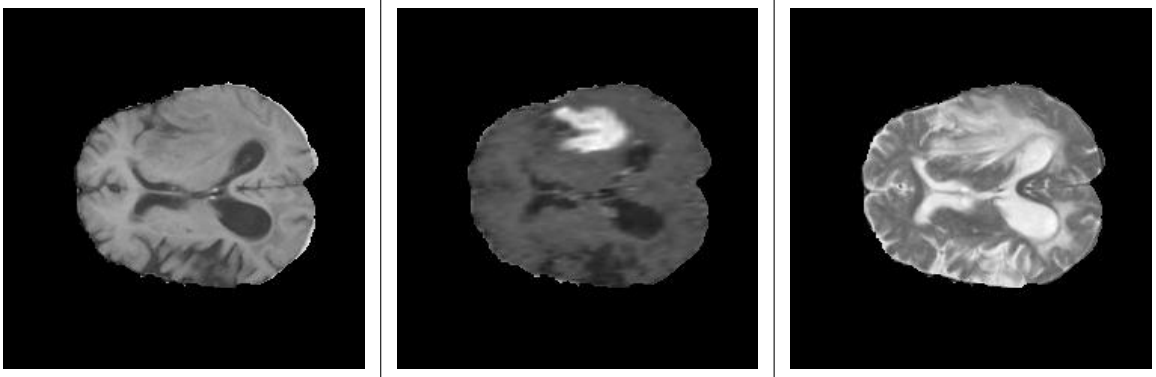
Markov Random Fields In Markov random fields (MRF)[90], the label (or the class) of a pixel is found, trying to maximize the probability $P(Y|X)$ where Y is the label output of whole image and X is the input image. In more mathematical representation, $argmax_y(P(Y|X))$ is found for each pixel x. In order for the probabilistic model to be a MRF, it needs to satisfy the following conditions for each pixel x in the image X and each label y in the label output Y:

1. It is always possible for a pixel to belong to any of the classes. $P(y = l) > 0$ for all $l \epsilon L$ where L is the available labels.

2. The probability of belonging to a class depends only on the labels of the neighbor pixels. $P(y_r|Y_{a-r}) = P(y_r|Y_{Nr})$ where y_r is a label, Y_{a-r} is all labels in Y except y_r and Y_{Nr} is the neighbor pixels of y_r.



(a) An example result (b) Ground truth (c) Flair



(d) T1 weighted (e) Contrast enhanced T1 (f) T2 weighted

Figure 5.4: An example result to show need for post-processing.

(a) shows the weaknesses of the method when compared with the ground truth (b). Note that edges need improvement. Also, the anomaly that is shown by the red arrow on the flair modality (c) is labeled as edema in (a) while on the ground truth(b) it is labeled as background.

3. The probabilities does not change according to the spatial position of pixel. $P(y_r|Y_{Nr})$ is same everywhere in the image.

Additionally, the probability has a Gibbs distribution:

$$P(y) = \frac{1}{Z}\exp(-U(y)) \tag{5.3}$$

where $Z = \sum_i \exp(-U(y))$.

In CRF, the calculation of clique potentials of labels vary. The CRF model used in this work is proposed in [91]. It makes use of the soft output of the network -the

probabilities of pixel labels obtained from softmax layer- and the raw pixel values of the input image to find a probability map. Then by finding the MAP, it reaches to final output labels. The Gibbs energy used in the proposed approach is

$$E(y) = \sum_i \Psi_u(y_i) + \sum_{i<j} \Psi_p(y_i, y_j) \tag{5.4}$$

where $\Psi_u(y_i)$ is the unary potential computed by the CNN classifier and $\Psi_p(y_i, y_j)$ is the pairwise potential as given in Eqn. 5.5 below.

$$\Psi_p(y_i, y_j) = \mu(y_i, y_j) \left(w^{(1)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \right) + w^{(2)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2} \right) \right) \tag{5.5}$$

The first term in the parenthesis compares the pixel values and pixel positions in order to provide that pixels closer to each other have the same class. The second term, on the other hand, is used to avoid small areas that are distant to other positive labeled areas -tumor regions in our case-.

Note that the code is taken from [91].

5.3 Largest Volume Filtering

During our experiments, we have observed that some networks fail in distinguishing between other anomalies and tumor related tissues. Thus, in order to filter out regions that are not connected to tumor, we have applied a 3D connected component analysis. Assuming that the tumor takes the most foreground space, only the larges 3D foreground volume is said to be the tumor. The main disadvantage is it can only be used to filter background and foreground. In other words, it does not provide any improvement on labeling the foreground data into its sub-classes. On the other hand, for brain tumor segmentation problem, it improved our whole tumor dice scores significantly.

CHAPTER 6

IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

This chapter will explain experimental results obtained from different networks and modifications explained in Chapter 4. The implementation details are also given.

6.1 Implementation Details

Framework In computer programming, a framework is a structure similar to a library, where the widely used functions for a specific task is pre-defined. The main difference between a library and a framework is that frameworks define the general flow of the code while libraries do not.

There are many deep learning frameworks like Caffe, theano, PyTorch and tensorflow to name a few. In this work, we have preferred tensorflow, which is based on python programming language, mostly because tensorboard -a visualization tool that automatically generates graphs of scalars from event files-. Moreover, it is rapidly improved since it is open source and supported by Google. I supports GPU processing and has a good documentation as well.

The methods are built using Python and Tensorflow 1.6 with GPU support (CUDA 9.0). Depending on the network size, the training is carried out on a laptop with an Nvidia GTX1060 (6GB) GPU or on a desktop computer with an Nvidia P5000 (16 GB). During training, only brain slices that has a tumor region greater than a threshold are chosen. By doing so, the class imbalance problem between tumor and non-tumor regions has been avoided.

6.2 Data Used In The Work

BraTS2017 Dataset BraTS dataset [15] [16] was used for both training and evaluation. It consists of 209 high grade glioma patients and 75 low grade glioma patients. Each patient has 4 different modalities of MRIs, namely, T1, T2, T2Flair and T1c (T1 with contrast agent). Each of these modalities are 3D matrices of size $240 \times 240 \times 155$. Regions such as skull and eye are stripped, leaving only the brain tissue. Thus, no pre-processing is needed for extracting the brain tissue.

Labels in BraTS2017 There are 4 classes in the BraTS 2017 dataset. One of them is background, while others are GD-enhancing tumor (ET), peritumoral edema (ED) and necrotic and non-enhancing tumor (NCR/NET). In the validation results, white, gray and dark gray regions correspond to ET, ED and NCR/NET respectively. In the comparison part, instead of using these classes, combinations of these are used. Whole tumor (WT) consists of ET, ED and NCR/NET while tumor core (TC) is the combination of ET and NCR/NET.

Peritumoral edema (ED) region corresponds to the accumulation of excess fluid around the tumor. It generally grows rapidly as a reaction to malignant tumor. As edema becomes larger, it creates pressure on the other parts of brain and can damage these parts. Necrotic/non-enhancing tumor (NCR/NET) region corresponds to dead tissue. It is one of the significant signs that the lesion is malignant. For example, if there is necrotic tissue, then the expert can say that the tumor is malignant. Otherwise, further tests are needed to decide whether it is benign or malignant. Enhancing tumor is the tumor tissue that is still alive. It continues spreading to other parts of brain and contains malignant cells.

6.3 Experimental Results

6.3.1 Metrics Used for Rating performance

Metrics used for rating the network performance are explained in this section.

6.3.1.1 Dice Coefficient

Dice Coefficient (Sørensen-Dice Score) is a similarity metric. First, the positive elements in the result that are equal to the ground truth are found. Then this result is divided by the total number of positives in both.

$$DICE = \frac{2|A \cap B|}{|A| + |B|} = \frac{2(TruePositives)}{(Positives) + (GroundTruthPositives)} \tag{6.1}$$

6.3.1.2 Sensitivity and Specificity

Sensitivity is a metric to measure the ratio of True Positives (TP) to Total Ground Truth Positives $(TP + FN)$ (Eqn. 6.2) while Specificity is a metric to measure the ratio of True Negatives (TN) to Total Ground Truth Negatives $(TN+FP)$ (Eqn. 6.3). In other words, as the sensitivity increases, the possibility of a negative labeled pixel's being a true negative increases, while as the specificity increases, the possibility of a positive labeled pixel's being a true positive increases.

$$Sensitivity = \frac{TP}{TP + FN} \tag{6.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6.3}$$

6.3.2 Results

The training results are provided for different metrics and all 5 classes, namely, necrotic and non-enhancing tumor(NCR/NET), edema(ED), whole tumor (ET, ED and NCR/NET), enhanced tumor and tumor core (ET and NCR/NET). The results are provided in terms of iteration steps. Each step is trained with a batch of size 8 and there are 12500 iteration steps in total.

6.3.2.1 Structure Based Comparison

Table 6.1: Validation results obtained using different architectures, different connections and different filter sizes

	High Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Single 3x3 U-Net	0.434	0.740	0.774	0.865	0.808	0.352	0.672	0.862	0.820	0.824	0.999	0.999	0.999	0.999	0.999
U-Net With Inception Modules	0.430	0.728	0.772	0.852	0.731	0.399	0.766	0.695	0.855	0.667	0.999	0.997	0.999	0.998	0.999
Single 3x3 U-Net and Dilation Module	0.481	0.753	0.807	0.866	0.782	0.450	0.762	0.805	0.865	0.764	0.999	0.998	0.999	0.998	0.999
Single 3x3 U-Net and Dilation Module with Residual	0.494	0.748	0.812	0.863	0.777	0.575	0.730	0.759	0.852	0.778	0.999	0.998	0.999	0.998	0.999
Multi-scale U-Net	0.572	0.766	0.824	**0.878**	0.829	0.565	0.752	**0.840**	0.870	0.837	0.999	0.998	0.999	0.998	0.999
Multi-scale U-Net with Dilation Module	0.666	0.764	0.825	0.869	**0.867**	0.727	0.714	0.798	0.834	**0.872**	0.999	0.999	0.999	0.999	0.999
Multi-scale U-Net and Dilation Module With Residual Connection	0.510	0.749	**0.827**	0.874	0.801	0.505	0.709	0.809	0.837	0.790	0.999	0.998	0.999	0.999	0.999
FCN-8s	0.480	0.701	0.749	0.850	0.783	0.527	0.655	0.735	0.821	0.804	0.999	0.998	0.999	0.998	0.999
FCN-8s with Inception Module	0.269	0.534	0.163	0.816	0.505	0.382	0.498	0.110	0.753	0.459	0.998	0.997	0.999	0.999	0.998
FCN-8s with Dilation Module	0.496	0.736	0.781	0.864	0.795	0.499	0.743	0.779	**0.871**	0.792	0.999	0.998	0.999	0.998	0.999
Multi-scale FCN-8s with Dilation Module	0.576	0.750	0.785	0.861	0.825	0.547	0.716	0.767	0.828	0.799	0.999	0.998	0.999	0.999	0.999
FCN-8s with Dilation and Inception Module	0.354	0.701	0.785	0.837	0.733	0.326	0.716	0.795	0.842	0.718	0.999	0.997	0.999	0.998	0.999
FCN-8s with Dilation and Residual	0.479	0.678	0.797	0.856	0.780	0.535	0.572	0.844	0.794	0.845	0.999	0.999	0.999	0.999	0.998
Multi-scale U-Net and Dilation Module With Residual Connection*	0.638	0.770	0.839	0.875	0.860	0.675	0.735	0.840	0.855	0.876	0.999	0.998	0.999	0.999	0.999
	Low Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Single 3x3 U-Net	0.581	0.666	0.536	0.844	0.624	0.460	0.631	**0.583**	0.759	0.524	0.999	0.997	0.999	0.999	0.999
U-Net With Inception Modules	0.481	0.682	0.536	0.862	0.520	0.366	0.789	0.441	0.852	0.401	0.999	0.994	0.999	0.998	0.999
Single 3x3 U-Net and Dilation Module	0.588	0.686	0.527	0.862	0.603	0.487	0.715	0.497	0.824	0.509	0.999	0.996	0.999	0.998	0.998
Single 3x3 U-Net and Dilation Module with Residual	0.641	0.690	**0.633**	**0.882**	0.661	0.620	0.684	0.498	0.854	0.622	0.998	0.997	0.999	0.998	0.998
Multi-scale U-Net	0.656	0.694	0.541	0.866	0.665	0.632	0.655	0.536	0.827	0.639	0.998	0.997	0.999	0.998	0.998
Multi-scale U-Net with Dilation Module	0.642	0.696	0.521	0.850	0.647	0.573	0.672	0.386	0.786	0.564	0.999	0.997	0.999	0.999	0.998
Multi-scale U-Net and Dilation Module With Residual Connection	0.672	0.698	0.602	0.876	**0.682**	0.641	0.658	0.523	0.826	**0.644**	0.998	0.997	0.999	0.999	0.998
FCN-8s	0.646	0.684	0.486	0.870	0.664	0.592	0.672	0.386	0.823	0.595	0.998	0.997	0.999	0.999	0.998
FCN-8s with Inception Module	0.459	0.616	0.040	0.818	0.536	0.390	0.590	0.041	0.752	0.466	0.998	0.997	0.999	0.998	0.998
FCN-8s with Dilation Module	0.532	0.689	0.532	0.876	0.574	0.422	0.777	0.468	**0.866**	0.464	0.999	0.995	0.999	0.998	0.999
Multi-scale FCN-8s with Dilation Module	0.562	0.696	0.467	0.839	0.566	0.417	0.730	0.327	0.767	0.417	0.999	0.996	0.999	0.999	0.999
FCN-8s with Dilation and Inception Module	0.404	0.647	0.415	0.795	0.478	0.293	0.722	0.487	0.779	0.396	0.999	0.995	0.999	0.996	0.998
FCN-8s with Dilation and Residual	0.654	0.591	0.430	0.841	0.648	0.619	0.497	0.539	0.761	0.643	0.998	0.998	0.999	0.999	0.997
Multi-scale U-Net and Dilation Module With Residual Connection*	0.607	0.709	0.446	0.835	0.598	0.470	0.742	0.306	0.770	0.455	0.999	0.996	0.999	0.999	0.999
	Average of HGG and LGG Results														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Single 3x3 U-Net	0.508	0.703	0.655	0.855	0.716	0.406	0.652	**0.723**	0.790	0.674	0.999	0.998	0.999	0.999	0.999
U-Net With Inception Modules	0.455	0.705	0.654	0.857	0.626	0.382	0.778	0.568	0.854	0.534	0.999	0.996	0.999	0.998	0.999
Single 3x3 U-Net and Dilation Module	0.534	0.720	0.667	0.864	0.693	0.468	0.738	0.651	0.845	0.637	0.999	0.997	0.999	0.998	0.999
Single 3x3 U-Net and Dilation Module with Residual	0.568	0.719	**0.723**	0.873	0.719	0.597	0.707	0.629	0.853	0.700	0.998	0.997	0.999	0.998	0.999
Multi-scale U-Net	0.614	0.730	0.683	0.872	0.747	0.598	0.703	0.688	0.849	**0.738**	0.999	0.998	0.999	0.998	0.999
Multi-scale U-Net with Dilation Module	0.654	0.730	0.673	0.860	**0.757**	0.650	0.693	0.592	0.810	0.718	0.999	0.998	0.999	0.999	0.999
Multi-scale U-Net and Dilation Module With Residual Connection	0.591	0.723	0.715	**0.875**	0.742	0.573	0.683	0.666	0.832	0.717	0.999	0.998	0.999	0.999	0.999
FCN-8s	0.563	0.693	0.618	0.860	0.724	0.559	0.663	0.561	0.822	0.700	0.999	0.997	0.999	0.999	0.999
FCN-8s with Inception Module	0.364	0.575	0.102	0.817	0.521	0.386	0.544	0.076	0.753	0.463	0.998	0.997	0.999	0.999	0.998
FCN-8s with Dilation Module	0.514	0.713	0.657	0.870	0.685	0.460	0.760	0.624	**0.869**	0.628	0.999	0.996	0.999	0.998	0.999
Multi-scale FCN-8s with Dilation Module	0.569	0.723	0.634	0.867	0.730	0.482	0.723	0.583	0.847	0.694	0.999	0.997	0.999	0.999	0.999
FCN-8s with Dilation and Inception Module	0.379	0.674	0.600	0.828	0.652	0.309	0.719	0.627	0.804	0.598	0.999	0.996	0.999	0.998	0.999
FCN-8s with Dilation and Residual	0.567	0.635	0.608	0.839	0.691	0.577	0.535	0.667	0.802	0.681	0.999	0.999	0.999	0.999	0.998
Multi-scale U-Net and Dilation Module With Residual Connection*	0.622	0.739	0.622	0.846	0.689	0.572	0.738	0.575	0.782	0.650	0.999	0.997	0.999	0.999	0.999

*Instead of using 1x1 convolutions, multi-scale network results are directly averaged.

