Sudan University of Science and Technology
College of Graduate Studies and Scientific Research
Biomedical Engineering Program

Improvement U-Net for MRI Brain Tumour Segmentation by Searching for Suitable Activation Function

تحسين شبكة U- لتقسيم صور ورم الدماغ بالرنين المغناطيسي من خلال البحث عن دالة التنشيط المناسبة

A dissertation submitted in partial fulfillment of requirement for M.Sc. degree in biomedical engineering

By:

Mushtaq Mahyoob Saleh Kassem

Supervisor:

Dr. Mohammed Abdullah

December 2019

الاستهـلال

بسم الله الرحمن الرحيم

قال تعـالى:

﴿ يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ ﴾

سورة [المجادلة الآية:11]

Dedication

To the soul of my Mother who taught me the meaning of life, sacrifices, and love.

To my father the origin of my success who always grands me love, support, and encouragement.

To my brothers and sisters …

To my wife who inspired me all the time and stand beside me (I love you forever).

To all my family and friends……

ACKNOWLEDGMENT

In the name Allah the Most beneficent, the Most Merciful. All thanks to Allah and I beseech His peace and benedictions on the noblest of mankind, Muhammad, peace and blessings of Allah be upon him and the generality of Muslims till the day of accountability.

First, I would like to express my grateful thanks to Sudan country for giving me opportunity to study my master's degree in one of its best Universities. I also would like to thank Sudan University of science and technology for opening the doors of knowledge and education for me. Furthermore, I thank the college of engineering, biomedical department staff, and all teachers that have taught me.

I sincerely express my profound appreciation to my supervisor Dr. Mohammed Abdullah for his guidance throughout the development of this work and also throughout my Master's program. You are indeed a role model. I have gained both academic knowledge and knowledge towards a unique approach to life and situations from you. It is a rare opportunity to work with you and I really appreciate the opportunity. Without any point of doubt, you have left a positive mark in my life and I will always remain grateful. I am very proud to be your student and to be my supervisor.

Similarly, I sincerely appreciate the efforts, contributions and continuous monitoring of the progress of the work from my Dr. Musab Elkheir Salih for his guidance and support throughout this study and throughout my Master's program. You always push me forward. I thank you for the guidance and valuable advices throughout this work. I will forever be grateful to you.

I would like to express my deep thanks to my country 'Yemen' for supporting me with full scholarship during my study. I will always be loyal to you.

Finally, from bottom of my heart I thank my brothers Dr: Sameer, Dr: Mohammed, and Dr: Amr for their always support and stand beside me. (I love you all). Moreover, I thank my friends Husam Hatem and Musab Alnoor for their encouragement and advice.

Table of Contents

List of Figures

List of Tables

List of abbreviations

MRI	Magnetic Reasoning Imaging
LGG	Low-Grade Gliomas
HGG	H-Grade Gliomas
WHO	World Health Organization
ABTA	American Brain Tumor Association
ReLU	Rectified Linear Unit
HardELiSH	Hard Exponential Linear Squashing
ELiSH	Exponential linear Squashing
ELU	Exponential Linear Unit
BraTS	Brain Tumor Segmentation
DCNN	Deep Convolution Neural Network
DNNs	Deep neural networks
FLAIR	Fluid-Attenuated Inversion-Recovery
GPUs	Graphics Processing Units
BNNs	Biological Neural Networks
ANNs	Artificial Neural Networks
FCNs	Fully Convolutional Networks
CNNs	Convolutional Neural Networks
MIMO	Multiple-Input, Multiple-Output
Cho	Choline-containing compounds
NAA	N-Acetyl-Aspartate
MLP	Multilayer Perceptron
DWT	Discrete Wavelet Transforms
BPNN	Back Propagation Neural Network
CSF	Cerebrospinal Fluid
WM	White Matter
GM	Gray Matter
EMMA	Ensembles of Multiple Models and Architectures
2D	Two Dimensional

3D	Three Dimensional
VAE	Variation Auto-Encoder
GB	Giga Bytes
WT	Whole Tumor
ET	Enhanced Tumor
TC	Tumor Core
PReLU	Parametric Rectilinear Function
DL	Deep learning
AFs	Activation Functions
Lr	Learning rate
SGD	Stochastic Gradient Descent
FP	False Positive
FN	False Negative
TP	True Positive

Abstract

This study proposes the usage of enhanced activation function to improve the performance of deep learning models used in MRI brain tumour segmentation. Activation function has a significant role in deep network stability, learning rate and accuracy of the resultant solution. The main advantage of this new activation function is the ability to provide more accurate results solving the problem of vanishing gradient in comparison to the common activation functions “ReLU”. Vanishing gradient affects the training rate and therefore, the weight updates and the overall network accuracy. This work aims to study the feasibility of increasing the accuracy of deep learning brain tumour segmentation using an enhanced activation function. In this study, U-Net deep learning model was chosen. A modified U-Net architecture was built for the segmentation task. Several enhanced activation functions were used besides the standard ReLU. The benchmark database used for the evaluation was BRAST 2015 dataset. The results showed that the HardELiSH activation function outperformed the standard ReLU activation function. This proves that the deep learning model performance in brain tumour segmentation can be enhanced with the choice of the activation function.

المستخلص

تقترح هذه الدراسة استخدام دالة التنشيط لتحسين أداء نماذج التعلم العميق المستخدمة في تشخيص ورم الدماغ بالرنين المغناطيسي. كما تلعب دالة التنشيط دورًا مهمًا في استقرار الشبكة العميقه ومعدل التعلم ودقة تحليل النتاتج.

الميزة الرئيسية لدالة التنشيط الجديدة هي القدرة على توفير نتائج أكثر دقة لحل مشكلة اختفاء التلاشي التدرجي اللانهائي مقارنة بدالة التنشيط الشائعه(ReLU).

أثبتت الدراسة تأثير التلاشي التدرجي اللانهائي على معدل التدريب وبالتالي يؤثر على تحديثات الوزن ودقة الشبكة الكلية. يهدف هذا العمل إلى دراسة جدوى U-Net المعدلة لمهمة زيادة دقة تشخيص ورم الدماغ بالتعلم العميق باستخدام دالة تنشيط محسنة. في هذه الدراسة ، تم اختيار نموذج U-Net بإستخدام العديد من دوال التنشيط المحسنة إلى جانب شائعة الاستخدام(ReLU). BRATS2015 إستخدامت كقاعدة بيانات في التقييم. أظهرت النتائج أن دالة التنشيط HardELiSH تفوقت على دالة التنشيط ReLU . هذه الدراسة أثبتت أنه يمكن تطوير أداء نموذج التعلم العميق في تشخيص ورم الدماغ بالإختيار الدقيق لدالة التنشيط.

CHAPTER ONE
INTRODUCTION

1.1 General view

Brain tumor is abnormal growth of cells in the brain. It is one of the most challenges disease nowadays in the world because it causes death to people suffering from it. Glial cells are the cause of gliomas that are the most common brain tumors. Gliomas are usually classified into low-grade gliomas (LGG) and high grade gliomas (HGG) which are malignant and more aggressive. According to the American Brain Tumor Association (ABTA). High-grade glioblastoma (HGGs) represent 74% of all malignant tumors and 26% of all primary brain tumors. World Health Organization (WHO) categorize HGGs as stage IV brain cancer [1]. So that, People who have such kind of this disease their predictions of survival in life is less than two years [2]. Such kinds of this diseases are difficult to treat if they are not diagnosed early in an appropriate and accurate way. Furthermore, the location of the tumor in the brain makes it so difficult to diagnose easily.

Magnetic Resonance Imaging (MRI) is a device used to scan different kinds of tumors in the brain, due to its excellent soft tissue contrast and capability for functional imaging it is used to locate the shape, size and location of the tumor in the brain. However, the diagnosing of the shape, size and location of the tumor in the early stage help to increase the patient life time[3]. The accurate delineation of tumor region in MRI sequences is of great importance since it allows volumetric measurement of the tumor, monitoring of tumor growth in the patient between multiple MRI scans and treatment planning with follow-up evaluation, including the prediction of overall survival[4].

There are different types of Brain tumors that are heterogeneous in appearance and shape which make MRI segmentation one of the most challenging tasks in medical image analysis. However, the segmentation for brain tumor are two types; either manual segmentation or automatical segmentation. Manual segmentation is not the appropriate method for diagnosis because it is time consuming and subjects to errors. Time consuming and less efficient results contribute to less accurate diagnoses. It has pointed inefficient results compared to automatic segmentations. Hence, the automatic segmentation is the best method to solve such problems faced by earlier methods.