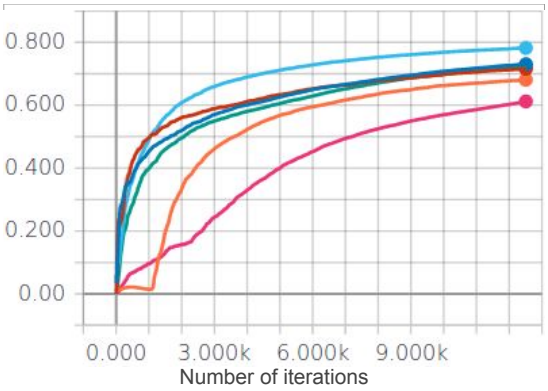
Table 6.2: General information on architectures

Name	Brief Information on Architecture	Approximate Number of Trainable Parameters
Single 3x3 U-Net	U-Net that has 2 convolution layers(with filter size 3x3) at each level	525,000
U-Net With Inception Modules	U-Net architecture with inception modules instead of convolution layers	3,800,000
Single 3x3 U-Net and Dilation Module	U-Net that has 2 convolution layers(with filter size 3x3) at each level combined with a dilation module	530,000
Single 3x3 U-Net and Dilation Module with Residual	U-Net that has 3 convolution layers at each level, connected with residual connections, combined with a dilation module	2,880,000
Multi-scale U-Net	Combination of 2 separate U-Nets that have 3x3 filter size and 5x5 filter size	1,900,000
Multi-scale U-Net with Dilation Module	Combination of 2 separate U-Nets that have 3x3 filter size and 5x5 filter size and a dilation module	1,905,000
Multi-scale U-Net and Dilation Module With Residual Connection	Combination of 2 separate U-Nets, each network has 3 convolution layers at each level, connected with residual connections, combined with a dilation module	800,000
FCN-8s	Original FCN-8s architecture with filter size 3x3	890,000
FCN-8s with Inception Module	Original FCN-8s architecture with convolution layers replaced with inception modules	6,400,000
FCN-8s with Dilation Module	Original FCN-8s architecture combined with a dilation module	895,000
Multi-scale FCN-8s with Dilation Module	Combination of 2 FCN-8s that have 3x3 filter size and 5x5 filter size and a dilation module	3,205,000
FCN-8s with Dilation and Inception Module	Original FCN-8s architecture with convolution layers replaced with inception modules, combined with a dilation module	6,405,000
FCN-8s with Dilation and Residual	FCN-8s architecture with residual connections and a dilation module	895,000

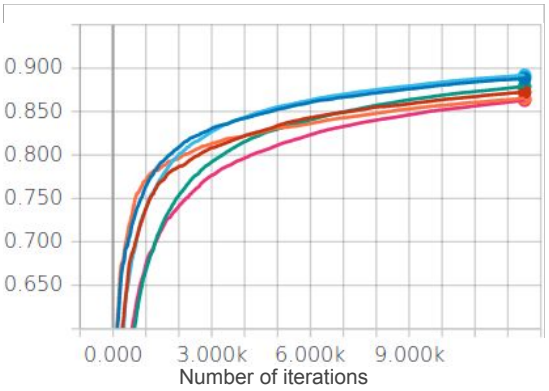
The results obtained during training of different architectures are given in Figures 6.1, 6.2, 6.3. For each network, a brief explanation and approximate number of trainable parameters are provided in Table 6.2. When Dice scores are considered, best results are obtained by the combination of two U-Nets that include a dilation module (Figure 6.1). We can also say that FCN-8s have good validation results despite its direct upsampling approach (Table 6.1).

It can be seen in Figure 6.2 that U-Net itself lacks the sensitivity. Also, the most sensitive method seems to be double U-Nets with a dilation module. Additionally, it produces confident results (Figure 6.3).

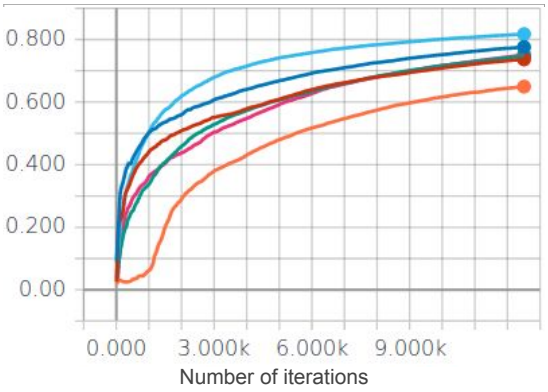
Residual connections do not seem to improve results on the training results(Figure 6.1), although they improve the validation results slightly (Table 6.1). There are two possible reasons of this. The first one is that the network is not that much deep. The second one is that the U-Net architecture already has some skip connections. Thus, it does not suffer from vanishing gradients problem. By the addition of extra residual connections, the propagation is improved without making a significant performance improvement.



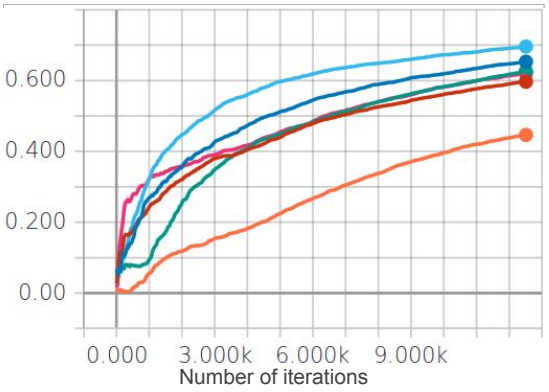
(a) Enhanced Tumor Dice Score



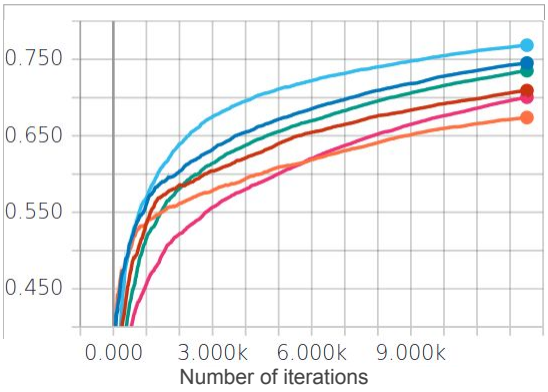
(b) Whole Tumor Dice Score



(c) Tumor Core Dice Score



(d) Necrotic/Non-enhancing Tumor Dice Score



(e) Edema Dice Score

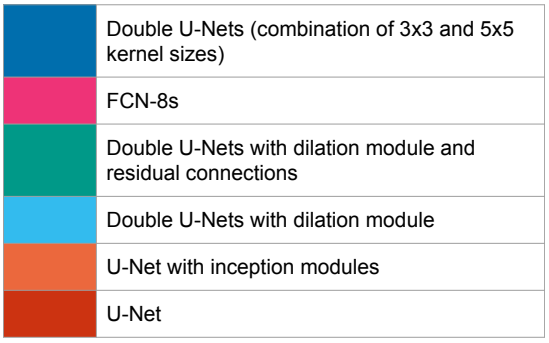
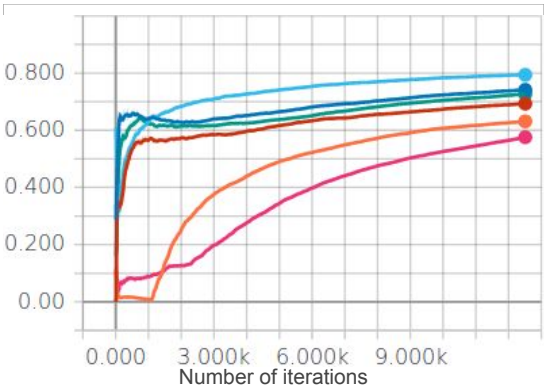
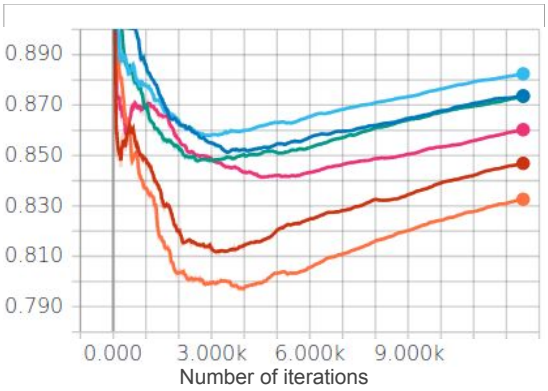


Figure 6.1: Dice scores obtained during training different architectures.

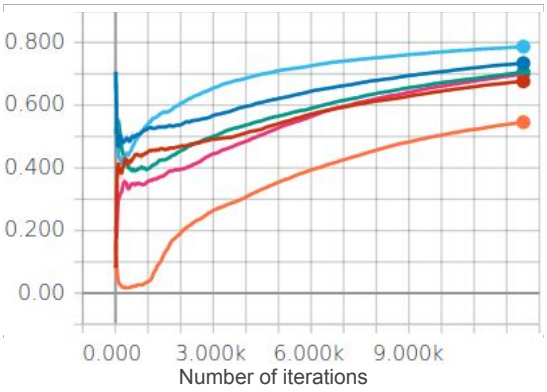
Inception modules did not improve the FCN-8s results most probably because the network was too wide. When applied on our single U-Net architecture, they have slightly improved the results. Therefore, instead of taking advantage of different filter



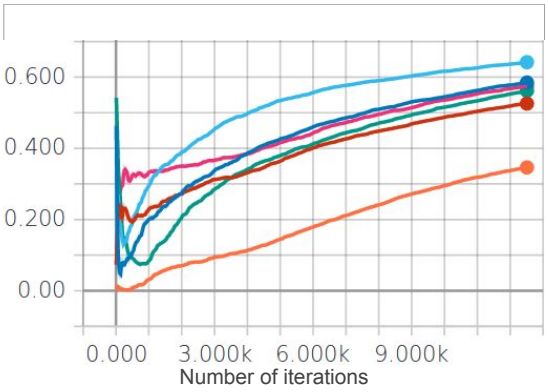
(a) Enhanced Tumor Sensitivity Result



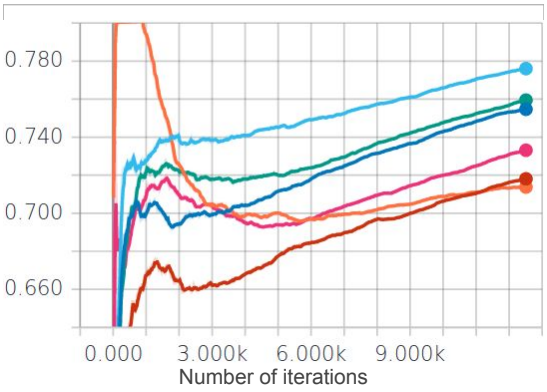
(b) Whole Tumor Sensitivity Result



(c) Tumor Core Sensitivity Result



(d) Necrotic/Non-enhancing Tumor Sensitivity Result



(e) Edema Sensitivity Result

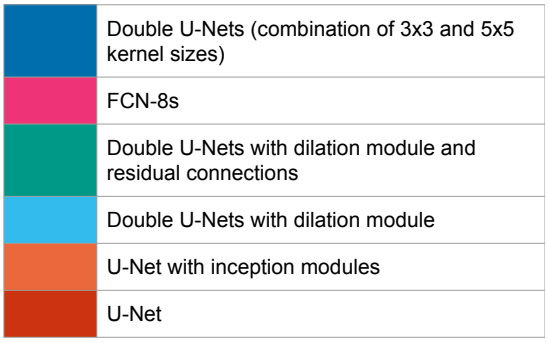
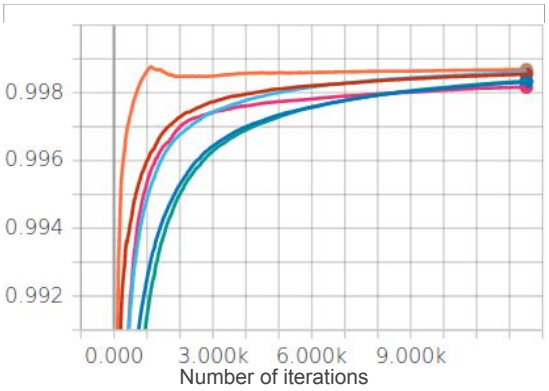


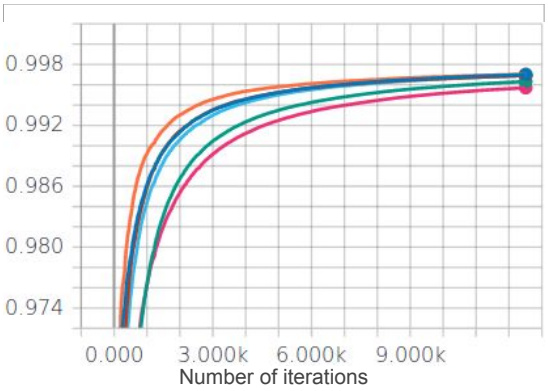
Figure 6.2: Sensitivity results obtained during training different architectures.

sizes in smaller modules, we have built a multi-scale U-Net(Table 6.2).

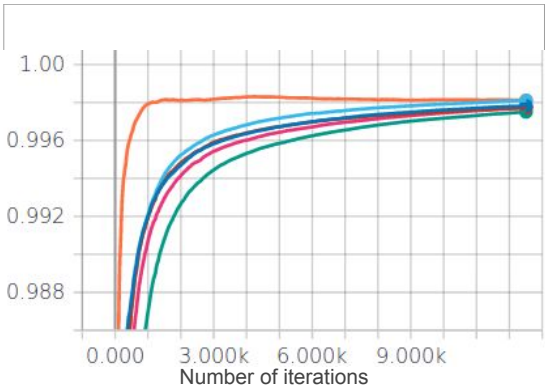
We see the effect of dilation module and residual connections in Figures 6.4 and 6.5. FCN-8s itself (6.4(c) ,6.5(c)), fails to produce a result with local details. Its



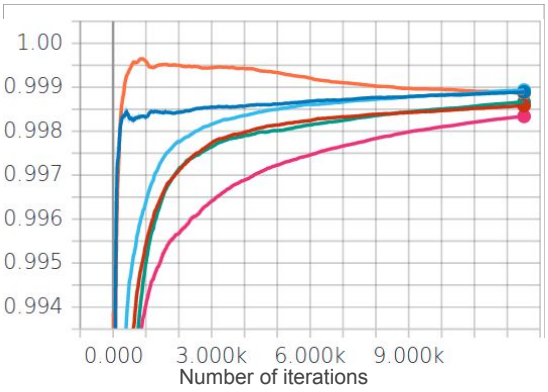
(a) Enhanced Tumor Specificity Result



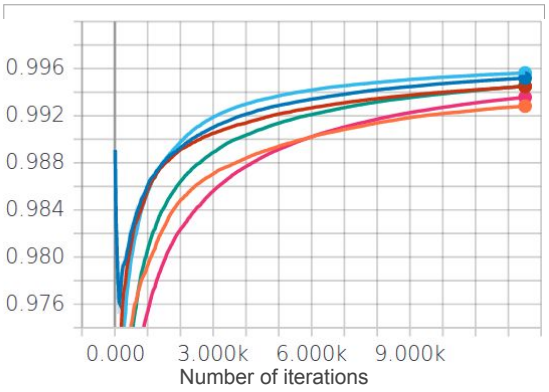
(b) Whole Tumor Specificity Result



(c) Tumor Core Specificity Result



(d) Necrotic/Non-enhancing Tumor Specificity Result



(e) Edema Specificity Result

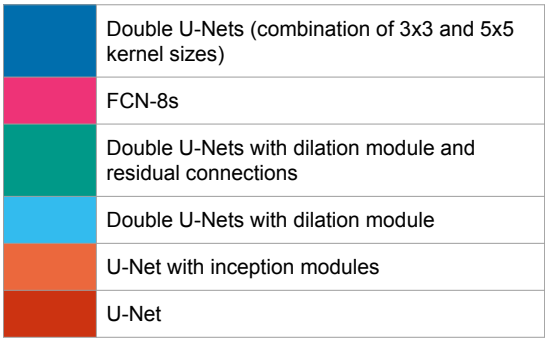
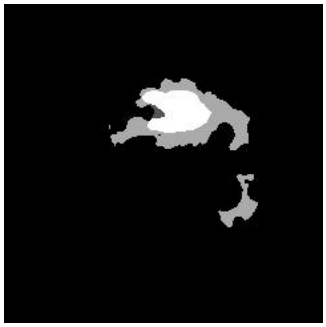


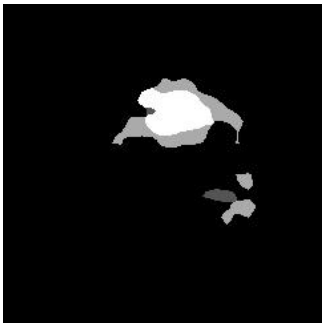
Figure 6.3: Specificity results obtained during training different architectures.

results have round edges. However, dilation module provides the local information and residual connections help to avoid overfitting(6.4(d) ,6.4(d)). We can also see how combining two different networks can improve the results (Figure 6.4(a) and

(a) Double U-Nets with dilation module and residual connections

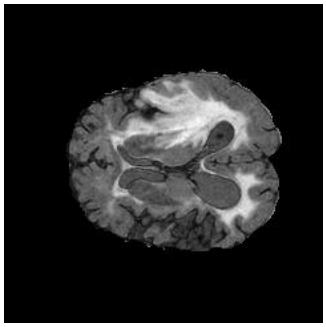


(b) Single U-Net with dilation module and residual connections

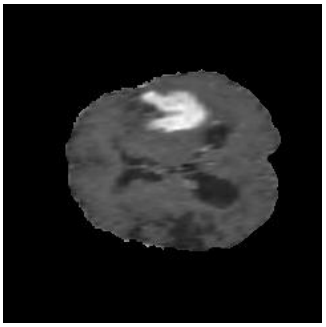


(c) FCN-8s

(d) FCN-8s with dilation module and residual connections

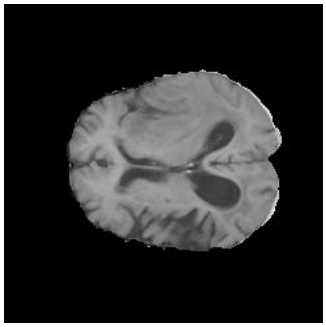


(e) Flair MRI

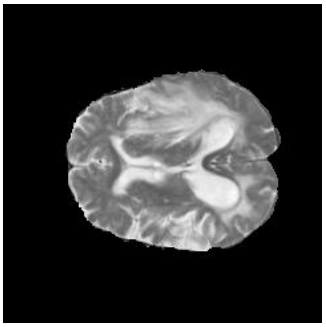


(f) Contrast enhanced MRI

(g) Ground truth



(h) T1 weighted MRI

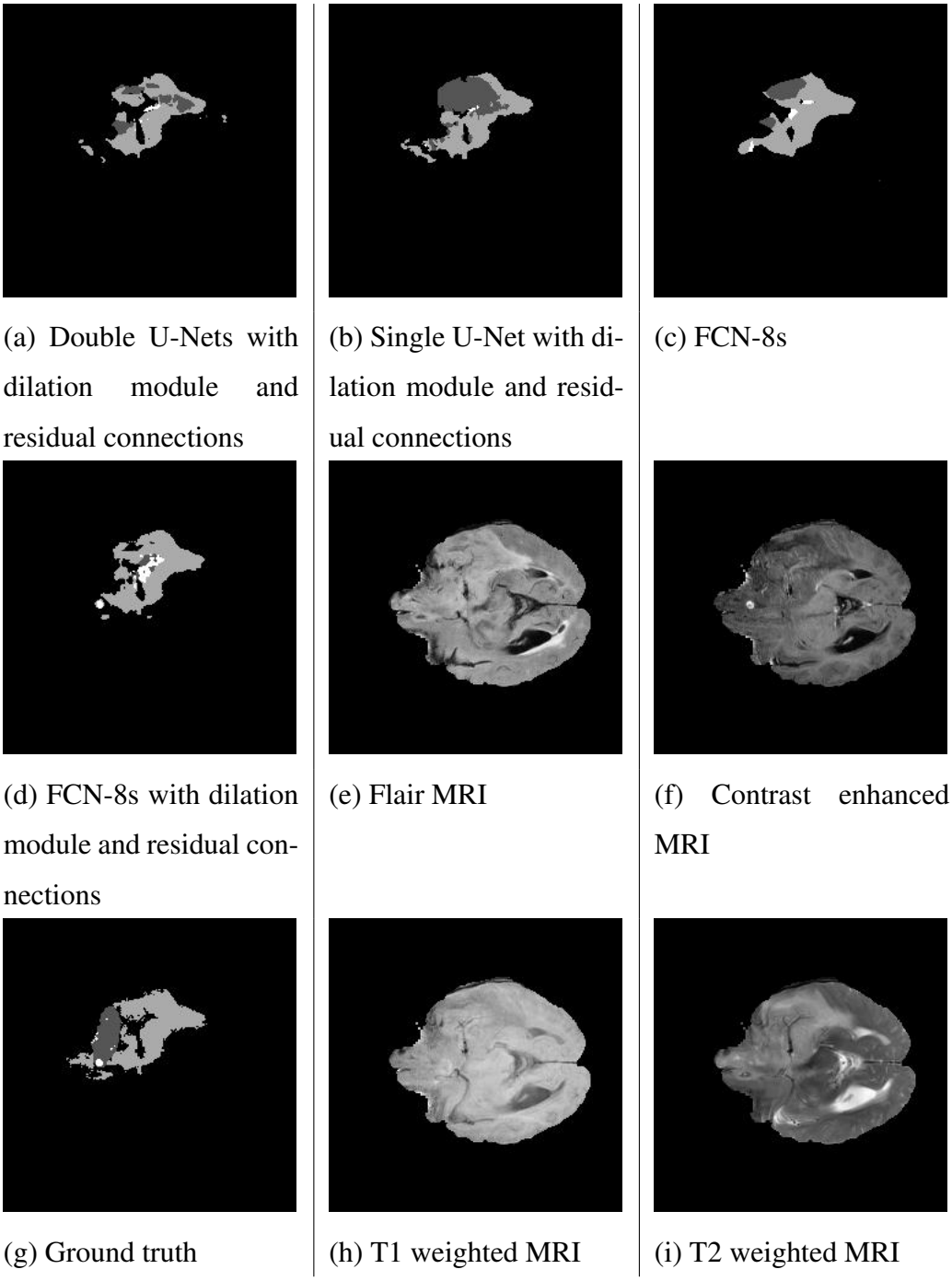


(i) T2 weighted MRI

Figure 6.4: Examples of validation results of HGG patients obtained by top performing structures.

(b)).

The improvement in local details with combination of two U-Nets, compared to a single one is also shown in Figures 6.4,6.4 (a) and (b) respectively.



(a) Double U-Nets with dilation module and residual connections

(b) Single U-Net with dilation module and residual connections

(c) FCN-8s

(d) FCN-8s with dilation module and residual connections

(e) Flair MRI

(f) Contrast enhanced MRI

(g) Ground truth

(h) T1 weighted MRI

(i) T2 weighted MRI

Figure 6.5: Examples of validation results of LGG patients obtained by top performing structures.

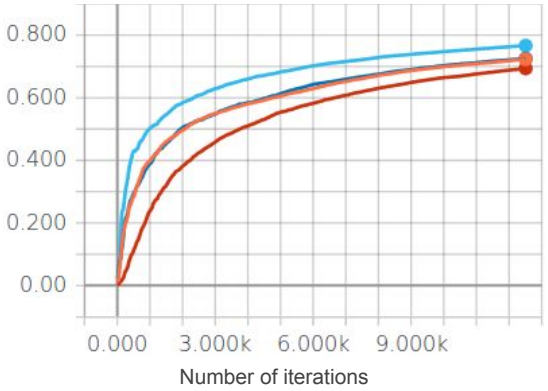
Note that multi-scale U-Net with a dilation module and residual blocks is selected as the reference architecture. It will be used for the rest of the comparisons in the following sections.

6.3.2.2 Comparison of Different Activation Functions

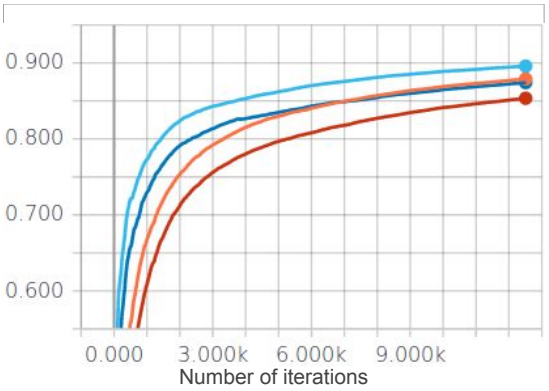
Table 6.3: Validation results obtained using different activation functions

	High Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
ReLU	0.510	0.749	0.827	0.874	0.801	0.505	0.709	0.809	0.837	0.790	0.999	0.998	0.999	0.999	0.999
LReLU	0.473	0.700	0.754	0.825	0.737	0.508	0.791	0.731	0.894	0.739	0.999	0.996	0.999	0.996	0.999
tanh	0.394	0.672	0.713	0.783	0.672	0.655	0.718	**0.813**	**0.902**	**0.863**	0.998	0.996	0.998	0.995	0.997
PReLU	0.525	0.738	**0.840**	**0.876**	**0.844**	0.666	0.775	0.802	0.832	0.848	0.999	0.997	0.999	0.999	0.999
	Low Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
ReLU	0.672	0.698	0.602	**0.876**	0.682	0.641	0.658	0.523	0.826	0.644	0.998	0.997	0.999	0.999	0.998
LReLU	0.627	0.678	0.519	0.814	0.631	0.509	0.774	0.475	0.823	0.525	0.999	0.994	0.999	0.996	0.999
tanh	0.612	0.625	0.325	0.796	0.595	0.578	0.659	0.513	0.832	0.619	0.998	0.995	0.998	0.995	0.996
PReLU	0.652	0.676	**0.641**	0.864	**0.694**	0.600	0.706	0.545	0.799	**0.658**	0.998	0.996	0.999	0.999	0.998
	Average of HGG and LGG Results														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
ReLU	0.591	0.723	0.715	**0.875**	0.742	0.573	0.683	0.666	0.832	0.717	0.999	0.998	0.999	0.999	0.999
LReLU	0.550	0.689	0.637	0.820	0.684	0.508	0.783	0.603	0.859	0.632	0.999	0.995	0.999	0.996	0.999
tanh	0.503	0.649	0.519	0.790	0.634	0.617	0.689	0.663	**0.867**	0.741	0.998	0.996	0.998	0.995	0.997
PReLU	0.588	0.707	**0.741**	0.870	**0.769**	0.633	0.740	**0.674**	0.816	**0.753**	0.999	0.996	0.999	0.999	0.999

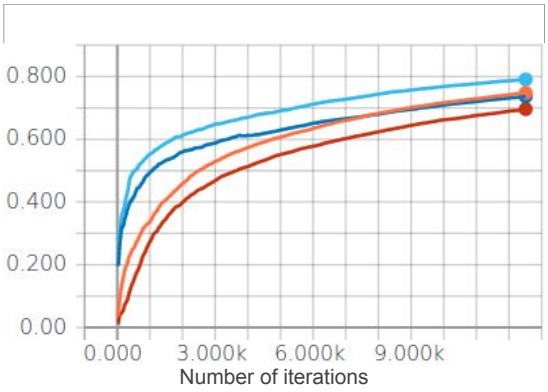
We have compared ReLU, LReLU, PReLU and tanh. According to Figure 6.6 which show dice scores obtained during training with different activation functions, PReLU is the top performing one. The validation results given in Table 6.3 on the other hand, shows that for the whole tumor, ReLU performs better. In some cases, ReLU can perform better because of its sparse results. Since tanh is a saturating function it causes vanishing gradients. Thus, its performance is the worst as expected. LReLU, having a constant leakage coefficient, lacks both the sparsity and trainability. Thus, its leakage coefficient needs fine-tuning for the specific problem. Finally, tanh shows the worst performance as expected due to its saturating characteristic which causes vanishing gradient problem.



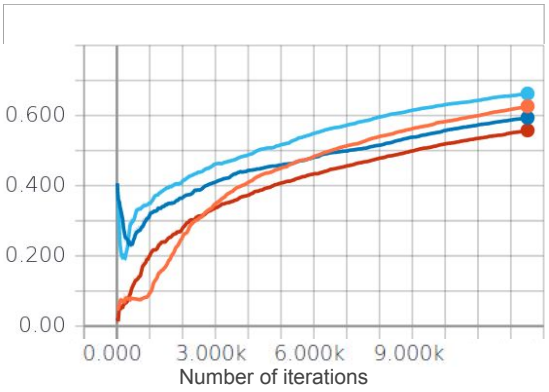
(a) Enhanced Tumor Dice Score



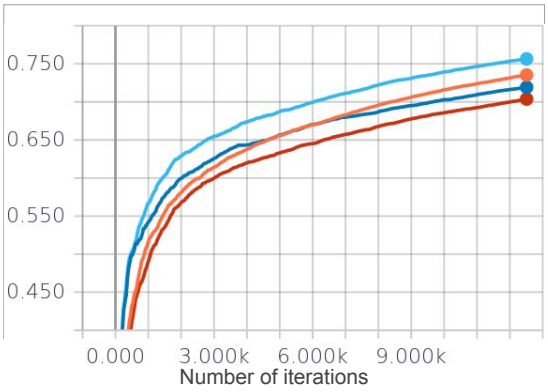
(b) Whole Tumor Dice Score



(c) Tumor Core Dice Score



(d) Necrotic/Non-enhancing Tumor Dice Score



(e) Edema Dice Score

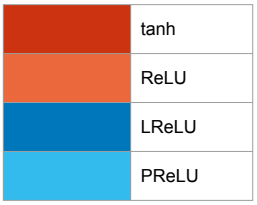
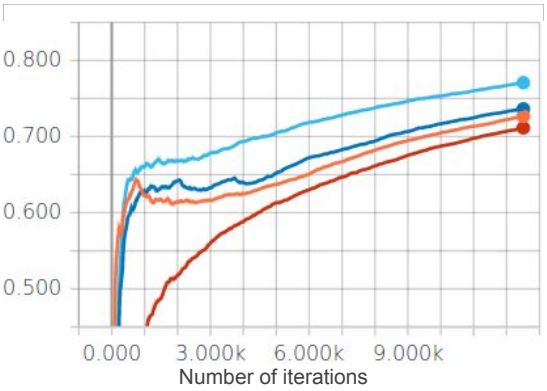
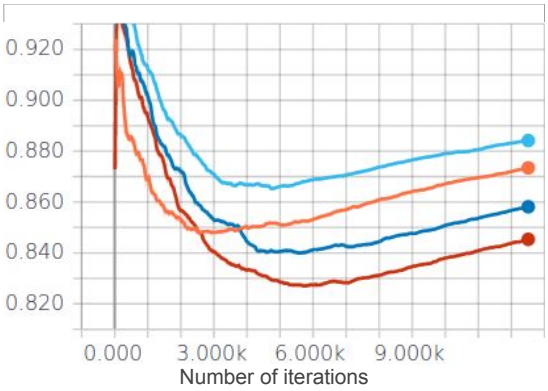


Figure 6.6: Dice scores obtained from different activation functions during training.

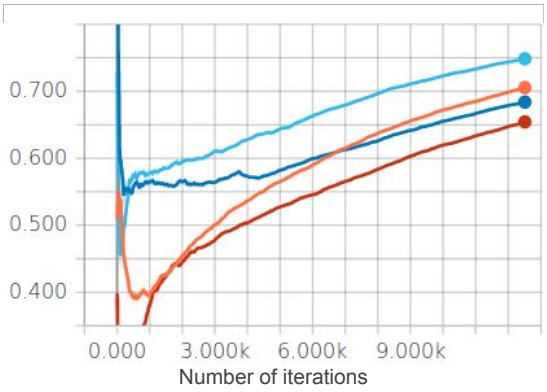
During training, as expected, tanh has the worst performance. Although LReLU seems to have a better performance than ReLU for the training 6.6, for the validation, ReLU performs better than LReLU. To be more specific, validation results of



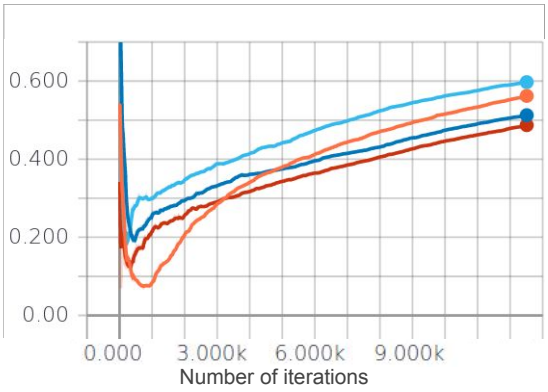
(a) Enhanced Tumor Sensitivity Result



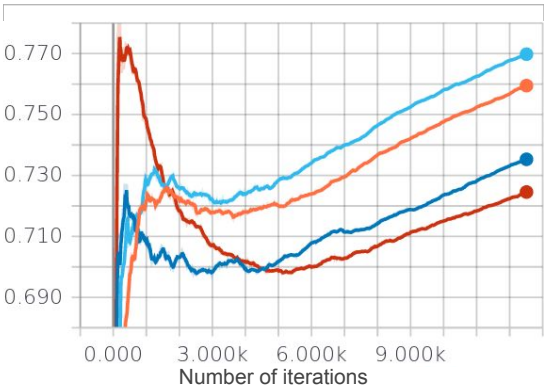
(b) Whole Tumor Sensitivity Result



(c) Tumor Core Sensitivity Result



(d) Necrotic/Non-enhancing Tumor Sensitivity Result



(e) Edema Sensitivity Result

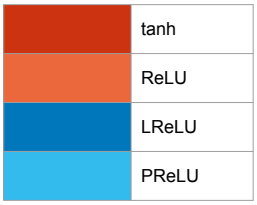
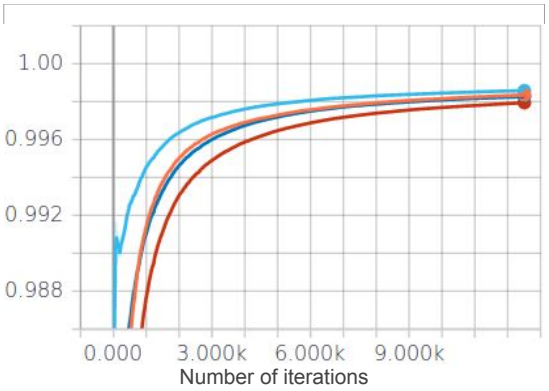
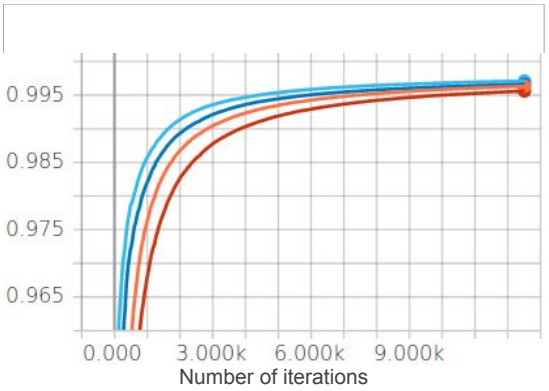


Figure 6.7: Sensitivity results obtained from different activation functions during training.