With improvement and advancement in technology a new method has appeared recently which is automatically and perfectly detects the brain tumor, this method is called deep convolution neural network (DCNN). It has approved to be the best method to detect the brain tumor and most widely used nowadays. It is most powerful technique which outperforms traditional methods.

In previous years' neural networks have emerged in many applications in medical field which are used in diagnosing some diseases [5]. Neural networks have been successfully implemented in diagnosis, prediction, identify, classify, detect, and monitoring of cancer. Machine learning also as a field of neural networks has branched into different technique [6]. One of the most important techniques that has appeared in recent years is deep learning, which has proved to be the most powerful technique in detection and diagnosing tumors in the body. Specifically, the brain tumor detection.

Deep neural networks (DNNs) or deep learnings are the current area for research in many applications which are a part of broad field of the artificial intelligence. It is a class of neural networks characterized by a significant number of layers of neurons [7].

DNNs combines between science and engineering for creating intelligent machines that have powerful mechanisms to deal with machines like humans do. It works based on supervised and unsupervised learnings, the supervised learnings are used for classifications of cancer which are very effective method for diagnosis and treatment [7].

The BraTS challenge started on 2012 and since then, it has always been focusing on the evaluation of state-of-the-art methods for the segmentation of brain tumors in multimodal magnetic resonance imaging (MRI) scans [8]. The last iteration, BraTS 2018 utilizes multi-institutional pre-operative MRI scans and focused on the segmentation of intrinsically heterogeneous (in appearance, shape, and histology) brain tumors, namely gliomas [9]. However, BraTS challenges play an important role in increasing the diagnosis accuracy through the evaluation of dice score for the segmented tumor.

1.2 Problem Statement

Brain cancer is the most dangerous disease in the world. Hence, it is the most difficult part to diagnose and treat in the body. Therefore, we need an accurate and efficient diagnosing method for earlier treatment of the patients in order to safe their life.

The shape of brain tumour is unpredictable and its segmentation represent one of the most challenging tasks in medical image analysis.

Manual segmentations are time consuming compared with automatic segmentation.

Deep learning based methods are the top ranked in the task of MRI brain tumour segmentation. However, the problem of vanishing gradient influenced the overall accuracy due to the inappropriate selection of activation functions.

1.3 Objective

1.3.1 General objectives

To apply deep learning model for accurate brain tumour segmentation.

1.3.2 Specific objective

1. To implement of 2D data with small patches to lower the computations process. Therefore, increasing the accuracy of detection.
2. To apply new activation function 'HardELiSH' to avoid vanishing gradient problem.

1.4 Thesis layout

This thesis is constructed from five chapters. The first chapter consists of general overview, statement of the problem and aim of the study. The second chapter is focused on the brain tumour and its types and causes, furthermore, MRI structure, artificial neural network, background on deep learning and literature review. The third chapter proposes the methodology of the study. The forth chapter explains the results and discussions while the fifth chapter represents the conclusion and recommendations.

CHAPTER TWO
Background and Literature Review

2.1 Brain Tumor

Tumor is uncontrolled abnormal growth of cells. It harms the healthy tissues and starts to spread in different sides and parts in the body leading to destroy tissue functions that cause death with time if are not diagnosed and treated earlier. However, the development of cell is controlled by DNAs. DNAs contain sequences that control gene expression.

Tumors in the brain begin when normal cells acquire some abnormal signs in mutations in their DNA which allow the cells to grow and divide. Hence, normal cells start to dysfunction and die, so that, the result of this process is soft and spongy mass of tissues called brain tumors.

2.1.1 Types of brain tumors

There are two main types of brain tumor, Primary brain tumor and secondary brain tumors in which Primary brain tumor initializes in the brain [10] such locations; brain-covering membranes (meninges), cranial nerves, pituitary gland or pineal gland. Primary brain tumors are much less common than are secondary brain tumors, in which the secondary brain tumor begins elsewhere and spreads to the brain.

There are many different types of primary brain tumors such as acoustic neuroma (schwannoma), Astrocytoma, also known as glioma, which includes anaplastic astrocytoma and glioblastoma, ependymoma, ependymoblastoma, germ cell tumor, medulloblastoma, meningioma, neuroblastoma, oligodendroglioma, and pineoblastoma.

Secondary brain tumors (metastatic) are tumors that result from cancer that begins elsewhere in the body and then spreads (metastasizes) to the brain. They most often occur in people who have a history of cancer. The most common types of cancer that can spread to the brain are breast cancer, colon cancer, kidney cancer, lung cancer, and melanoma.

2.1.2 Stages of brain tumors

There are four stages for brain tumor, first stage occurs when tissues are composed of benign cells which are very similar to brain cells and grow slowly. While, second stage happens when the tissues are composed of malignant cells which contains more abnormal cells than cells in fist stage which is benign and has less abnormal cells. Hence, in the third

stage the tissues are composed of malignant cells and grow quickly. Finally, the fourth stage in which the tissues are malignant cells and grow more rapidly than third stage.

Brain tumor are two types, benign tumor and malignant tumor. Hence, benign tumor grows slowly and rarely spread to other parts in the brain but may cause some pressures on surrounding tissues. It may sometimes transform to malignant[10] and this process called malignant transformation.[11]. The most common types of benign brain tumors are: Meningioma's: these tumors arise from arachnoid matter and they are benign of stage one. These tumors represent around 34% of primary brain tumors, Acoustic Neuroma which is benign tumor of the nerve of hearing. It is located in the angle between cerebellum and the pons. Furthermore, chondroma which is also a benign tumor which arises in the base of the skull. Craniopharyngioma is rarely occurs which is benign tumor that forms above the pituitary gland. One more type is called Cysticastrocytomas, which are benign in the form of stage one, these are usually treated through surgical removal [12].

Coming to the malignant brain tumors, unlike benign tumors, malignant tumor is a type of tumor that spreads quickly which is very harmful and cause death and affects others parts in the brain and in the body if not treated early. Furthermore, it can come back after treated [13] (Lois et al. 2007). The most common types of malignant tumors are: High-grade astrocytomas, Oligodendrogliomas, Ependymomas, Glioblastoma, and Mixed gliomas.

2.1.3 Magnetic Resonance Imaging (MRI)

Magnetic resonance imaging (MRI) is a noninvasive technique which is used for diagnosing the diseases through producing details images about specific part inside the body. It uses strong magnetic fields and radio waves. It depends on the relaxation properties of protein nuclei in water and lipids. It has many functions such as, employs nonionizing radio frequency signals to obtain its images, detects various features in tissues via varying scanning parameters, creates cross sectional images besides oblique planes, and superior to detect and identify tumors [14].

Different types of MRI modalities used in segmenting and extracting tumour regions:

a) FLAIR images: Fluid-Attenuated Inversion-Recovery MRI: bright signal of the CSF (cerebrospinal fluid) is suppressed which allows a better detection of small hyper-intense lesions

b) T1-weighted MRI: image contrast is based predominantly on the T1 (longitudinal) relaxation time of tissue, tissue with short T1 relaxation time appears brighter (hyper intense).
c) T2-weighted MRI: image contrast is based predominantly on the T2 (transverse) relaxation time of tissue, tissue with long T2 relaxation time appears brighter (hyper intense).
d) T1-weighted MRI after administration of contrast media (T1c): many tumours show signal enhancement after administration of contrast agent.