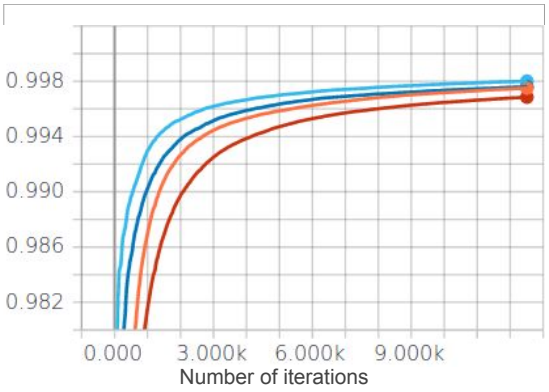
ReLU and PReLU are similar Table 6.3. When Figures 6.9 and 6.10 are considered, one can see that LReLU and tanh fails by labeling some background pixels as fore-



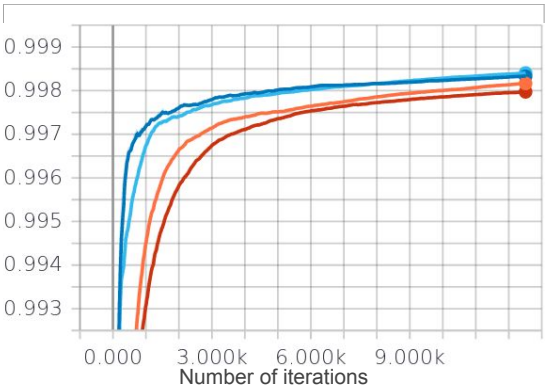
(a) Enhanced Tumor Specificity Result



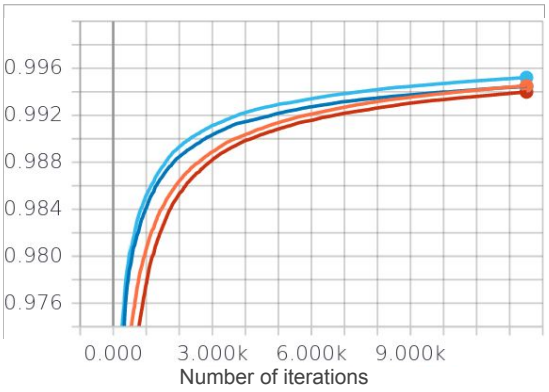
(b) Whole Tumor Specificity Result



(c) Tumor Core Specificity Result



(d) Necrotic/Non-enhancing Tumor Specificity Result



(e) Edema Specificity Result

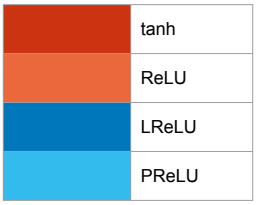
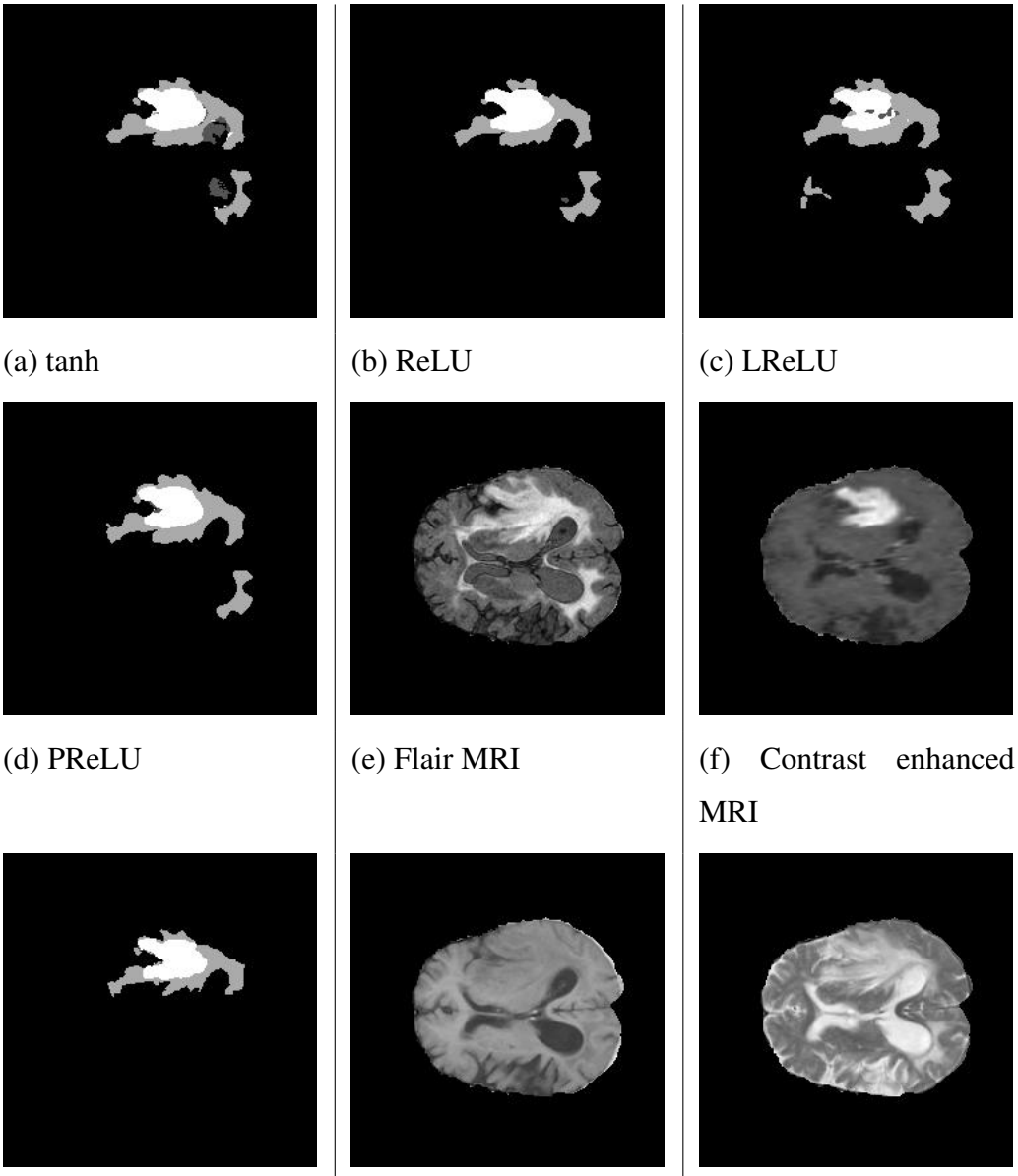


Figure 6.8: Specificity results obtained from different activation functions during training.

ground. Although there is not any significant difference between ReLU and PReLU for the HGG results(Figure 6.9), PReLU more correctly labels the sub-classes when

LGG results are considered (Figure 6.10).

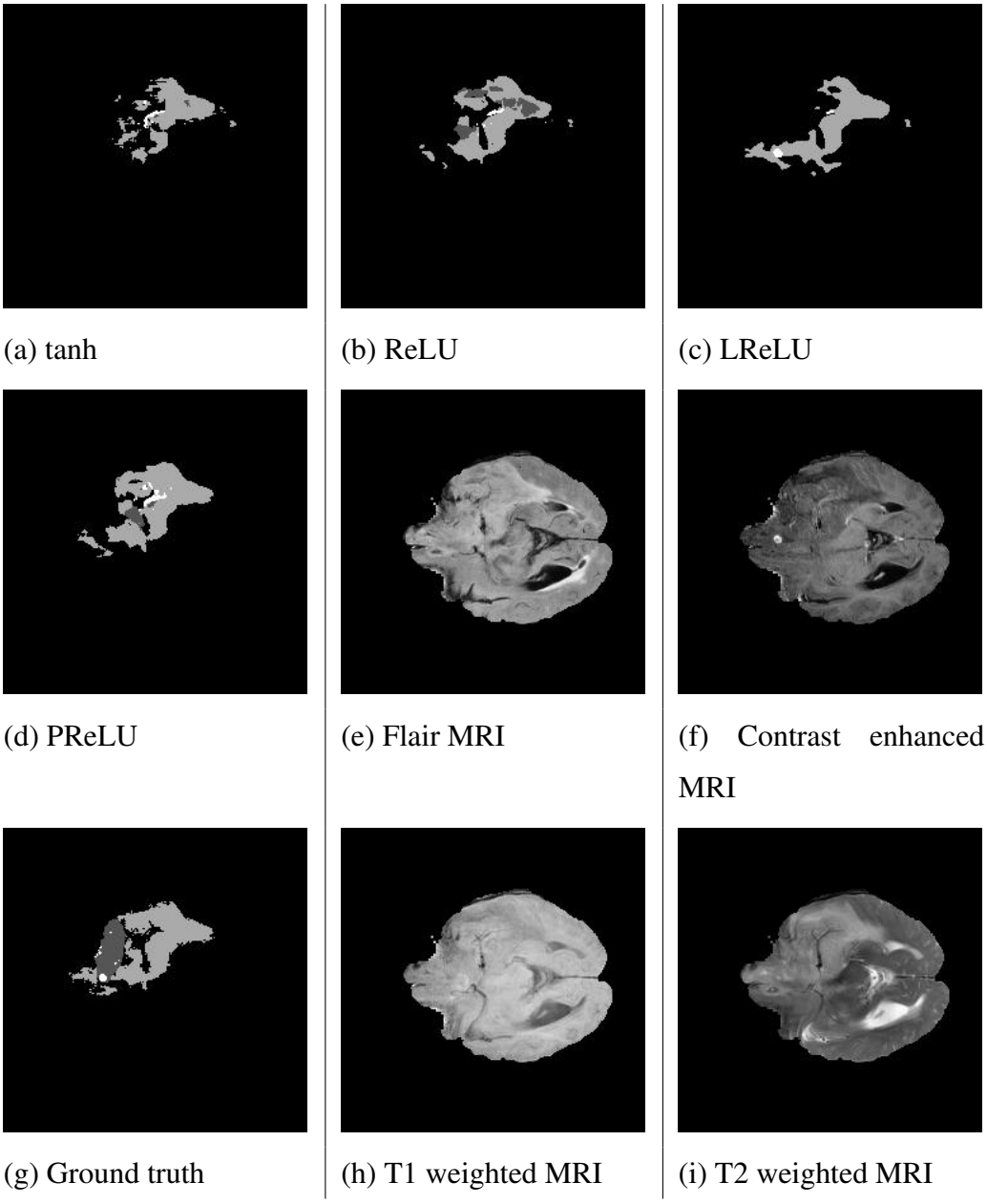


(a) tanh (b) ReLU (c) LReLU

(d) PReLU (e) Flair MRI (f) Contrast enhanced MRI

(g) Ground truth (h) T1 weighted MRI (i) T2 weighted MRI

Figure 6.9: Examples of validation results of HGG patients obtained using different upscale methods.



(a) tanh (b) ReLU (c) LReLU

(d) PReLU (e) Flair MRI (f) Contrast enhanced MRI

(g) Ground truth (h) T1 weighted MRI (i) T2 weighted MRI

Figure 6.10: Examples of validation results of LGG patients obtained using different upscale methods.

Table 6.4: Validation results obtained using different upsampling methods

	ED														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Bilinear Interpolation	0.510	0.749	0.827	**0.874**	0.801	0.505	0.709	0.809	0.837	0.790	0.999	0.998	0.999	0.999	0.999
Deconvolution	0.518	0.717	0.820	0.860	0.798	0.513	0.655	0.775	0.800	0.772	0.999	0.999	0.999	0.999	0.999
Subpixel	0.489	0.720	0.813	0.869	0.769	0.664	0.638	0.861	**0.853**	**0.888**	0.998	0.999	0.999	0.998	0.998
Trainable Bilinear Interpolation	0.642	0.753	**0.836**	0.870	**0.865**	0.627	0.670	**0.867**	0.814	0.878	0.999	0.999	0.999	0.999	0.999
	Low Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Bilinear Interpolation	0.672	0.698	0.602	**0.876**	**0.682**	0.641	0.658	0.523	0.826	0.644	0.998	0.997	0.999	0.999	0.998
Deconvolution	0.663	0.676	0.505	0.828	0.657	0.583	0.608	0.374	0.729	0.564	0.999	0.998	0.999	0.999	0.999
Subpixel	0.622	0.555	0.548	0.867	0.638	0.760	0.442	**0.643**	**0.847**	**0.776**	0.995	0.998	0.999	0.998	0.995
Trainable Bilinear Interpolation	0.556	0.663	**0.605**	0.834	0.565	0.448	0.644	0.538	0.780	0.456	0.999	0.997	0.999	0.998	0.999
	Average of HGG and LGG Results														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT		ET	WT	TC
Bilinear Interpolation	0.591	0.723	0.715	**0.875**	**0.742**	0.573	0.683	0.666	0.832	0.717	0.999	0.998	0.999	0.999	0.999
Deconvolution	0.591	0.697	0.663	0.844	0.728	0.548	0.632	0.575	0.765	0.668	0.999	0.998	0.999	0.999	0.999
Subpixel	0.555	0.637	0.681	0.868	0.704	0.712	0.540	**0.752**	**0.850**	**0.832**	0.997	0.999	0.999	0.998	0.997
Trainable Bilinear Interpolation	0.599	0.708	**0.721**	0.852	0.715	0.538	0.657	0.702	0.797	0.667	0.999	0.998	0.999	0.999	0.999

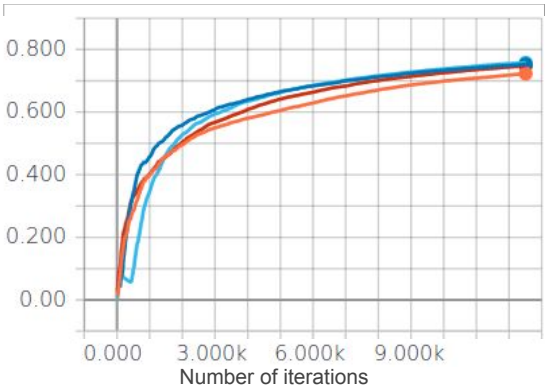
6.3.2.3 Comparison of Different Upsampling Methods

Although bilinear interpolation is not trainable and performs worse than other methods when training dataset is considered, it achieves best validation results when whole tumor dice score is considered Table 6.4. The table also shows that trainable bilinear interpolation also performs well in the validation experiments. It achieves to improve the result of sub-classes but not the whole tumor. A classical deconvolution approach failed to improve results, while sub-pixel convolution method achieved to take the second place when whole tumor is considered.

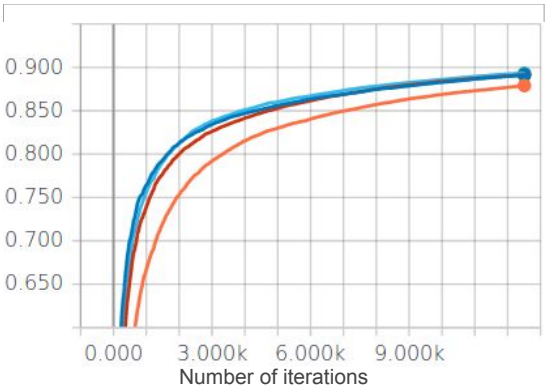
During training, bilinear interpolation has shown slightly worse performance while trainable bilinear interpolation has shown slightly better performance 6.11.

Deconvolution, unlike the training results (Figure 6.12 significantly decreases sensitivity as can be seen in Figures 6.14(b) and 6.15(b), although it is the only method that correctly labeled non-tumor tissue as background.