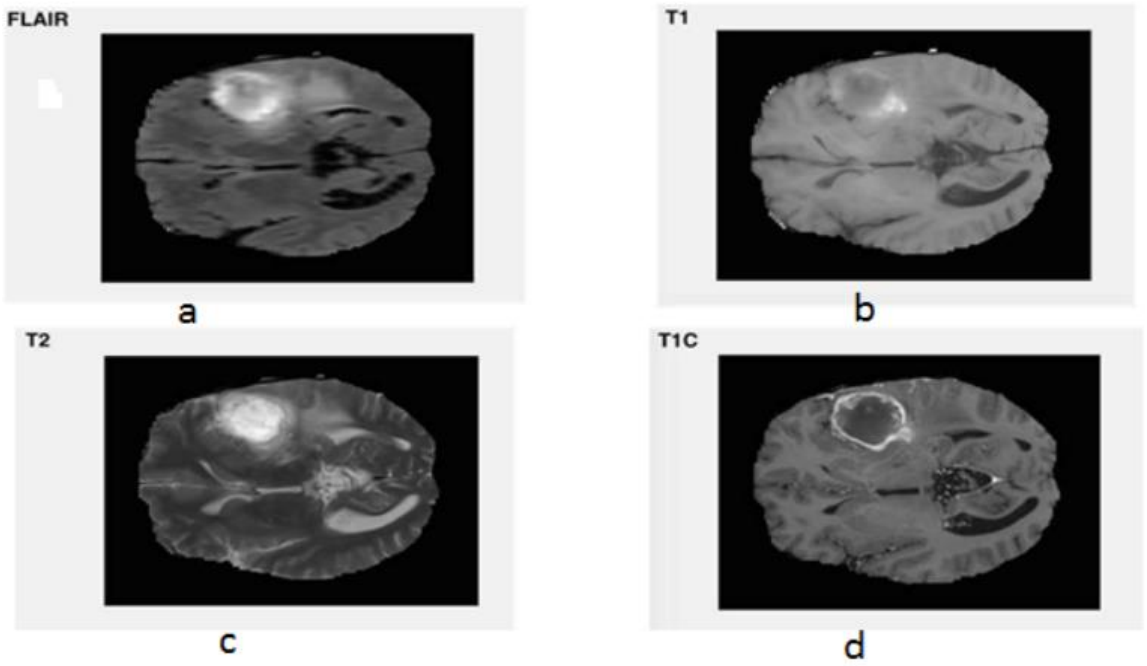


Figure 2. 1: Different modalities of MRI.

Usually, an expert radiologist uses MRI technique to generate a sequence of images (Flair, T1, T1 contrast, T2 ...etc.) to identify different regions of tumor. This variety of images helps radiologists and people working in this field to extract different types of information about the tumor (shape volume ...etc.) Fully Automatic Brain Tumor Segmentation using End-to-End Incremental Deep Neural Networks in MRI images, [15]**.**

MR images have multiple imaging file formats which describe how the data is organized in the image file and how these images should be interpreted for correct loading and

visualization. Some formats have a separate header file illustrating the data contained in the data file while others support the storage of the header and data in one file.

2.1.4 Machine Learning

Machine Learning is a field of study that gives computers the capability to learn without being explicitly programmed. It is one of the most exciting technologies that we have ever come across. It has the ability to learn the computer similar to humans. Machine learning is classified into three types: Supervised learning which works under the supervision of a teacher similar to human, unsupervised learning where this type learns from its own without any association response, and reinforcement learning which connected to applications for which the algorithm must make decisions.

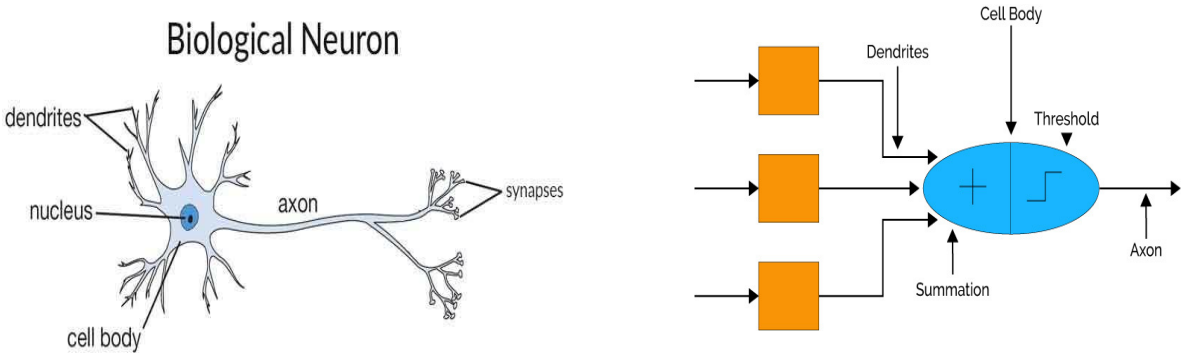
2.1.5 Artificial Neural Networks (ANNs) are a machine learning technique based on biological neural networks (BNN). It is also known as a "Neural Nets" which are a computational tool modeled through the interaction of neuron in the nervous system of the human brain and other organisms,[16].

Artificial Neural Networks, in general, is a biologically inspired network of artificial neurons applied on computer to perform specific tasks. The neurons are the basic processing element in the ANN. These neurons are not the same as the neurons in the human body but in terms of functionality, they work in the same manner. Hence, they are named as artificial neurons. They have a normal range of output between (-1, +1). These neurons could also be (0, 1) compute the sum of weighted inputs and then applied a non-linear transfer function to the computed sum[16].

Neural network is known as "Neural Net" which refers to both the biological and artificial networks. Mathematically, neural nets are nonlinear. It is a sequence of layer, each layer represents a non-linear combination of non-linear functions from the previous layer. Neurons in an ANN are arranged into layers. Each neuron is a multiple-input, multiple-output (MIMO) system that receives signals from the inputs, produces a resultant signal, and transmits that signal to all outputs.

Practically, the input layer is the first layer that interacts the environment and the final layer is called output layer. Layers between the input and the output layer are known as hidden layers. By increasing the number of layers the complexity of an ANN increases, and thus its

computational capacity requires the addition of more hidden layers, and more neurons per layer[16].



(a) **(b)**

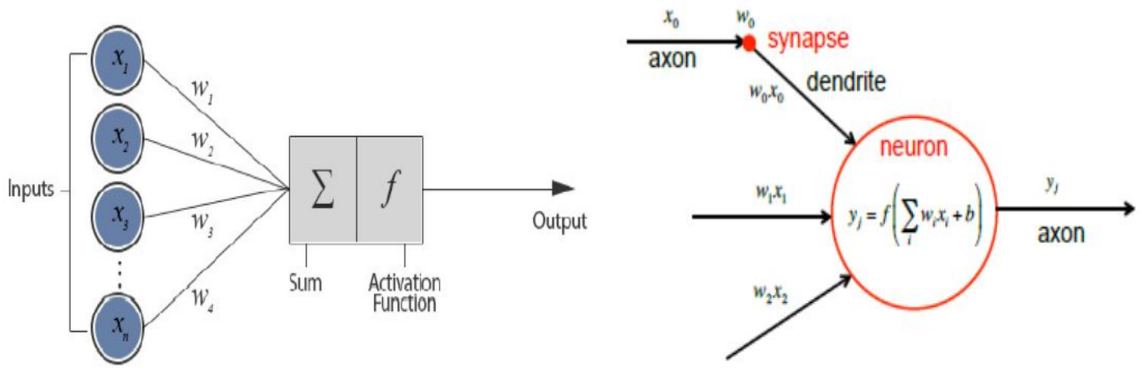
Figure 2. 2: Atypical diagram for biological neuron (a) and its application in ANN (b).

The dendrite receives signals from other neurons; soma (cell body) sums all the coming signals to generate input. Hence, axon transmits signals from one neuron to another. When the sums reach a threshold value, the neuron is fired and begins to transmit while synapses place of interaction of one neuron with others. The amount of signal transmitted depends on the synaptic weights.

The dendrites in the Biological Neural Network are analogous to the weighted inputs based on their synaptic interconnection in the Artificial Neural Network. The cell body is comparable to the artificial neuron unit in the Artificial Neural Network which also comprises of summation and threshold unit. Axon carries output that is analogous to the output unit in case of aartificial neural network. So, ANN is modelled using the working of basic biological neurons.

The similarity of Artificial Neural Networks with Biological Neural Network in which neural networks resemble the human brain in which neural network acquires knowledge through learning. Hence, the knowledge is stored within inter-neuron connection strengths known as synaptic weights.

Neural network archeticture:



(a) (b)

Figure 2. 3: (a) Simple neural network , (b) Connections to a neuron in the brain[7].

The inputs (x_1, x_2, …..x_i) and the weights (w_1, w_2, ...w_i) are real numbers and can be positive or negative. All the inputs are individually added together and passes into activation function [7].

Each neuron can be seen as a connection between a lower and a higher layer. However, the general structure is multiplying of the input by weight, adding a bias and executing an activation function. Thus, its mathematical operation can be written as:

$y_1 = f(\sum w_i * x_i + b)$…… (1)

where f: the activation function, w: the weights and b: the bias[7] .

As shown in the figure (2.3) ANN can be viewed as weights in which artificial neurons are nodes, between the neuron outputs and neuron inputs the edges with weights are connected. The ANN receives information inputs in the form of images in vector form. These inputs are mathematically designated by the notation x(n) for n number of inputs. Each input is multiplied by its corresponding weights. Weights are the information used by the neural network to solve a problem. Typically, the strength of the interconnection between neurons inside the Neural Network is represented by weight. The weighted inputs are all summed up inside computing unit (artificial neuron). In case the weighted sum is zero, bias is added to make the output non- zero or to scale up the system response. The input and weight for bias is always equal one.

Today, ANN represents a major extension to computation. It provides better results and performance than the traditional statistical tools for the prediction and classification purposes in various applications[17]. ANNs offer short computation times, low computational burden and the opportunity of reformulating the problem thereby

considering only on the important variables and parameters from the given data set or certain unknown areas of interest. There are different types of neural networks that are designed and developed for various applications[16].

2.1.6 Deep Learning

In recent years, deep artificial neural networks have won numerous contests in pattern recognition and machine learning[18]. It has recently become very popular and it is now successfully applied for a wide range of applications. However, as more complex and deeper networks are of interest, strategies are required to make neural network training more efficient and more stable.

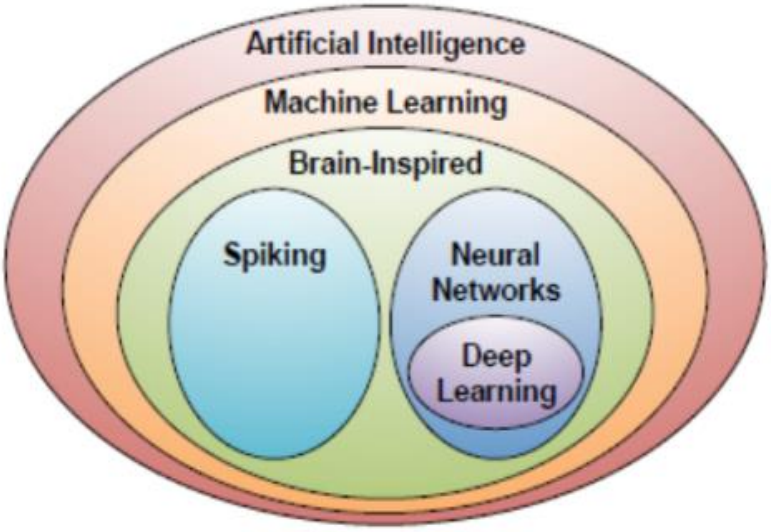


Figure 2. 4: Deep learning as a sub-branch of neural network

To learn representations of data, we use computational models which are composed with multiple processing. These methods have improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. deep learning has made a great revolution in identifying, classifying, and quantifying patterns in medical images [19].

Deep learning discovers complex structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional networks have made a huge impact in developing processing images, video,

speech and audio, whereas recurrent networks have shone light on sequential data such as text and speech [20].

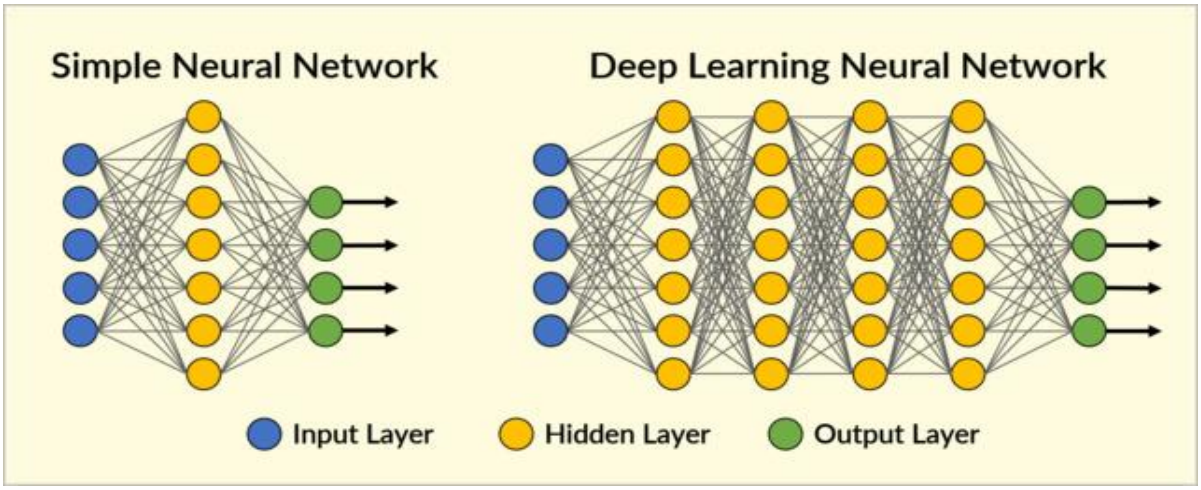


Figure 2. 5: Deep learning neural network structure [21].

Impressive improvements by deep learning, over other machine learning techniques in the literature, have been demonstrated. Those successes have been attractive enough to draw an attention of researchers in the field of computational medical imaging to investigate the potential of deep learning in medical images acquired with CT, MRI, PET, and X-ray, for example. [19].

Deep learning is a sub-branch of neural networks. It is characterized by a significant number of layers of neurons. Therfore, it is a super powerful technique that has appeared in recent years which works based on convolution neural network.

Compared with traditional methods, deep learning neural network has proved to be the most powerful method detects the tumor in a very automatic and accurate way. It has the ability to be trained with large database. Furthermore, it is fast in computations and gives results with high accuracy. Today deep learning is booming at a much higher rate than any before because of the following possible reasons; increasing dataset sizes, increasing model sizes, and increasing accuracy, complexity and real-world impact.

The main reasons behind the great success of deep learning over traditional machine learning models is the advancements in neural networks in which it learns high-level features from data in an incremental manner, which eliminates the need of domain expertise and hard feature extraction. And it solves the problem in an end to end manner. Furthermore, the appearance of GPU and GPU-computing libraries make the model can be

trained 10 to 30 times faster than on CPUs. And the open source software packages provide efficient GPU implementations. Finally, several available efficient optimization techniques also contribute the final success of deep learning, such as dropout, batch normalization, Adam optimizer and others, ReLU activation function and its variants, with that, we can update the weights and obtain the optimal performance[22].

2.1.7 Convolutional Neural Networks

CNNs are an outstanding branch of deep learning applications to visual purposes. Theses networks are made up of neurons with learnable weights and biases. Hence, CNNs work based on a feed-forward neural network and are widely used for image recognition and classification. Each neuron receives several inputs and takes a weighted sum over them, which passes them through activation function that has been selected to responds with output.

CNNs operate with volumes with inputs of multi-channelled images, unlike the neural networks which operate in vectors[4]. However, CNNs based models have proved their effectiveness and superiority over traditional medical image segmentation algorithms [10].

Due to the lack of labelled training data and computational power limitations (since 1990), it was not possible to train deep CNNs without over-fitting. As a result, proposals in this field were discontinued for some years. Therefore, traditional methods were used for segmentations. With time, graphics processing units (GPUs) have been created which are more powerful and capable for more training datasets. CNNs learn the relationships among the pixels of input images by extracting representative features using convolution and pooling operations.

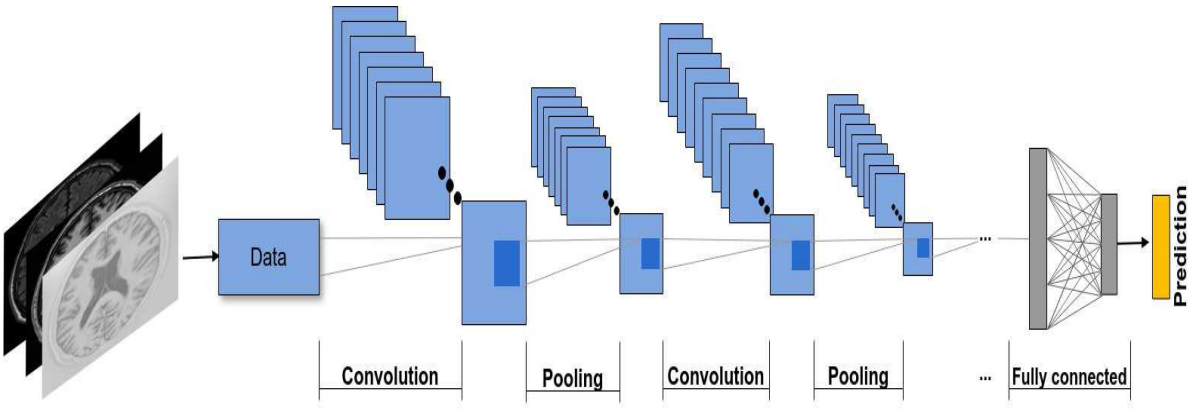


Figure 2. 6: CNN architecture [11].

From the figure (2.6) shown above, the output of each convolution operation at each layer is activated using activation functions before applying pooling operations.