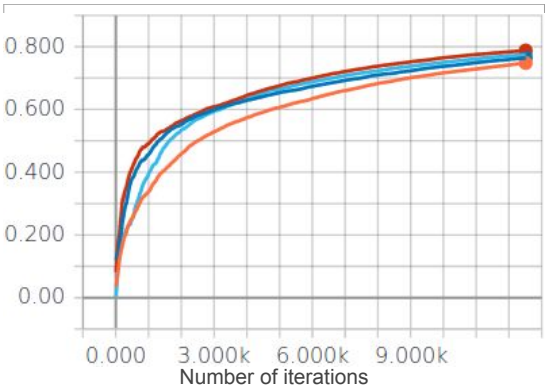
Trainable bilinear interpolation has shown to lose sensitivity. Figure 6.14 (c) shows



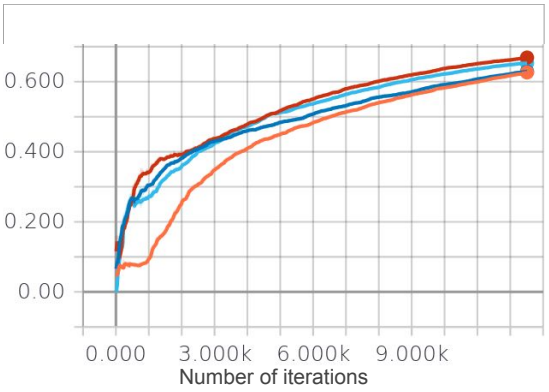
(a) Enhanced Tumor Dice Score



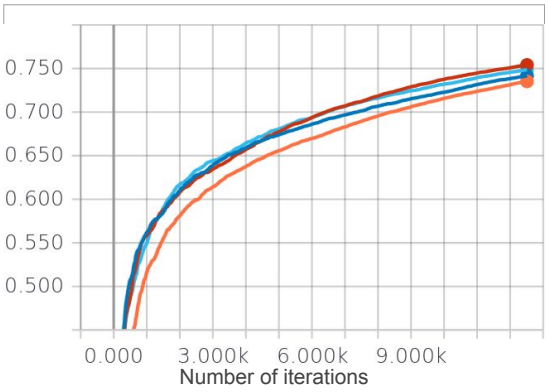
(b) Whole Tumor Dice Score



(c) Tumor Core Dice Score



(d) Necrotic/Non-enhancing Tumor Dice Score



(e) Edema Dice Score

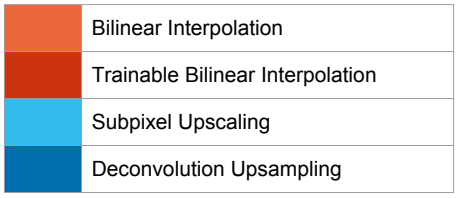
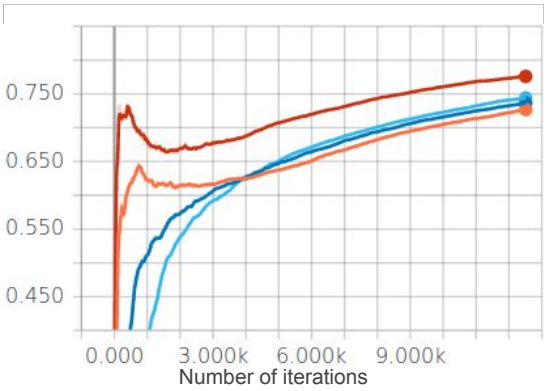


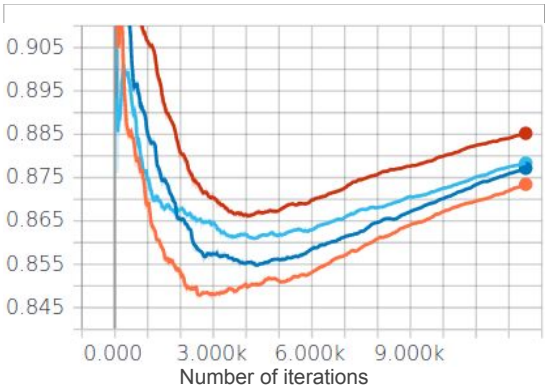
Figure 6.11: Dice scores obtained from training with different upsampling methods.

that the right tail of tumor tissue could not be found by trainable bilinear interpolation.

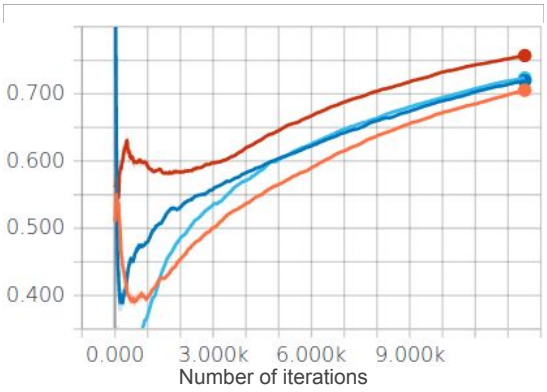
Subpixel upscaling has the best specificity results for the training 6.13. However, it is not the case for the validation results 6.4.



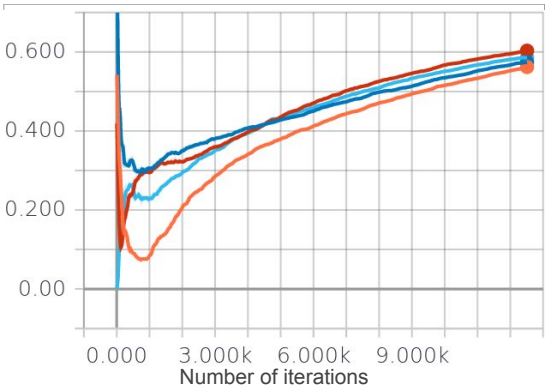
(a) Enhanced Tumor Sensitivity Result



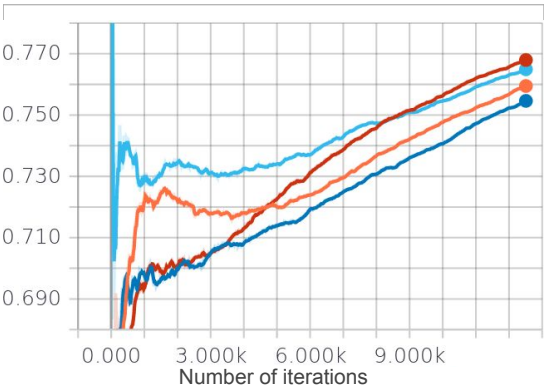
(b) Whole Tumor Sensitivity Result



(c) Tumor Core Sensitivity Result



(d) Necrotic/Non-enhancing Tumor Sensitivity Result



(e) Edema Sensitivity Result

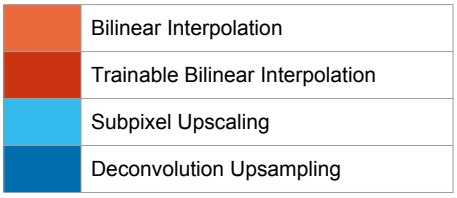
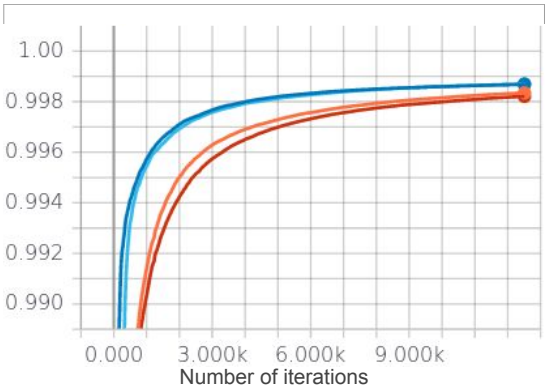
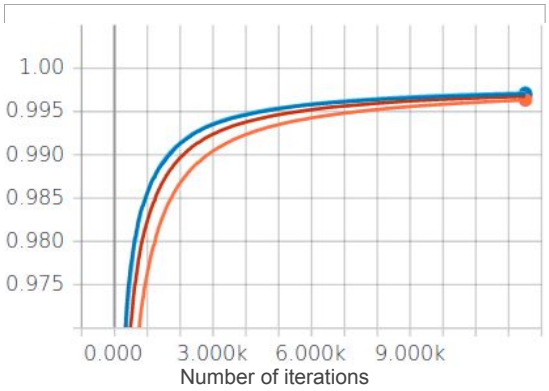


Figure 6.12: Sensitivity results obtained from training with different upsampling methods.

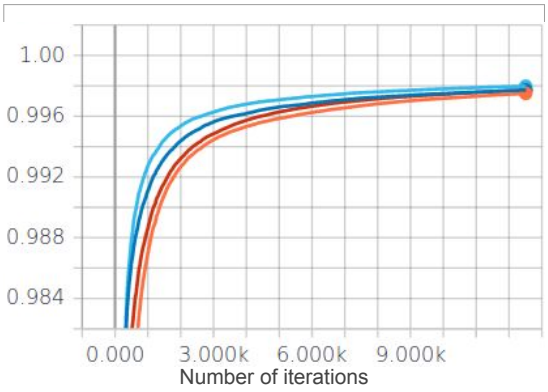
When LGG results are considered (Figure 6.15), deconvolution has the worst results while bilinear interpolation is obviously the top performing method.



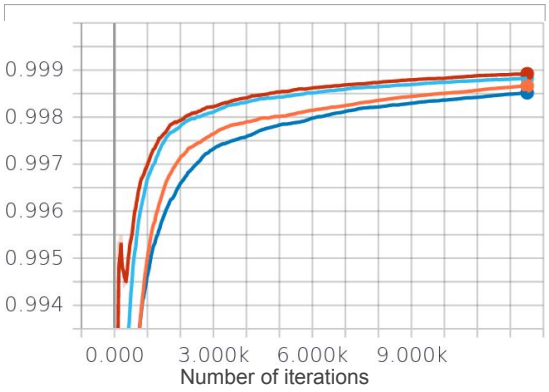
(a) Enhanced Tumor Specificity Result



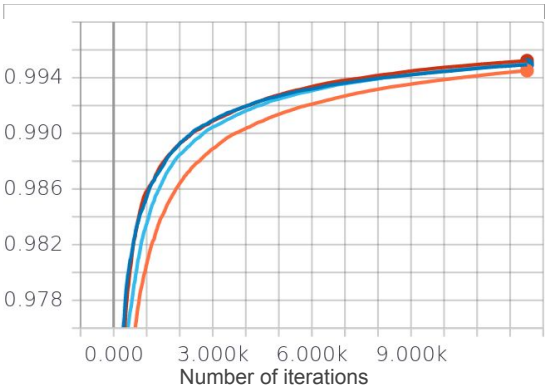
(b) Whole Tumor Specificity Result



(c) Tumor Core Specificity Result



(d) Necrotic/Non-enhancing Tumor Specificity Result



(e) Edema Specificity Result

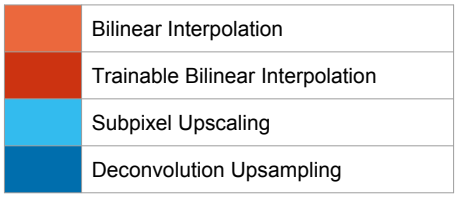
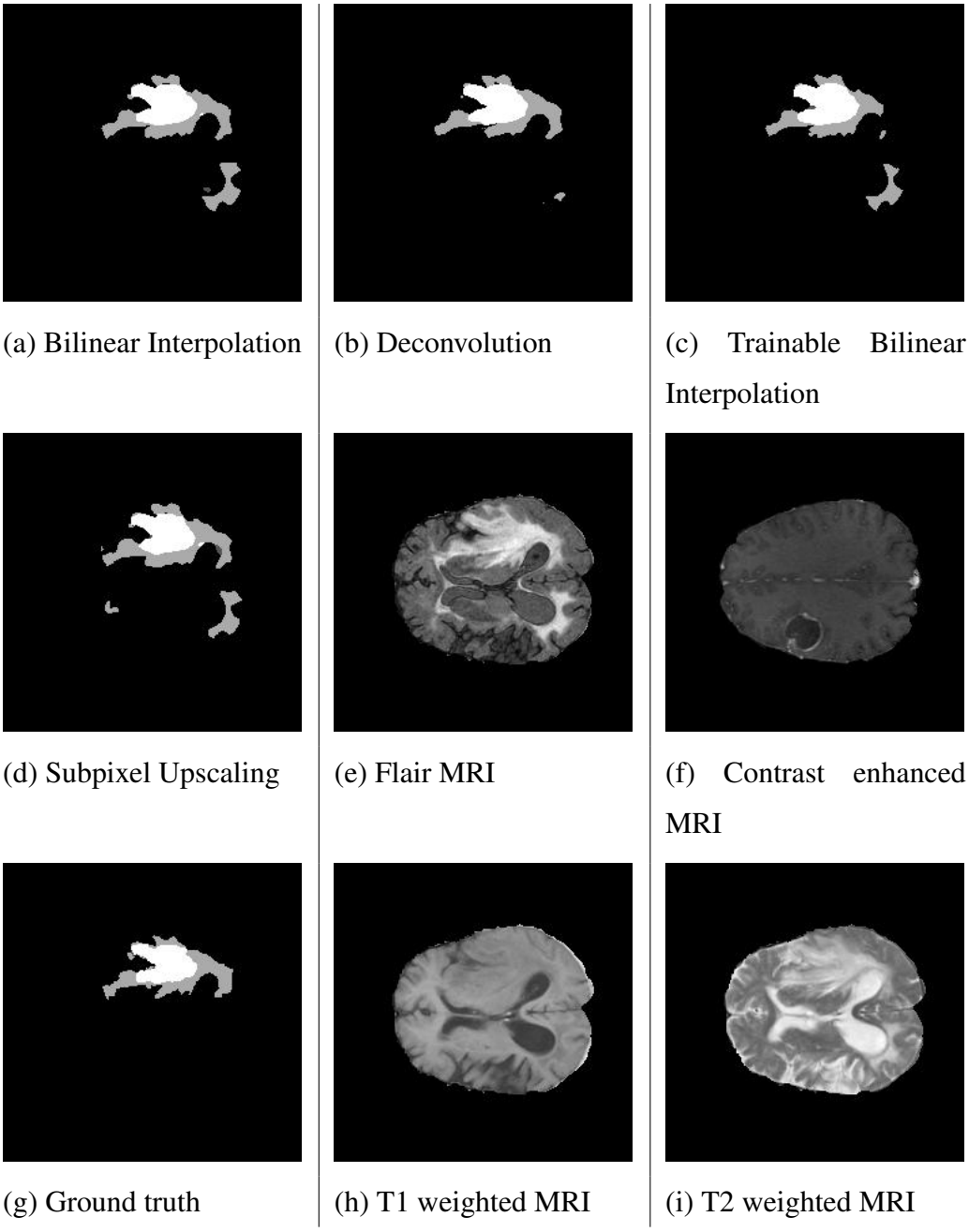


Figure 6.13: Specificity results obtained from training with different upsampling methods.

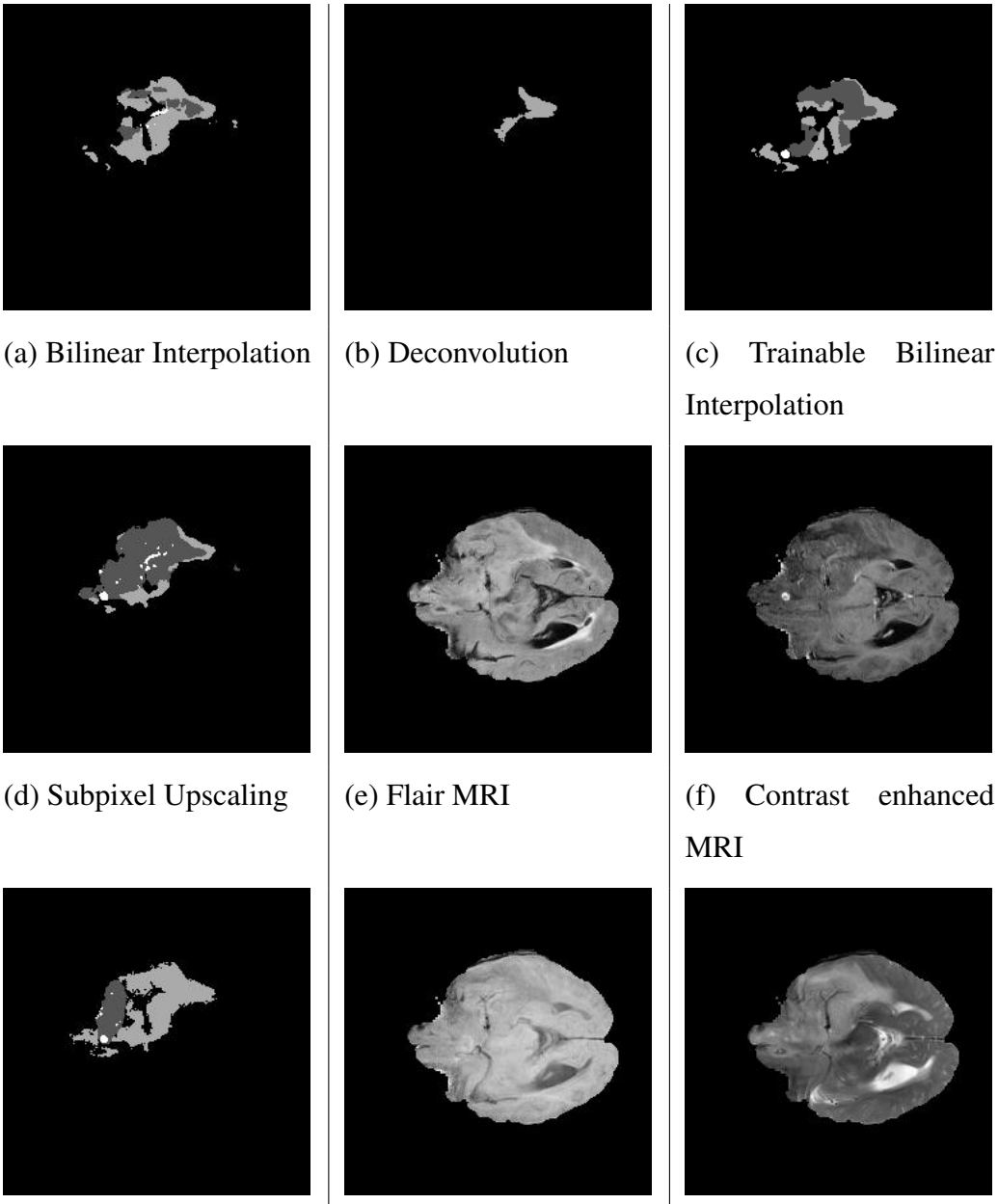


(a) Bilinear Interpolation (b) Deconvolution (c) Trainable Bilinear Interpolation

(d) Subpixel Upscaling (e) Flair MRI (f) Contrast enhanced MRI

(g) Ground truth (h) T1 weighted MRI (i) T2 weighted MRI

Figure 6.14: Examples of validation results of HGG patients obtained using different upsampling methods.