The convolution operation produces different numbers of feature maps, depending on the numbers of filters are used. The pooling operations reduce the spatial dimensions of each feature map. After convolution and pooling the layers, the feature maps are flattened in the fully connected layer before a prediction is made using linear classifiers [11].

The neurons in the input layer receive some values and propagate them to the neurons in the middle layer of the network, which is also frequently called a 'hidden layer'.

(CNNs) Image segmentation with CNN involves feeding segments of an image as input to a convolutional neural network, which labels the pixels. The CNN cannot process the whole image at once. It scans the image, looking at a small "filter" of several pixels each time until it has mapped the entire image.

Advantages of using CNNs in brain tumor detection

- Automatic learning techniques
- Well-engineered algorithms
- Deal with high dimensional data
- Slight shift invariance is achieved using pooling layer
- Very deep CNN architectures replaced the conventional convolutional layer with more powerful representation while using less computational resources
- CNNs operate with volumes with inputs of multi-channelled images, unlike the neural networks which operate in vectors
- CNNs have proved their effectiveness and superiority over traditional medical image segmentation algorithms
- CNNs update their weights after calculating the error of each batch.

Fully Convolution Networks (FCNs) use convolutional layers to process varying input sizes and can work faster. The final output layer has a large receptive field and corresponds to the height and width of the image, while the number of channels corresponds to the number of classes. In order to determine the context of the image, including the location of objects, the convolutional layers classify every pixel.

Ensemble learning Synthesizes the results of two or more related analytical models into a single spread. We can increase the prediction accuracy and reduce generalization error by using ensemble learning. It enables accurate classification and segmentation of images through the generation a set of weak base-learners that classify parts of the image, and combine their output, instead of trying to create one single optimal learner.

Classification of ANN

Feedforward neural network

The feedforward neural network is type of artificial neural network. It works based on the flow of information in one direction from the input nodes through the hidden layers and to the output nodes. There is no loops in the network.

Single layer perceptron

It is another type of ANN which consists of a single layer of output nodes, the inputs are connected directly to the outputs by a series of weights. In each node the sum of the products of the weights and the inputs is calculated. If the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1).

Multi- layer percept ron(MLP)

MLP consists of multiple layers of computational units which interconnected in a feed-forward way. Each neuron is connected from one layer to the neurons of the subsequent layer. In this networks a sigmoid function appliesd as an activation function.

The most popular technique used in MLP is back propagation Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, in order to reduce the value of the error function by some small amount the algorithm adjusts the weights of each connection. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small [23].

Backpropagation neural network

Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).

The Backpropagation algorithm uses delta rule or gradient descent to look for the minimum value of the error function in weight space. The solution to the learning problem is the weights that minimize the error function.

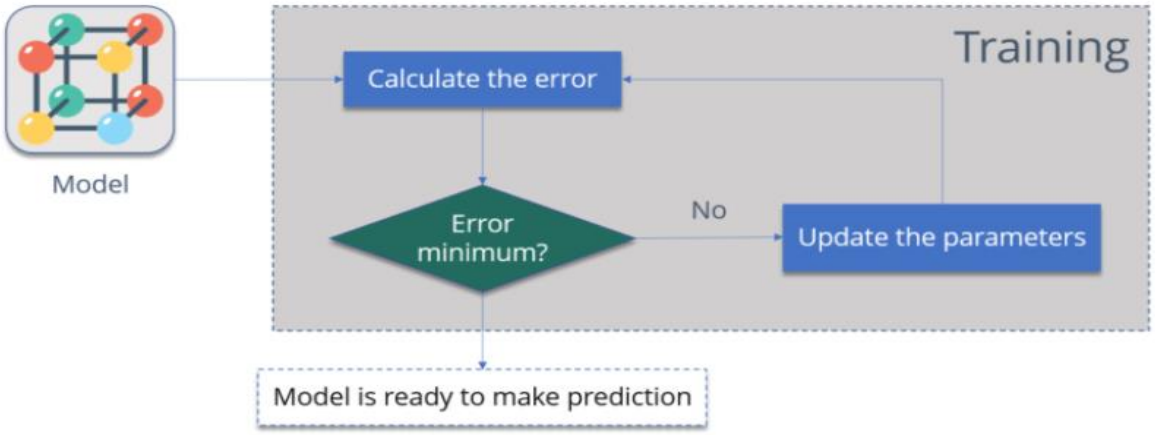


Figure 2. 7: Back propagation algorithm

From the above diagram we can summarize the following points: Calculate the error – How far is your model output from the actual output. Morever, minimum Error – Check whether the error is minimized or not. Furthermore, update the parameters – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum. Finally, model is ready to make a prediction – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.[24].

Rojas (2005) [25], proposed that the back propagation algorithm can be decomposed in the following four steps: (i) Feed-forward computation (ii) Back propagation to the output layer (iii) Back propagation to the hidden layer, and (iv) Weight updates. The algorithm is stopped when the value of the error function has become sufficiently small.

2.1.8 U-Net Architecture

U-Net is a Fully Convolutional Network (FCN) that segments medical images in a very better way. It predicts each pixel's class. The U-Net architecture is built upon the Fully Convolutional Network. Compared to FCN-8, U-net is symmetric and skip connections between the downsampling path and the upsampling path which instead of a sum apply a concatenation operator. These skip connections provide local information to the global information in the process of upsampling. It has a large number of feature maps in the upsampling path because of symmetry, which allows to transfer information.

U-Net architecture is separated in 3 parts: (1) The contracting/downsampling path, (2) Bottleneck, and (3) The expanding/upsampling path.

Contracting/downsampling path is composed of 4 blocks. Each block is composed of 3x3 Convolution Layer plus activation function (with batch normalization), 3x3 Convolution Layer plus activation function (with batch normalization), and 2x2 Max Pooling.

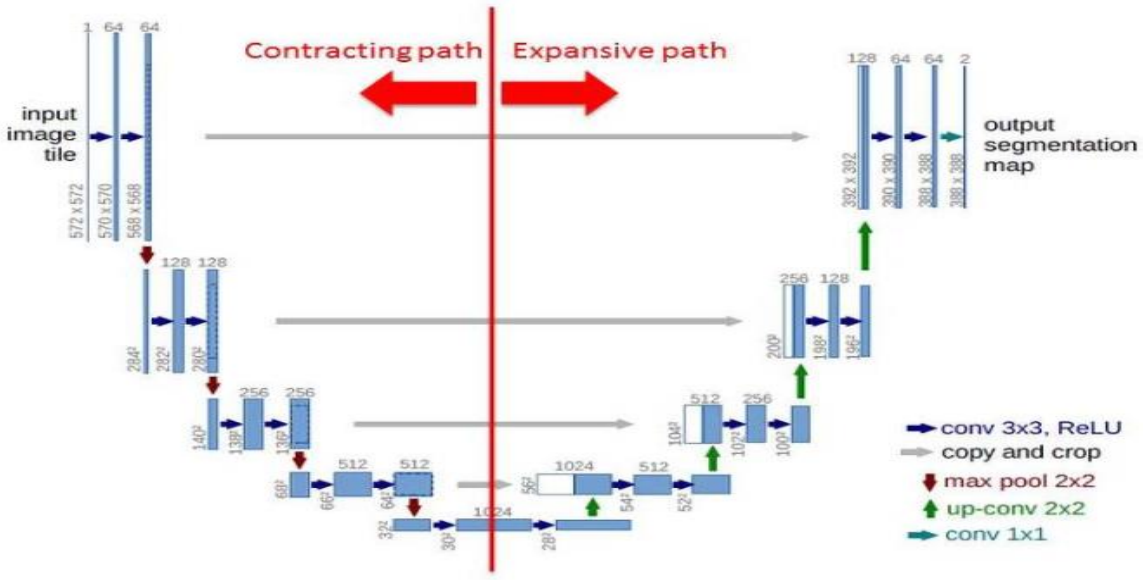


Figure 2. 8: U-Net architecture [26]

The number of feature maps doubles at each pooling, starting with 64 feature maps for the first block, 128 for the second, and so on. In order to be able to do segmentation, the contracting path capture the context of the input image. Then the information will then be transferred to the upsampling path by skip connections.

Bottleneck is a part of the network located between the contracting and expanding paths. The bottleneck is built from simply two convolutional layers (with batch normalization), with dropout. Expanding/upsampling path is composed of four blocks. Each of these blocks is composed of deconvolution layer with stride two, concatenation with the corresponding cropped feature map from the contracting path, 3x3 Convolution layer plus activation function (with batch normalization), and 3x3 Convolution layer plus activation function (with batch normalization). The purpose of this expanding path is to enable precise localization combined with contextual information from the contracting path [26].