(a) Bilinear Interpolation (b) Deconvolution (c) Trainable Bilinear Interpolation

(d) Subpixel Upscaling (e) Flair MRI (f) Contrast enhanced MRI

(g) Ground truth (h) T1 weighted MRI (i) T2 weighted MRI

Figure 6.15: Examples of validation results of LGG patients obtained using different upsampling methods.

Table 6.5: Validation results obtained using different cost functions

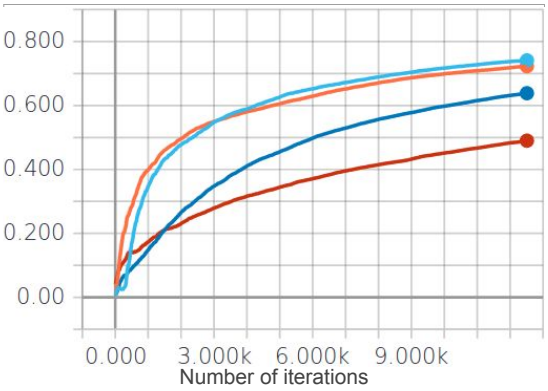
	High Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Cross Entropy	0.509	0.748	**0.827**	**0.874**	0.801	0.514	0.706	0.809	0.837	0.790	0.999	0.998	0.999	0.999	0.999
Weighted Cross Entropy	0.217	0.461	0.674	0.532	0.474	0.693	0.816	**0.913**	**0.979**	**0.933**	0.993	0.985	0.997	0.977	0.991
Cross Entropy + Adversarial Loss (GAN)	0.591	0.728	0.819	0.854	**0.819**	0.635	0.726	0.787	0.796	**0.819**	0.999	0.998	0.999	0.999	0.999
Dice Loss	0.623	0.772	0.824	**0.874**	**0.859**	0.629	0.754	0.898	0.879	0.914	0.999	0.998	0.999	0.998	0.999
	Low Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Cross Entropy	0.672	0.700	0.602	**0.876**	**0.682**	0.642	0.659	0.523	0.826	0.644	0.998	0.998	0.999	0.999	0.998
Weighted Cross Entropy	0.413	0.426	0.462	0.613	0.460	0.629	0.655	**0.744**	**0.946**	**0.715**	0.990	0.984	0.998	0.978	0.988
Cross Entropy + Adversarial Loss (GAN)	0.645	0.687	0.600	0.852	0.675	0.554	0.689	0.490	0.773	0.602	0.999	0.997	0.999	0.999	0.998
Dice Loss	0.617	0.695	**0.632**	0.872	0.658	0.547	0.714	0.638	0.851	0.595	0.998	0.996	0.999	0.998	0.998
	Average of HGG and LGG Results														
	Dice Coefficient					Sensitivity					Specificity				
	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC	NT	ED	ET	WT	TC
Cross Entropy	0.591	0.724	0.715	**0.875**	0.742	0.578	0.683	0.666	0.832	0.717	0.999	0.998	0.999	0.999	0.999
Weighted Cross Entropy	0.315	0.443	0.568	0.573	0.467	0.661	0.735	**0.829**	**0.963**	**0.824**	0.991	0.985	0.998	0.978	0.990
Cross Entropy + Adversarial Loss (GAN)	0.618	0.708	0.710	0.853	0.747	0.595	0.708	**0.639**	**0.785**	**0.711**	0.999	0.998	0.999	0.999	0.999
Dice Loss	0.620	0.734	**0.728**	0.873	**0.759**	0.588	0.734	0.768	0.865	0.755	0.999	0.997	0.999	0.998	0.999

6.3.2.4 Comparison of Different Loss Functions

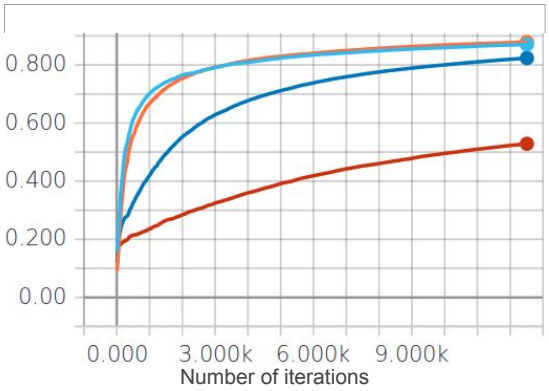
In theory, we would expect a weighted loss would perform better due to the class imbalance problem. Unfortunately, it is not the case in real. [52] also states this by saying that the weighted loss affects the way network is trained and biases are arranged according to the weighted loss. Thus, as shown in Table 6.5 it results in an over-sensitive but non-specific result for classes with higher weights.

The results show that cross entropy performs better than dice loss in terms of dice score 6.16. Similar to training results (Figures 6.16,6.17,6.18), it can be seen on Figures 6.19 6.20(a),(b) that dice loss increases the sensitivity compared to cross entropy. However, it lacks specificity (Figure 6.18). The over sensitivity of the weighted cross entropy can be observed in Figures 6.19(c) and 6.20(c).

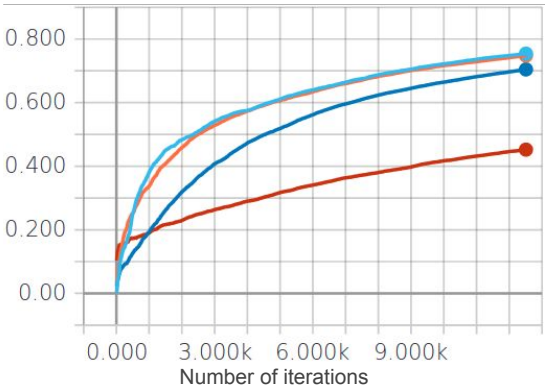
GAN did not improve the result as expected since it is added as a regularization. In some cases, although it can find the whole tumor region, it fails to correctly classify sub-classes significantly for the LGG patients (Figure 6.22(d)).



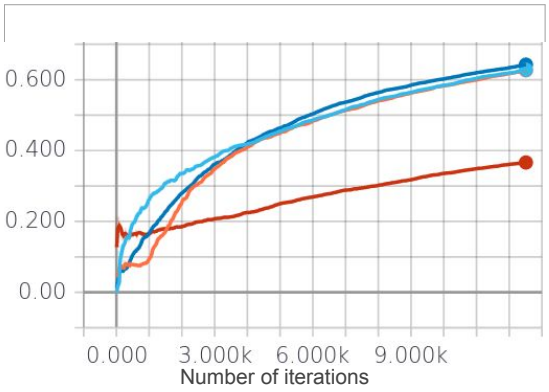
(a) Enhanced Tumor Dice Score



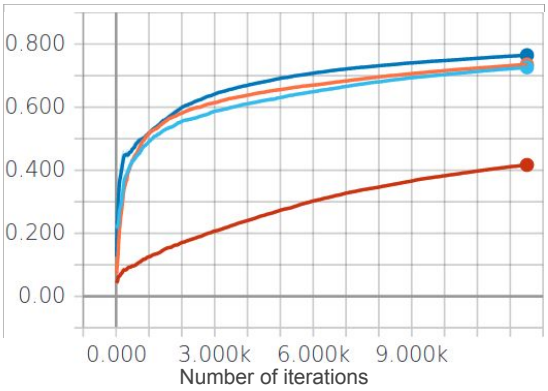
(b) Whole Tumor Dice Score



(c) Tumor Core Dice Score



(d) Necrotic/Non-enhancing Tumor Dice Score



(e) Edema Dice Score

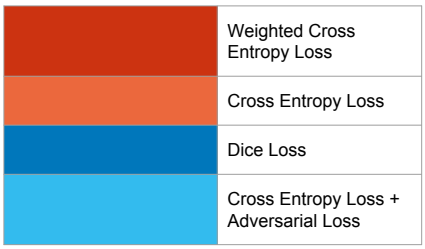
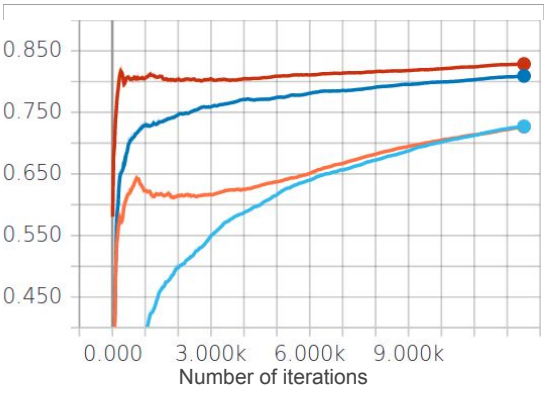
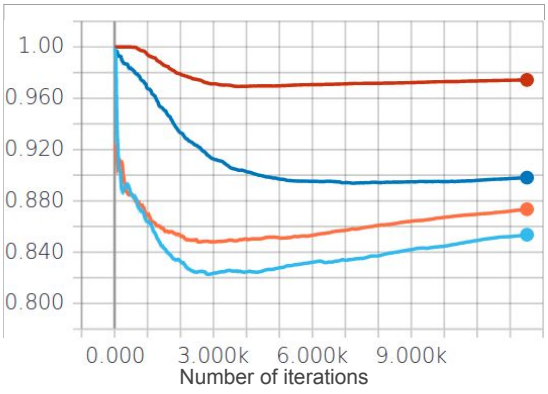


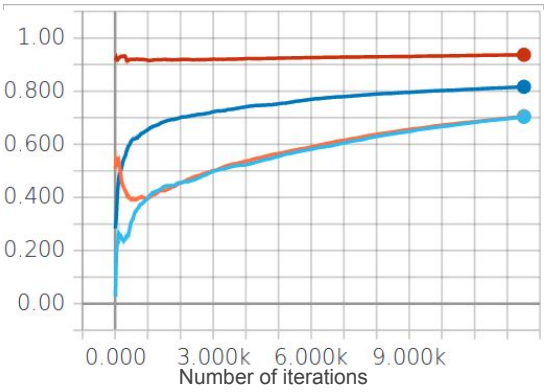
Figure 6.16: Dice scores obtained from training with different loss functions.



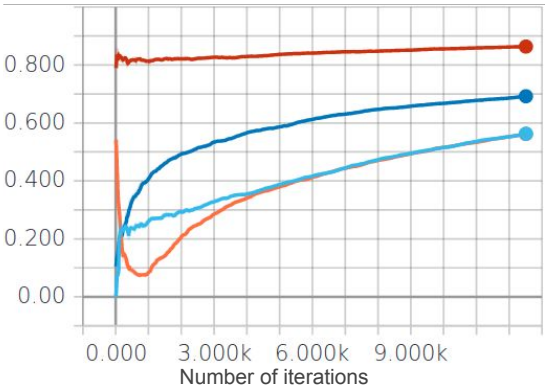
(a) Enhanced Tumor Sensitivity Result



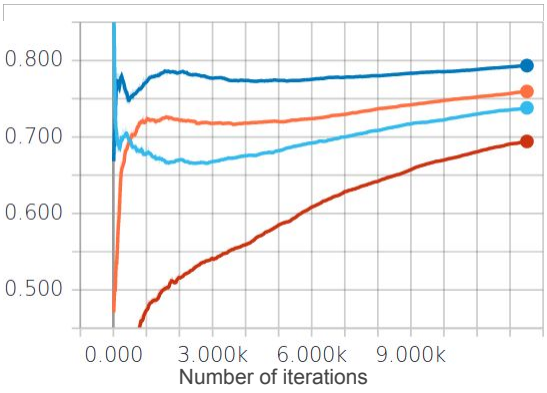
(b) Whole Tumor Sensitivity Result



(c) Tumor Core Sensitivity Result



(d) Necrotic/Non-enhancing Tumor Sensitivity Result



(e) Edema Sensitivity Result

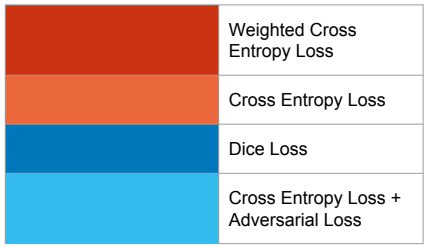
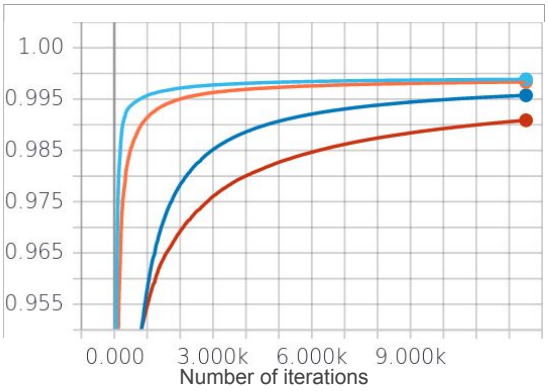
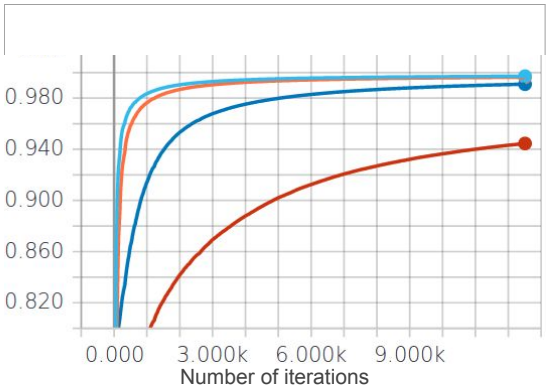


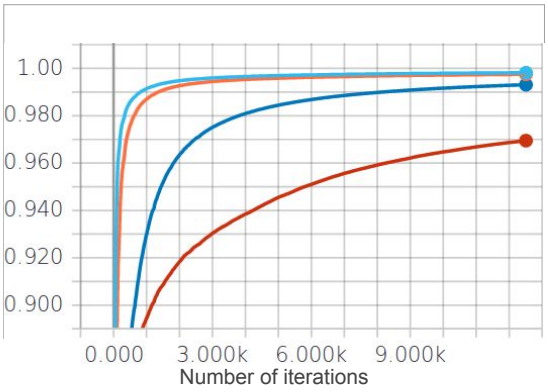
Figure 6.17: Sensitivity results obtained from training with different loss functions.