Advantages of U-Net architecture

General information's are obtained by the U-Net through the combination of the location of information from the downsampling path with the contextual information in the upsampling path to finally obtain a combining localization and context, which is necessary to predict a good segmentation map. Furthermore, there is no dense layer, so that the images of different sizes can be used as input since the only parameters to learn on convolution layers are the kernel since the size of the kernel is independent from input image size.

2.2 Literature review

Image segmentation is most difficult task in locating and differentiating the normal and abnormal area of tumors while segmenting image. The manual brain tumor segmentation is time consuming and does not provide the result required. Hence, to overcome this problem we go for automatic detection and segmentation. In recent years, many methods are developed for automation of imaging, scanning, detection and segmentation.

Review studies based on arteficail neural network (deep learning)

The Multimodal Brain Tumour Segmentation (BraTS) challenge at MICCAI annual conference is an annual challenge since 2012. BraTS challenge is a competition that plays a significant role in the development of brain tumour segmentation methods. It can be observed that the top-ranked methods are based on deep learning methods.

Kamnitsas et al. (2017) the winners of the BraTS 2018 challenge, have proposed a study in Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation.

They proposed to ensemble several models for robust segmentation (EMMA). EMMA takes advantage of an ensemble of several independently trained architectures. In particular, EMMA combined DeepMedic, FCN and U-net models and ensembled their segmentation predictions. During training they used a batch size of 8, and a crop of 64x64x64 3D patch. EMMA's ensemble of different models demonstrated a good generalization performance winning the BraTS 2017 challenge.

The U-Nets were trained with input patches of size 64x64x64. The patches were sampled only from within the brain, with equal probability being centred around a voxel from each of the four labels. They were trained minimizing cross entropy via AdaDelta and Adam respectively, with different optimization, regularization and augmentation meta-parameters. The trained models are then applied fully convolutional on whole volumes for inference. All layers use batch normalisation, ReLUs and zero-padding. They used Tensorflow framework for implementation. The ensembles models are:

(a). DeepMedic process model

DeepMedic model is a fully 3D, multi-scale CNN, designed with a focus on efficient processing of 3D images. It employs parallel pathways, with the secondary taking as input down-sampled context. The first of the two models used is the residual version previously

employed in BRATS 2016. The second is a wider variant, with double the number of filters at each layer.

(b). Fully Connected Network (FCN) model

The second FNC is larger, replacing each convolutional layer with a residual block with two convolutions. The third is also residual-based, but with one less down-sampling step. All layers use batch normalisation, ReLUs and zero-padding.

(c). U-Net model

The U-Nets were trained with input patches of size 64×64×64. The patches were sampled only from within the brain, with equal probability being centred around a voxel from each of the four labels.

They were trained minimizing cross entropy via AdaDelta and Adam respectively, with different optimization, regularization and augmentation meta-parameters. The trained models are then applied fully convolutionally on whole volumes for inference. All layers use batch normalisation, ReLUs and zero-padding, [27].

Andriy (2018) the winner of the BraTS 2018 challenge, has proposed a study on 3D MRI brain tumor segmentation using autoencoder regularization, he described a semantic segmentation network for tumor subregion segmentation from 3D MRIs based on encoder-decoder architecture. Due to a limited training dataset size, a variational auto-encoder branch is added to reconstruct the input image itself in order to regularize the shared decoder and impose additional constraints on its layers.

This study follows the encoder-decoder structure of CNN, with asymmetrically large encoder to extract deep image features, and the decoder part reconstructs dense segmentation masks. The author's also added the variation auto-encoder (VAE) branch to the network to reconstruct the input images jointly with segmentation in order to regularize the shared encoder. At inference time, only the main segmentation encode-decoder part is used.

The author's used the largest crop size of 160x192x128 but compromise the batch size to be 1 to be able to fit network into the GPU memory limits. They also output all 3 nested tumor subregion directly after the sigmoid (instead of using several networks or the softmax over the number of classes). Finally, the author's added an additional branch to regularize the shared encoder, used only during training. No any additional training data

used and only the provided training set are used. The author's implemented their network in Tensorflow and trained it on NVIDIA Tesla V100 32GB GPU using BraTS 2018 training dataset (285 cases) without any additional in-house data.

Input is a four channel 3D MRI crop, followed by initial 3x3x3 3D convolution with 32 filters. Each green block is a ResNet-like block with the group normalization. The output of the segmentation decoder has three channels (with the same spatial size as the input) followed by a sigmoid for segmentation maps of the three tumor subregions (WT, TC, ET), [28].

Xue Feng et al. (2018) the winners of the BraTS 2018 challenge**,** developed an ensemble of 3D U-Nets for brain tumor segmentation. Zero padding was used to make sure the spatial dimension of the output is the same with the input. For each encoding block, a VGG like network with two consecutive 3D convolutional layers with kernel size 3 followed by the activation function and batch norm layers were used. The parametric rectilinear function (PReLU) was used with trainable parameter α as the activaton function. The number of features was doubled while the spatial dimension was halved with every encoding block, as in conventional U-Net structure. To improve the expressiveness of the network, a large number of features were used in the first encoding block. Dropout with ratio 0.5 was added after the last encoding block. Symmetric decoding blocks were used with skipconnections from corresponding encoding blocks. Features were concatenated to the de-convolution outputs. The extracted segmentation map of the input patch was expanded to the multi-class the ground truth labels (3 foreground classes and the background). Weighted/non-weighted cross entropy was used as the loss function, [29].

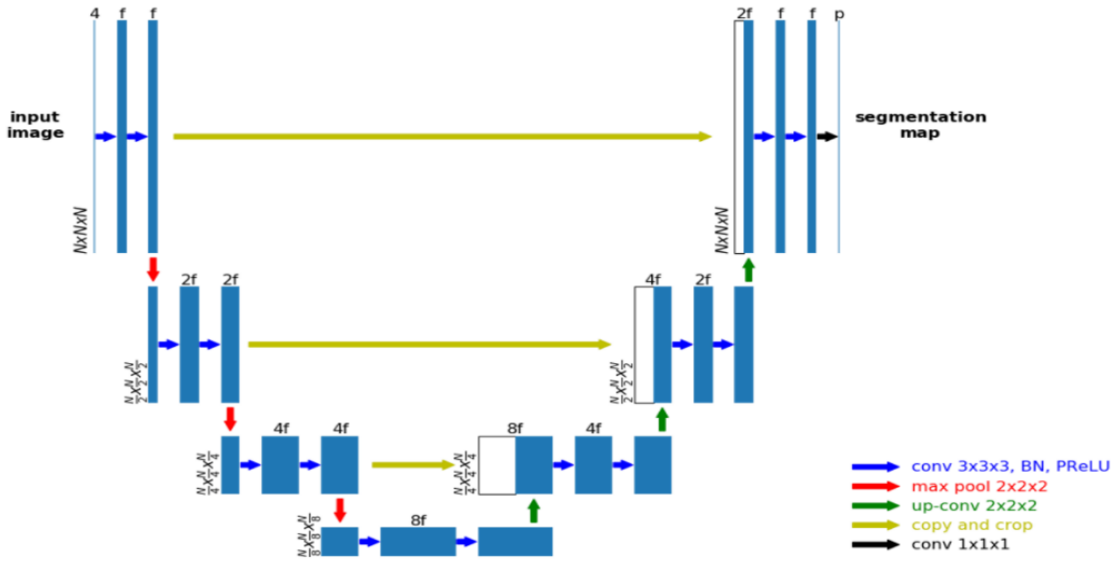


Figure 2. 9: 3D U-Net structure with 3 encoding and 3 decoding blocks.

Hao Dong et al (2019), presented a study a fully automatic brain tumor detection and segmentation method using the U-Net based deep convolution networks. Based on the experiments on a well-established benchmarking (BRATS 2015) datasets, which contain both HGG and LGG patients, they have demonstrated that their method can provide both efficient and robust segmentation compared to the manual delineated ground truth. In addition, compared to other state-of-the-art methods, their U-Net based deep convolution networks can also achieve comparable results for the complete tumor regions, and superior results for the core tumor regions. In their current study, the validation has been carried out using a five-fold cross-validation scheme; however, they can envisage a straightforward application on an independent testing datasets and further applications for multi-institutional and longitudinal datasets. The proposed method makes it possible to generate a patient-specific brain tumor segmentation model without manual interference, and this potentially enables objective lesion assessment for clinical tasks such as diagnosis, treatment planning and patient monitoring [1].