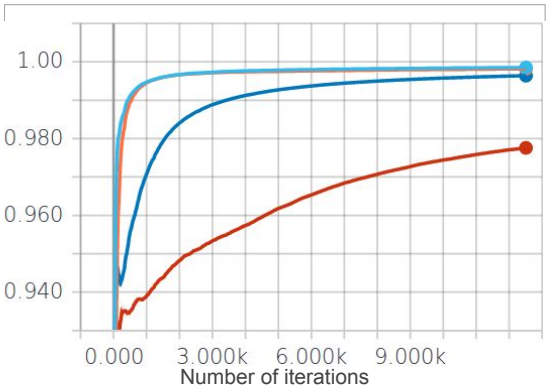
(a) Enhanced Tumor Specificity Result



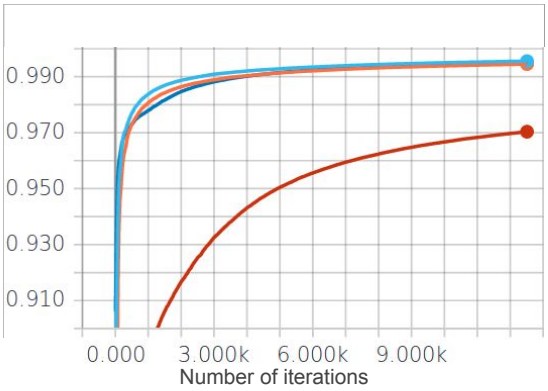
(b) Whole Tumor Specificity Result



(c) Tumor Core Specificity Result



(d) Necrotic/Non-enhancing Tumor Specificity Result



(e) Edema Specificity Result

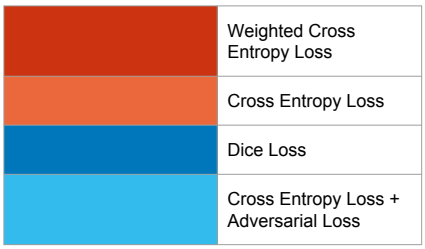
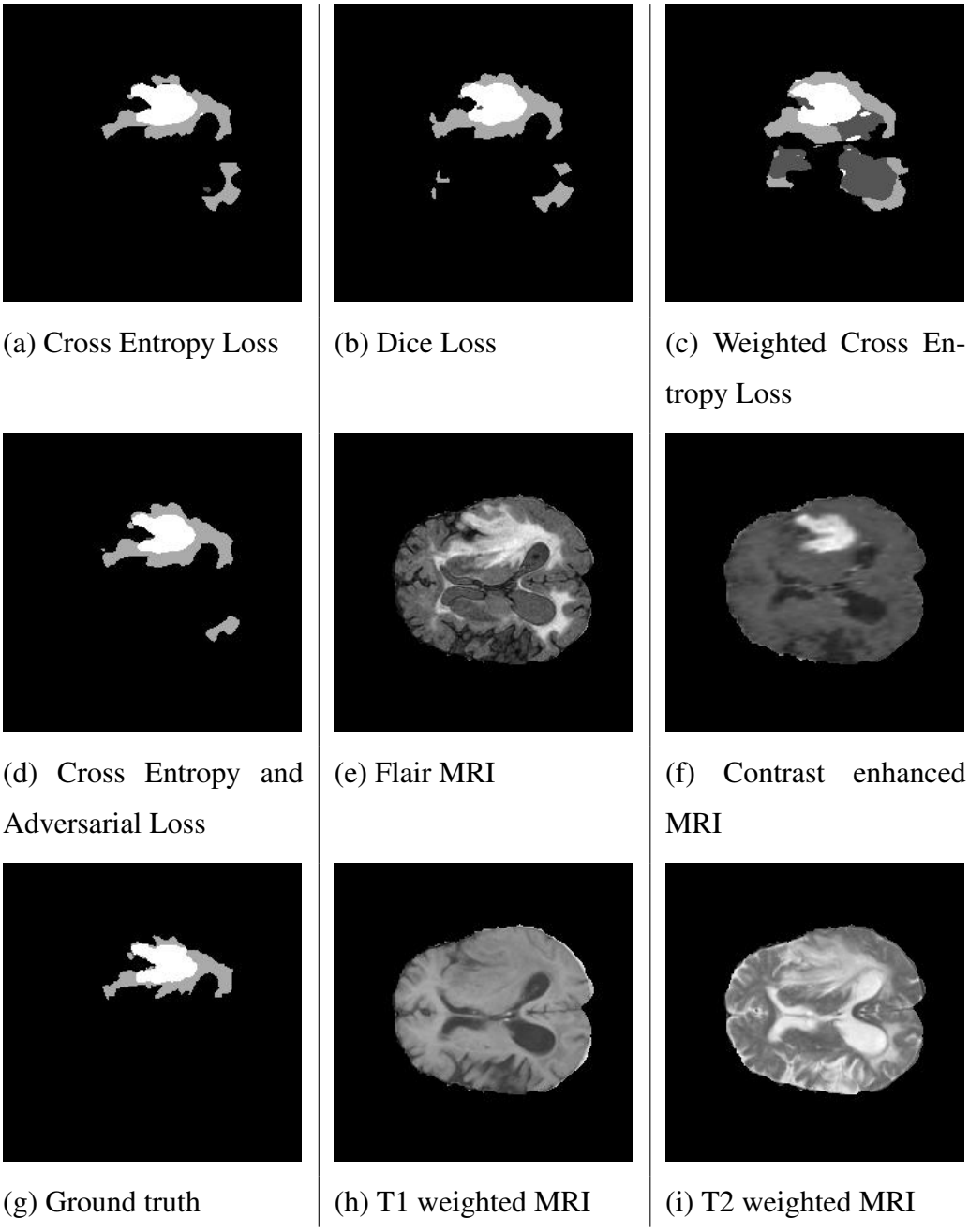


Figure 6.18: Specificity results obtained from training with different loss functions.



(a) Cross Entropy Loss

(b) Dice Loss

(c) Weighted Cross Entropy Loss

(d) Cross Entropy and Adversarial Loss

(e) Flair MRI

(f) Contrast enhanced MRI

(g) Ground truth

(h) T1 weighted MRI

(i) T2 weighted MRI

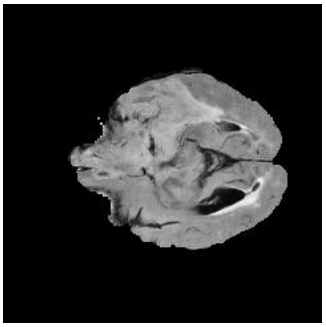
Figure 6.19: Examples of validation results of HGG patients obtained using different loss functions.

(a) Cross Entropy Loss

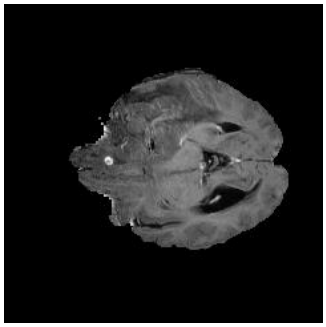
(b) Dice Loss

(c) Weighted Cross Entropy Loss

(d) Cross Entropy and Adversarial Loss

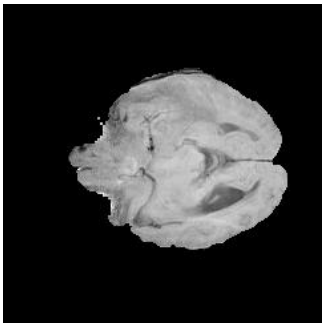


(e) Flair MRI

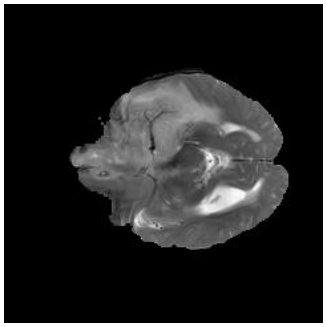


(f) Contrast enhanced MRI

(g) Ground truth



(h) T1 weighted MRI



(i) T2 weighted MRI

Figure 6.20: Examples of validation results of LGG patients obtained using different loss functionss.

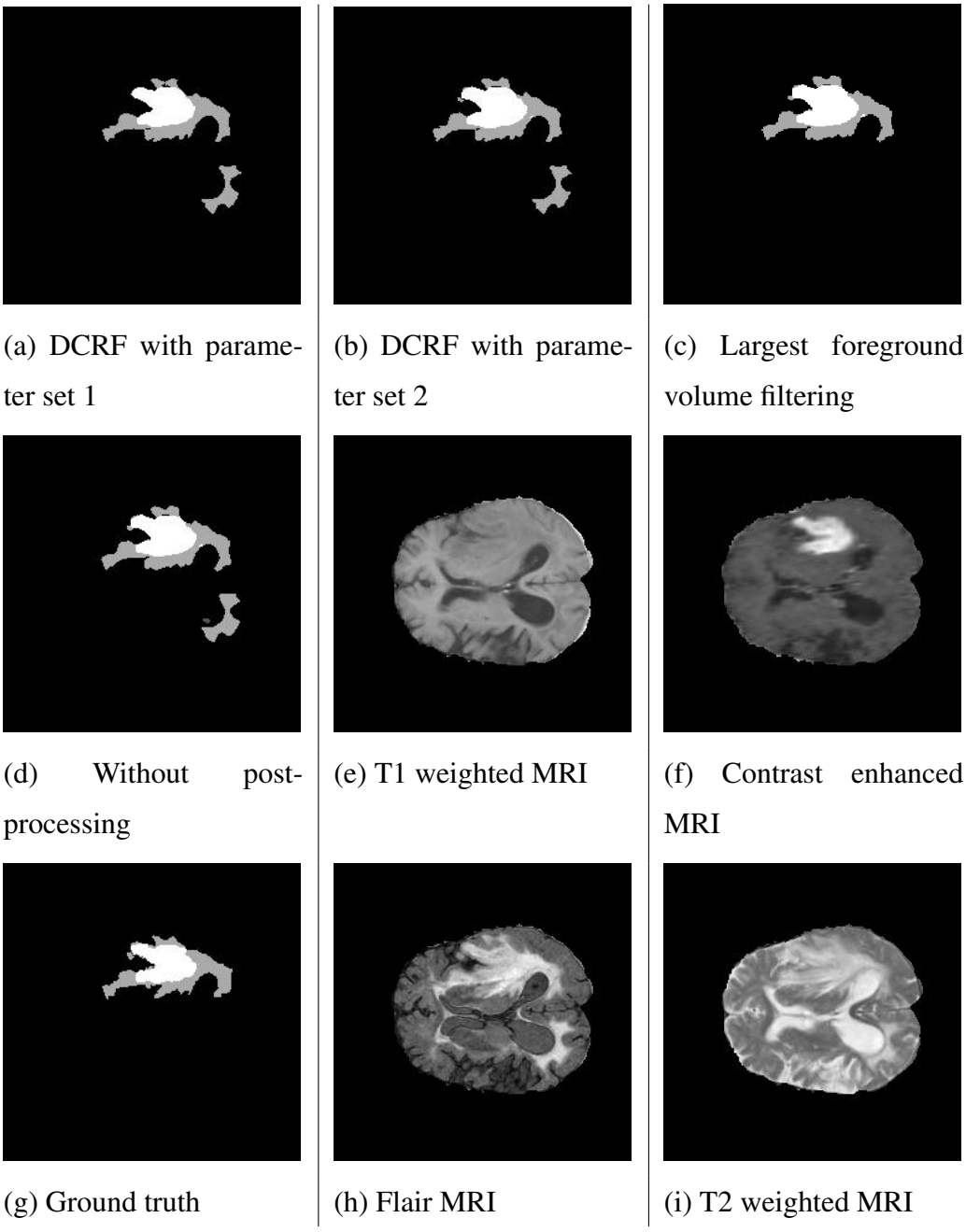
Table 6.6: Validation results obtained using different post processing methods. DCRF-1 and DCRF-2 correspond to conditional random field applied with different parameter choices.

	High Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC
Original	0.510	0.749	**0.827**	0.874	0.801	0.505	0.709	**0.809**	**0.837**	0.790	0.999	0.998	0.999	0.999	0.999
DCRF-1	0.465	0.653	0.760	0.817	0.744	0.465	0.580	0.736	0.734	0.743	0.999	0.999	0.999	0.999	0.999
DCRF-2	0.498	0.712	0.800	0.849	0.764	0.498	0.664	0.799	0.796	0.782	0.999	0.999	0.999	0.999	0.999
Largest Foreground Volume Selection	0.512	0.756	0.816	**0.891**	0.790	0.517	0.724	0.808	0.826	0.778	0.999	0.999	0.999	0.999	0.999
	Low Grade Gliomas														
	Dice Coefficient					Sensitivity					Specificity				
	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC
Original	0.672	0.698	**0.602**	**0.876**	**0.682**	0.641	0.658	0.523	**0.826**	0.644	0.998	0.997	0.999	0.999	0.998
DCRF-1	0.570	0.607	0.457	0.838	0.613	0.577	0.543	0.486	0.753	0.606	0.999	0.999	0.999	0.999	0.999
DCRF-2	0.590	0.620	0.444	0.862	0.626	0.590	0.620	0.561	0.802	0.623	0.999	0.999	0.999	0.999	0.998
Largest Foreground Volume Selection	0.530	0.641	0.392	0.890	0.623	0.579	0.655	0.507	0.813	0.618	0.998	0.998	0.999	0.999	0.998
	Average of HGG and LGG Results														
	Dice Coefficient					Sensitivity					Specificity				
	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC	NT/NCR	ED	ET	WT	TC
Original	0.591	0.723	**0.715**	0.875	0.742	0.573	0.683	0.666	**0.832**	0.717	0.999	0.998	0.999	0.999	0.999
DCRF-1	0.518	0.630	0.608	0.827	0.678	0.521	0.561	**0.611**	0.744	**0.674**	0.999	0.999	0.999	0.999	0.999
DCRF-2	0.544	0.666	0.622	0.855	0.695	0.544	0.642	0.680	0.799	0.703	0.999	0.999	0.999	0.999	0.999
Largest Foreground Volume Selection	0.521	0.699	0.604	**0.891**	0.707	0.548	0.689	0.658	0.820	0.698	0.999	0.998	0.999	0.999	0.999

6.3.2.5 Comparison of Different Post-processing Methods

Our post-processing methods include conditional random fields and generative (conditional) adversarial networks. We have adapted a CRF which aims to find a maximum a posteriori, from the soft output of the segmentation network, which maximizes the probability of neighbor pixels' belonging to the same class. However, this approach is not very practical in a multi-class problem since there might exist only few pixels surrounded by other classes. Moreover, the parameters in the energy function needs to be set very carefully for the specific problem. Thus, we have not observed any improvements using CRF as a post-processor 6.6. With two different parameter sets, DCRF has decreased the sensitivity of whole tumor significantly on LGG results 6.22(a),(b).

Finally, as can be seen in Figures 6.21(e) and 6.22(e), largest foreground volume filtering helps avoiding some false negatives.



(a) DCRF with parameter set 1

(b) DCRF with parameter set 2

(c) Largest foreground volume filtering

(d) Without post-processing

(e) T1 weighted MRI

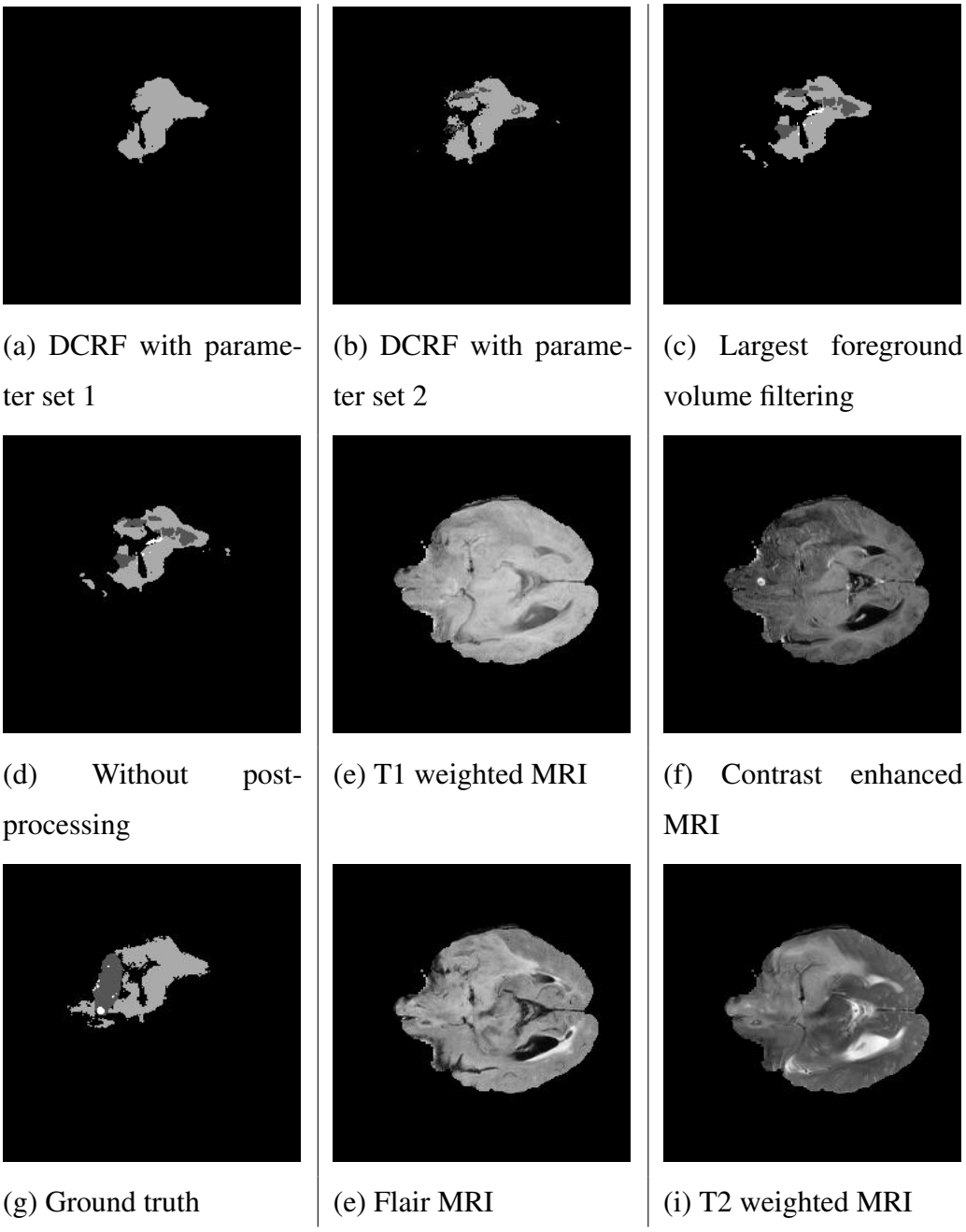
(f) Contrast enhanced MRI

(g) Ground truth

(h) Flair MRI

(i) T2 weighted MRI

Figure 6.21: Examples of validation results of HGG patients obtained using different upscale methods.