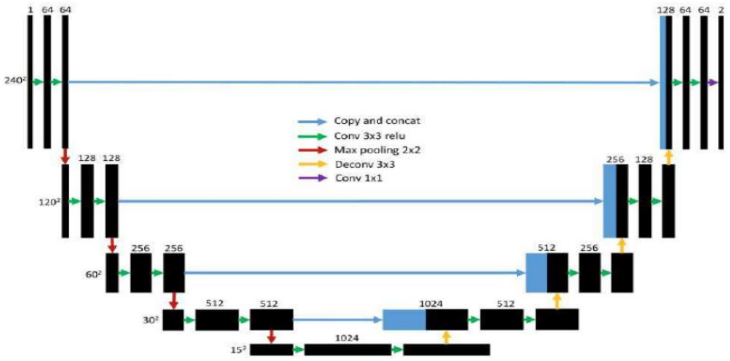


Figure 2. 10: The developed U-Net architecture

Adel Kermi, et al, (2019) presented a fully automated and efficient brain tumor segmentation method based on 2D Deep Convolutional Neural Networks (DNNs) which automatically extracts the whole tumor and intra-tumor regions, including enhancing tumor, edema and necrosis, from pre-operative multimodal 3D-MRI. The network architecture was inspired by U-net and has been modified to increase brain tumor segmentation performance.

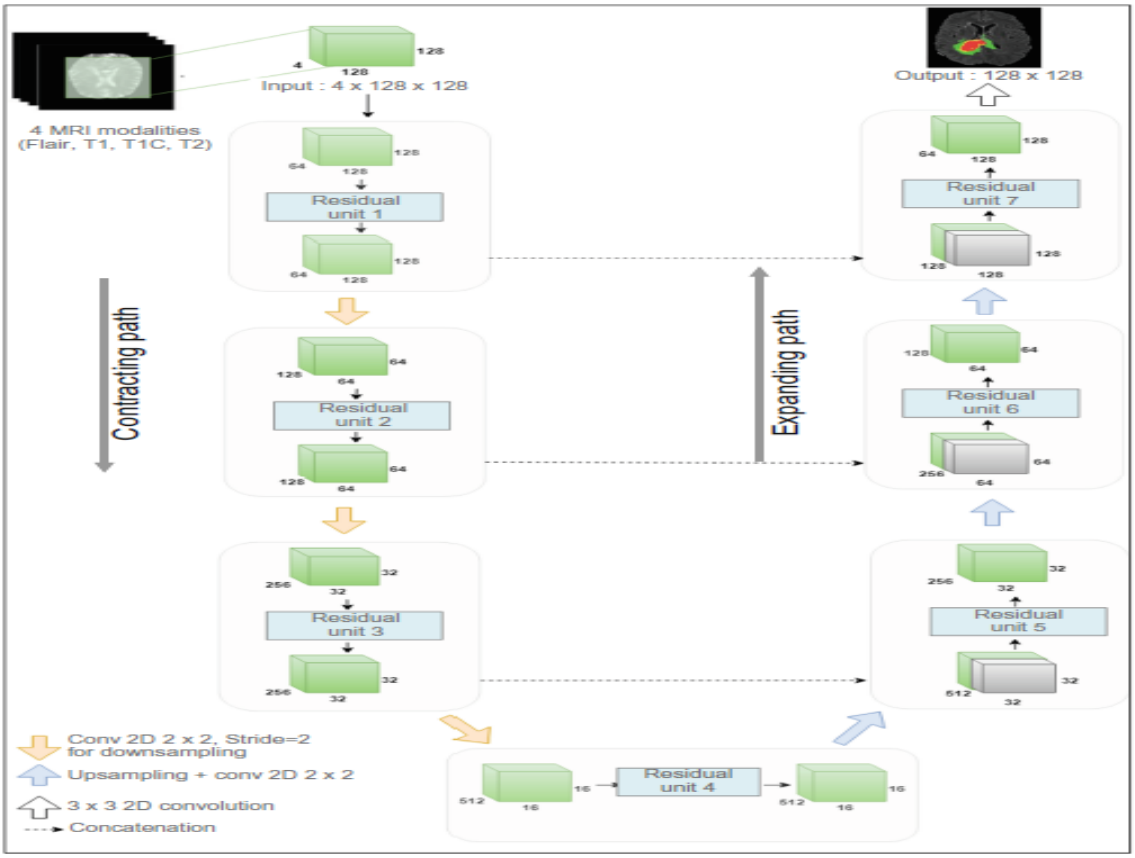


Figure 2. 11: Architecture of the proposed Deep Convolutional Neural Network

The proposed method was tested and evaluated quantitatively on both BraTS'2018 training and challenge validation datasets. Therefore, the total learning computation time of the 285 multimodal MRI volumes of BraTS'2018 training dataset is 185 h on a Cluster machine with Intel Xeon E5-2650 CPU@ 2.00 GHz (64 GB) and NVIDIA Quadro 4000–448 Core CUDA (2 GB) GPU. The average segmentation time of a brain tumor and its components from a given MRI volume is about 62 s on the same GPU [30].

CHAPTER THREE
Methodology

This chapter describes and discuss the methods are used for segmentation of brain tumor by using deep learning. Our contribution are based on BraTS challenges and previous studies in this field.

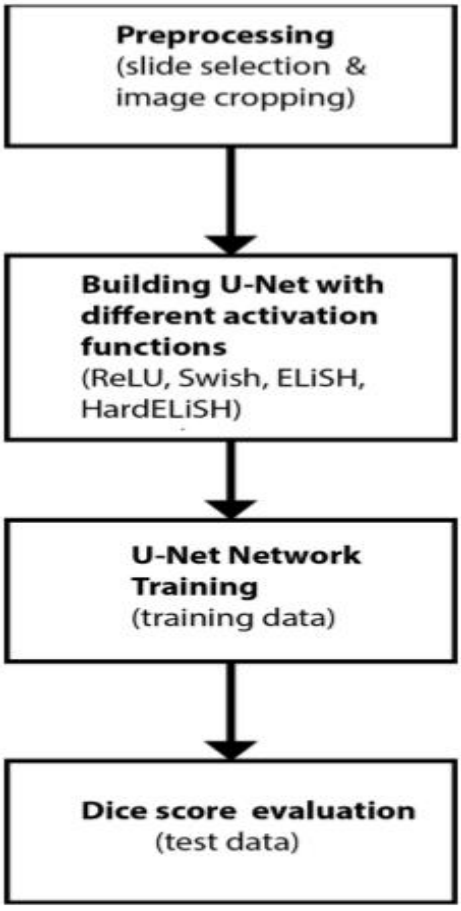


Figure 3. 1: Block diagram that shows the steps of detection of brain tumour by deep learning

3.1 Pre-processing:

1. Database, training and test datasets

This study used The Multimodal Brain Tumour Segmentation Tumour (BRATS 2015) as a benchmark database. The database is fully annotated patients data and it is composed of 220 HGG and 54 LGG each has four MRI modalities images (T1, T1c, T2, Flair). The brain tumour is classified by experts into the whole tumour (based on Flair), tumour core (based on T2), and enhanced tumour based on T1c as shown in

Figure. 3.1. In this study, only HGG patient's data were used. The data is divided into training and testing groups. The initial images dimension is 240×240×155, due to the limitation in computing power of the standard computer, only 20 images slices for each patient were selected and the size is reduced to 128×128. The ground-truth images were used as target images and the dice score as an error measure.