(a) DCRF with parameter set 1

(b) DCRF with parameter set 2

(c) Largest foreground volume filtering

(d) Without post-processing

(e) T1 weighted MRI

(f) Contrast enhanced MRI

(g) Ground truth

(e) Flair MRI

(i) T2 weighted MRI

Figure 6.22: Examples of validation results of LGG patients obtained using different upscale methods.

The results of the top performing methods in BraTS 2017 validation phase are given in Table 6.7 ([52], [43], [60]).

Table 6.7: Validation results of top performing methods as reported in BraTS 2017 leaderboard.

	Dice Coefficient			Sensitivity			Specificity		
	ET	WT	TC	ET	WT	TC	ET	WT	TC
biomedia1 [52]	0.738	0.901	0.797	0.783	0.895	0.762	0.998	0.995	0.998
MIC_DKFZ [43]	0.776	0.903	0.819	0.803	0.902	0.786	0.998	0.996	0.999
UCL-TIG [60]	0.786	0.905	0.838	0.771	0.915	0.822	0.999	0.995	0.998
Multi-scale U-Net with PReLU	0.741	0.870	0.769	0.674	0.816	0.753	0.999	0.999	0.999

It can be seen from Table 6.7 that our method can not reach the performance of top performing methods. Although it produces more specific results, it lacks sensitivity. It is important to note that all of the three methods have a 3D network architecture. Moreover, [52] uses multiple networks (six separately trained networks) and combines their result. Similarly, [60] has 3 cascaded networks that extracts whole tumor, tumor core and enhanced tumor regions respectively. By using findings in this thesis, the performance of such complex architectures can be improved further.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this work, we have investigated the performance of different CNN based methods applied to brain MRI segmentation problem.

First, different FCN architectures, namely, FCN-8s and U-Net, were experimented. Combination of two separate U-Nets that have 3×3 and 5×5 filter sizes respectively was chosen as the based method due to its performance. On the selected base architecture, structural modifications which are inception modules, dilation modules and additional residual connections have been inserted. The most significant performance improvement was achieved by adding dilation modules. Residual connections have also improved the result while inception modules have shown to disturb the performance.

Among activation functions, PReLU and ReLU had the best results without a significant difference in between. As expected, tanh has the worst performance while LReLU needs to be tuned for its leakage coefficient.

Bilinear interpolation, despite being non-trainable, has shown the best performance overall. Among others, sub-pixel was the second best. Trainable bilinear interpolation has shown better results on HGG data undoubtedly because there are more training samples than LGG data.

For the loss functions, cross entropy loss and dice loss yield similar results. On the contrary, weighted cross entropy loss, which would be expected to solve the imbalanced class problem, has produced over-sensitive and unconfident results. Although GAN would be expected to act as a regularizer, it did not improve the results either.

For the post-processing, DCRF did not perform well because its hyperparameters need fine-tuning. In spite of its simplicity, filtering the largest 3D connected foreground volume has increased the results of whole tumor, while it adversely affected sub-classes.

To conclude, considering that the state-of-the-art methods use a 3D network, it is remarkable to achieve similar results by a 2D network. Our best performing method, without post-processing, achieves a dice score of 0.875, 0.742, 0.715 for WT, TC and ET respectively while the best results reported in the BraTS 2017 are 0.901 , 0.825, 0.764([52, 60]). When the post-processing is included, we reach a dice score of 0.891 for whole tumor.

In future, we plan to expand our algorithms to 3D domain since we could not achieved the state-of-the-art performance with a 2D network. However, this will limit the usage of the network because the depth of the MRI data can change. We will also expand the CRF post-processing by adding temporal information. Additionally, in some cases, state-of-the-art methods trained a separate network and proposed a cascaded system [60]. Another approach that can improve the results is to pre-train the network on large datasets such as ImageNet [92]. We plan to pre-train the downsampling part of the U-Net or FCN-8s, by adding a fully connected layer at the end. Then we can use the parameters of this pre-trained network, excluding the first layer (due to the change in the number of channels) and fully connected layer at the end.

REFERENCES

[1] M. Tayyab, "Conduction of nerve impulse."

[2] G. J. B. Orr, "Momentum and learning rate adaptation."

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[4] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints*, mar 2016.

[5] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," *CoRR*, vol. abs/1609.05158, 2016.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[7] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016.

[9] N. I. of Biomedical Imaging and Bioengineering, "Magnetic resonance imaging (mri)."

[10] OECD, "Magnetic resonance imaging (mri) exams (indicator)," 2018. doi: 10.1787/1d89353f-en.

[11] ABTA, "Brain tumor statistics," Jan. 2018.

[12] K. Haris, S. N. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos, "Hybrid image segmentation using watersheds and fast region merging," *IEEE Transactions on Image Processing*, vol. 7, pp. 1684–1699, Dec 1998.

[13] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient n-d image segmentation," *Int. J. Comput. Vision*, vol. 70, pp. 109–131, Nov. 2006.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[15] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. R. Gerstner, M.-A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. T. Ryan, D. Sarikaya, L. H. Schwartz, H.-C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Székely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. B. Ünal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. V. Leemput, "The multimodal brain tumor image segmentation benchmark (brats).," *IEEE Trans. Med. Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015.

[16] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, J. Freymann, K. Farahani, and C. Davatzikos, "Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features," *Scientific data*, vol. 4, 9 2017.

[17] B. Song, C.-R. Chou, X. Chen, A. Huang, and M.-C. Liu, "Anatomy-guided brain tumor segmentation and classification," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, B. Menze, O. Maier, M. Reyes, S. Winzeck, and H. Handels, eds.), (Cham), pp. 162–170, Springer International Publishing, 2016.

[18] H. Hooda, O. P. Verma, and T. Singhal, "Brain tumor segmentation: A performance analysis using k-means, fuzzy c-means and region growing algorithm," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 1621–1626, May 2014.

[19] L. Szilágyi, L. Lefkovits, and B. Benyó, "Automatic brain tumor segmentation in multispectral mri volumes using a fuzzy c-means cascade algorithm," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp. 285–291, Aug 2015.

[20] M. Anitha, T. Selvy, and V. Palanisamy, "Automated detection of white matter lesions in brain images using spatio-fuzzy and spatio possibilistic clustering models," *Computer Science and Engineering: An International Jo urnal*, vol. 2, 2012.

[21] N. Menon and R. Ramakrishnan, "Brain tumor segmentation in mri images using unsupervised artificial bee colony algorithm and fcm clustering," in *2015 International Conference on Communications and Signal Processing (ICCSP)*, pp. 0006–0009, April 2015.

[22] E. A. Rios Piedra, B. M. Ellingson, R. K. Taira, S. El-Saden, A. A. T. Bui, and W. Hsu, "Brain tumor segmentation by variability characterization of tumor boundaries," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, B. Menze, O. Maier, M. Reyes, S. Winzeck, and H. Handels, eds.), (Cham), pp. 206–216, Springer International Publishing, 2016.

[23] R. Achanta, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels."

[24] N. S. G. and V. Khairnar, "Brain tumor detection based on symmetry information," *CoRR*, vol. abs/1401.6127, 2014.

[25] N. S. G., D. Shah, V. Khairnar, and S. Kadu, "Brain tumor detection based on bilateral symmetry information," *CoRR*, vol. abs/1412.3009, 2014.

[26] A. Bianchi, J. Miller, E. Tan, and A. Montillo, "Brain tumor segmentation with symmetric texture and symmetric intensity-based decision forests," vol. 2013, pp. 748–51, 04 2013.

[27] P. Aljabar, R. Heckemann, A. Hammers, J. Hajnal, and D. Rueckert, "Multi-atlas based segmentation of brain images: Atlas selection and its effect on accuracy," *NeuroImage*, vol. 46, no. 3, pp. 726 – 738, 2009.

[28] S. Bauer, C. Seiler, T. Bardyn, P. Buechler, and M. Reyes, "Atlas-based segmentation of brain tumor images using a markov random field-based tumor growth model and non-rigid registration," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 4080–4083, Aug 2010.

[29] M. T. M. Park, J. Pipitone, L. H. Baer, J. L. Winterburn, Y. Shah, S. Chavez, M. M. Schira, N. J. Lobaugh, J. P. Lerch, A. N. Voineskos, and M. M. Chakravarty, "Derivation of high-resolution mri atlases of the human cerebellum at 3t and segmentation using multiple automatically generated templates," *NeuroImage*, vol. 95, pp. 217 – 231, 2014.

[30] R. Ayachi and N. Ben Amor, "Brain tumor segmentation using support vector machines," in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (C. Sossai and G. Chemello, eds.), (Berlin, Heidelberg), pp. 736–747, Springer Berlin Heidelberg, 2009.

[31] S. Bauer, L.-P. 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 *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011* (G. Fichtinger, A. Martel, and T. Peters, eds.), (Berlin, Heidelberg), pp. 354–361, Springer Berlin Heidelberg, 2011.

[32] L. Lefkovits, S. Lefkovits, and L. Szilágyi, "Brain tumor segmentation with optimized random forest," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, B. Menze, O. Maier, M. Reyes, S. Winzeck, and H. Handels, eds.), (Cham), pp. 88–99, Springer International Publishing, 2016.

[33] Y. Li, F. Jia, and J. Qin, "Brain tumor segmentation from multimodal magnetic resonance images via sparse representation," *Artificial Intelligence in Medicine*, vol. 73, pp. 1 – 13, 2016.

[34] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, pp. 640–651, Apr. 2017.

[35] K. Kamnitsas, C. Ledig, V. F. J. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient multi-scale 3d CNN with fully connected CRF for accurate brain lesion segmentation," *CoRR*, vol. abs/1603.05959, 2016.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[37] P. Amorim and V. Chagas, "3d u-nets for brain tumor segmentation in miccai 2017 brats challenge," in *MICCAI BraTS 2017 Proceedings*, 2017.

[38] A. Beers, K. Chang, J. M. Brown, E. Sartor, C. P. Mammen, E. R. Gerstner, B. R. Rosen, and J. Kalpathy-Cramer, "Sequential 3d u-nets for biologically-informed brain tumor segmentation," *CoRR*, vol. abs/1709.02967, 2017.

[39] S. cao, B. Qian, C. Yin, X. Li, and S. Chang, "3d u-net for multimodal brain tumor segmentation," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[40] L. S. Castillo, L. A. Daza, L. C. Rivera, and P. Arbeláez, "Volumetric multimodality neural network for brain tumor segmentation," 2017.

[41] X. feng and C. Meyer, "Patch based 3d u-net for brain tumor segmentation," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[42] M. Catà, A. Casamitjana, I. Sánchez, M. Combalia, and V. Vilaplana, "Masked v-net: an approach to brain tumor segmentation," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[43] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[44] C. Wang and O. Smedby, "Automatic brain tumor segmentation using 2.5 d u-nets," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[45] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," *CoRR*, vol. abs/1606.06650, 2016.

[46] F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," *CoRR*, vol. abs/1606.04797, 2016.

[47] V. Alex, M. Safwan, and G. Krishnamurthi, "Brain tumor segmentation from multi modal mr images using fully convolutional neural network," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[48] Y. Hu and Y. Xia, "Automatic brain tumor segmentation using a 3d deep detection classification model," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[49] M. Soltaninejad, L. Zhang, T. LAmbrou, G. Yang, N. Allinson, and X. Ye, "Mri brain tumor segmentation using random forests and fully convolutional networks," in *MICCAI BraTS 2017 Proceedings*, 2017.

[50] P. D. Chang, "Fully convolutional neural networks with hyperlocal features for brain tumor segmentation," in *MICCAI BraTS 2016 Proceedings*, 2016.

[51] K. Kamnitsas, E. Ferrante, S. Parisot, C. Ledig, A. Nori, A. Criminisi, D. Rueckert, and B. Glocker, "Deepmedic for brain tumor segmentation," in *MICCAI Brain Lesion Workshop*, October 2016.

[52] K. Kamnitsas, W. Bai, E. Ferrante, S. G. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. C. H. Lee, B. Kainz, D. Rueckert, and B. Glocker, "Ensembles of multiple models and architectures for robust brain tumour segmentation," *CoRR*, vol. abs/1711.01468, 2017.

[53] X. Li, X. Zhang, and Z. Luo, "Brain tumor segmentation via 3d fully dilated convolutional networks," in *Multimodal Brain Tumor Segmentation Benchmark, Brain-lesion Workshop, MICCAI 2017*, 09/2017 2017.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[56] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[58] G. Kim, "Brain tumor segmentation using deep u-net," in *MICCAI BraTS 2017 Proceedings*, 2017.

[59] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks," *CoRR*, vol. abs/1709.00382, 2017.

[61] M. Rezaei, K. Harmuth, W. Gierke, T. Kellermeier, M. Fischer, H. Yang, and C. Meinel, "Conditional adversarial network for semantic segmentation of brain tumor," *CoRR*, vol. abs/1708.05227, 2017.

[62] L. G. Nyul, J. K. Udupa, and X. Zhang, "New variants of a method of mri scale standardization," *IEEE Transactions on Medical Imaging*, vol. 19, pp. 143–150, Feb 2000.

[63] K. Pawar, Z. Chen, N. J. Shah, and G. Egan, "Residual encoder and convolutional decoder neural network for glioma segmentation," in *MICCAI BraTS 2017 Proceedings*, 2017.

[64] H. Nharath, S. Colleman, D. Sima, and S. V. Huffel, "Tumor segmentation from multi-parametric mri using random forest with superpixel and tensor based feature extraction," in *MICCAI BraTS 2017 Proceedings*, 2017.

[65] A. Ellwaa, A. Hussein, E. AlNaggar, M. Zidan, M. Zaki, M. A. Ismail, and N. M. Ghanem, "Brain tumor segmantation using random forest trained on iteratively selected patients," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, B. Menze, O. Maier, M. Reyes, S. Winzeck, and H. Handels, eds.), (Cham), pp. 129–137, Springer International Publishing, 2016.

[66] R. Meier, S. Bauer, J. Slotboom, R. Wiest, and M. Reyes, "Appearance- and context-sensitive features for brain tumor segmentation," in *MICCAI BraTS 2014 Proceedings*, 2014.

[67] M. Shaikh, G. Anand, G. Acharya, A. Amrutkar, V. Alex, and G. Krishnamurthi, "Brain tumor segmentation using dense fully convolutional neural network," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, eds.), (Cham), pp. 309–319, Springer International Publishing, 2018.

[68] F. Zhou, T. Li, H. Li, and H. Zhu, "Tpcnn: Two-phase patch-based convolutional neural network for automatic brain tumor segmentation and survival prediction," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, eds.), (Cham), pp. 274–286, Springer International Publishing, 2018.

[69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[70] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[71] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *CoRR*, vol. abs/1611.08408, 2016.

[72] Z. Li, Y. Wang, and J. Yu, "Brain tumor segmentation using an adversarial network," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, eds.), (Cham), pp. 123–132, Springer International Publishing, 2018.

[73] W. McCulloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[74] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[75] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 09 1951.

[76] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *Ann. Math. Statist.*, vol. 23, pp. 462–466, 09 1952.

[77] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Erlbaum, 1986.

[78] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[79] N. Qian, "On the momentum term in gradient descent learning algorithms.," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[81] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Tech. Rep. UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[82] T. Tieleman and G. Hinton, "Coursera: Neural networks for machine learning," vol. 4.

[83] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[84] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[85] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[86] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (Washington, DC, USA), pp. 1520–1528, IEEE Computer Society, 2015.

[87] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[88] P. Moeskops, M. Veta, M. W. Lafarge, K. A. J. Eppenhof, and J. P. W. Pluim, "Adversarial training and dilated convolutions for brain MRI segmentation," *CoRR*, vol. abs/1707.03195, 2017.

[89] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[90] Y. A. Rozanov, *Markov Random Fields*, pp. 55–102. New York, NY: Springer New York, 1982.

[91] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *CoRR*, vol. abs/1210.5644, 2012.

[92] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.