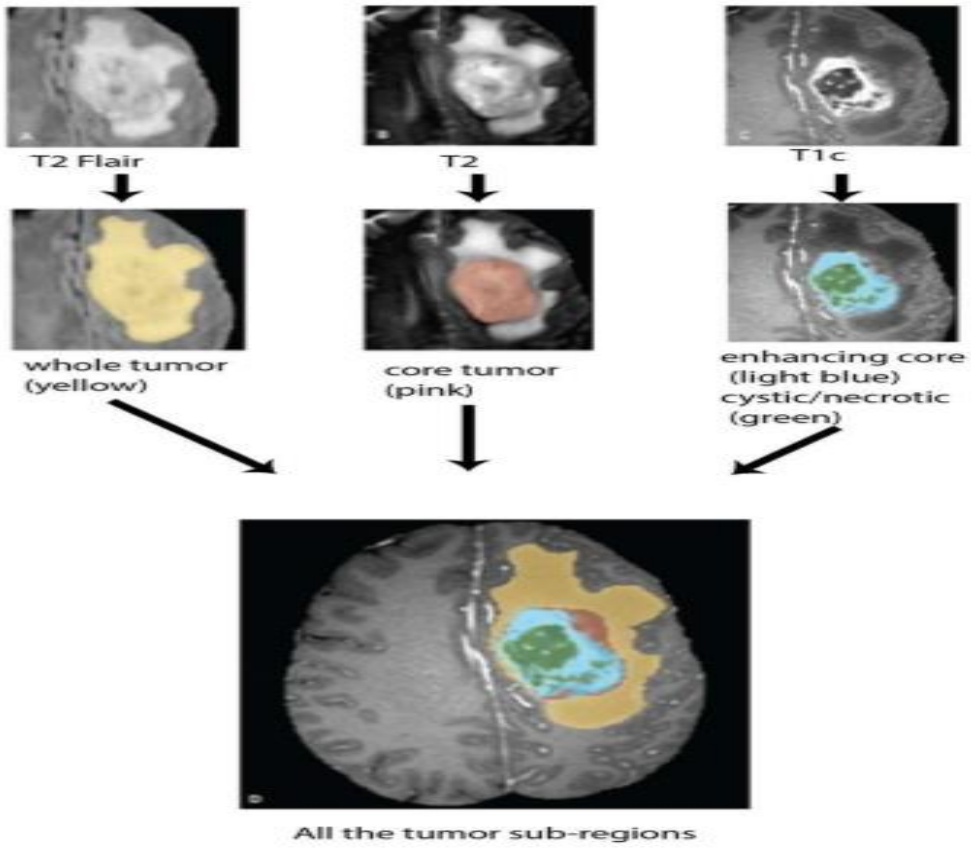


Figure 3. 2: The procedure of Glioma sub-regions segmentation by expert [8].

3.2 Hardware and software

We used advanced standard computer (CORE i5 7th Generation) to train the network with single GPU, 2GB memory, NVidia developer and Cuda toolkit 10.2.

The softwer used in this study is Python (version 3.7). The training and testing codes of the network are implemented by using PyTorch library (Spyder).

3.3 Training process

Before we can start training our model we need to configure the learning process. We need to specify an optimizer, a loss function and optionally some metrics like accuracy. The loss function is a measure on how good our model is at achieving the given objective. An optimizer is used to minimize the loss function by updating the weights using the gradients. The dataset is divided into training and testing data. Each activation function is trained with thirty epochs, where the epoch is defined as a complete pass over all the training samples. Hence, we used only HGGs for training with 5700 images per each activation function; each iteration took 190 images for training. We set the filters size to 3×3 in all our convolutional layers. We divided the patches into small patches to avoid overfitting. Moreover, padding is used as true so that the input shape is the same as the output while stride is used with one to modify the amount of movement over the image. The optimizer parameters used in this procedure as follow: Learning rate = 0.0001, momentum=0.99 and dropout =0.2. We visualized our training and testing accuracy and loss for each epoch so we can get information about the performance of our model. The accuracy and loss over epochs are saved in the history variable and we used Matplotlib to visualize this data.

3.4 U-Net deep learning Model

Figure 3.1 shows the block diagram of the method used of the evaluation of the enhanced activation function in the task of brain tumour segmentation.

This research performs brain tumor segmentation using 2-D U-Net which is a fast, efficient and simple network that has become popular in the semantic segmentation domain.

U-Net is a widespread deep network for segmentation. It has been the first model to be used for image segmentation in the medical field by Ronneberger, et al. in 2015 [26]. U-Net has multi-channel architecture, which suits well the multi-channel input of MRI images and the multi-class classification task. MR images has a high similarity and correlation feature in the intensities among neighboring voxels, so they are good for the convolution blocks constructing the U-Net. Essentially, U-Net is a deep-learning framework based on fully convolutional networks[31]. It comprises two parts:

1. A contracting path similar to an encoder, to capture context from a compact feature representation.

2. A symmetric expanding path similar to a decoder, which allows accurate localization . This step is done to retain boundary information (spatial information) despite down sampling and max-pooling performed in the encoder stage[19] .

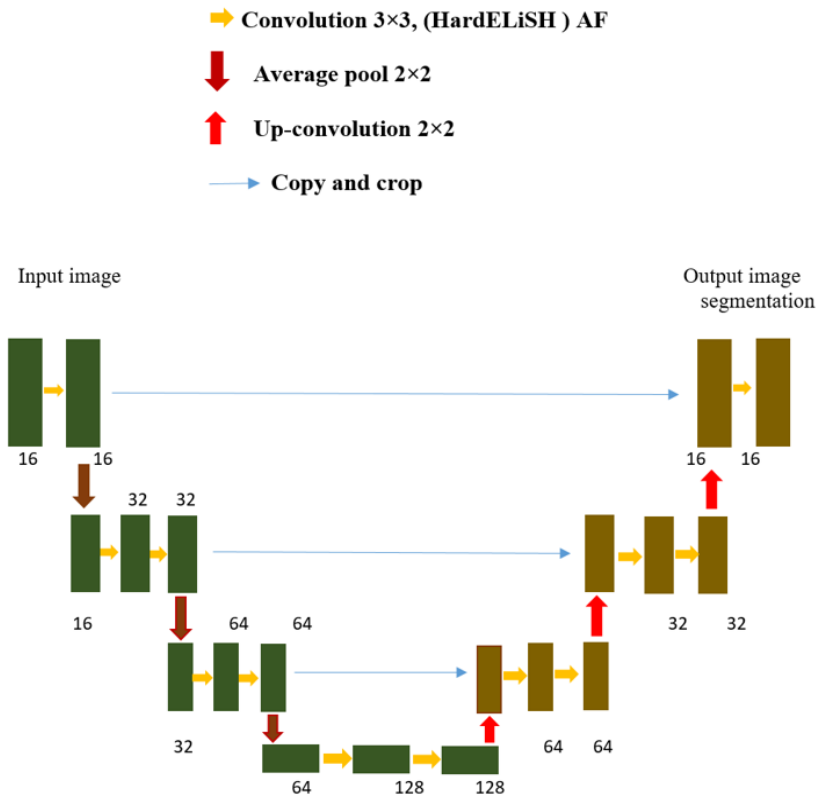


Figure 3. 3: Proposed U-net architecture model

A modified U-Net model is used in this study as shown in Figure (3.3)above. The architecture of the U-Net network consists of an encoder and decoder path. It is built from convolution layers. Each convolution layer is followed by batch normalization, average pooling and activation function. In general, it is composed of down-sampling and up-sampling paths. The down-sampling path is constructed from four CNN blocks with two layers for each block and filter size of 3 × 3, a stride of 1 in both directions and batch normalization. While the up-sampling path is built from deconvolution layer with 3 × 3 filter size and stride of 2 × 2. Hence, the feature maps are doubled and at the same time are decreased by two. A crop feature map is applied followed with two 3x3 convolutions, each

with activation function. In the last layer, a 1x1 convolution is applied to connect the 16-component features to the required number of classes that are foreground and background.

3.4.1 Activation Functions

The activation functions is used to compute the weighted sum of input and biases, which is used to decide if a neuron can be fired or not. It also manipulates the presented data through some gradient processing usually gradient descent and afterwards produce an output for the neural network, that contains the parameters in the data[32].

To achieve the state-of-the-art performances, the Deep learning (DL) architectures use activation functions (AFs), that perform diverse computations between the hidden layers and the output layers of any given DL architecture[33]. Many activation functions have been used in deep learning like ReLU, Swish, ELU, sigmoid …etc. However, the activation function is the heart of neural network, so that we focused on selecting the appropriate one in order to get solution to the vanishing gradient problem and to increase the accuracy of detection.

Due to the universal approximation properties of the activations functions the research in this field was mostly concentrated on squashing functions such as Sigmiod and Tanh. However, training DNNs using such functions suffers from the vanishing gradient problem.

To overcome this problem, various non-squashing functions were introduced, where the most notable example is Rectified Linear Unit (ReLU). In particular, as the derivative of positive inputs in ReLU are one the gradient cannot vanish. Therefore, as all negative values are mapped to zero, there is not information flow in DNNs for negative values. This problem is known as dying ReLU. Hence, Swish, EliSH, and HardELiSH were introduced.

The Swish Activation function

Ramachandran et al. proposed the Swish activation function in 2017 [34] . It is derived from the sigmoid activation function and it uses the reinforcement learning based automatic search technique to compute the function. The properties of the swish function include smoothness, non-monotonic, bounded below and unbounded in the upper limits. The smoothness property makes the swish function produce better optimization and generalization results when used in training deep learning architectures, which is defined as follows:

$$f(x) = x \cdot sigmoid(x) = \frac{x}{1+e^{-x}} \quad \text{…….. (3.1)} \quad [34].$$

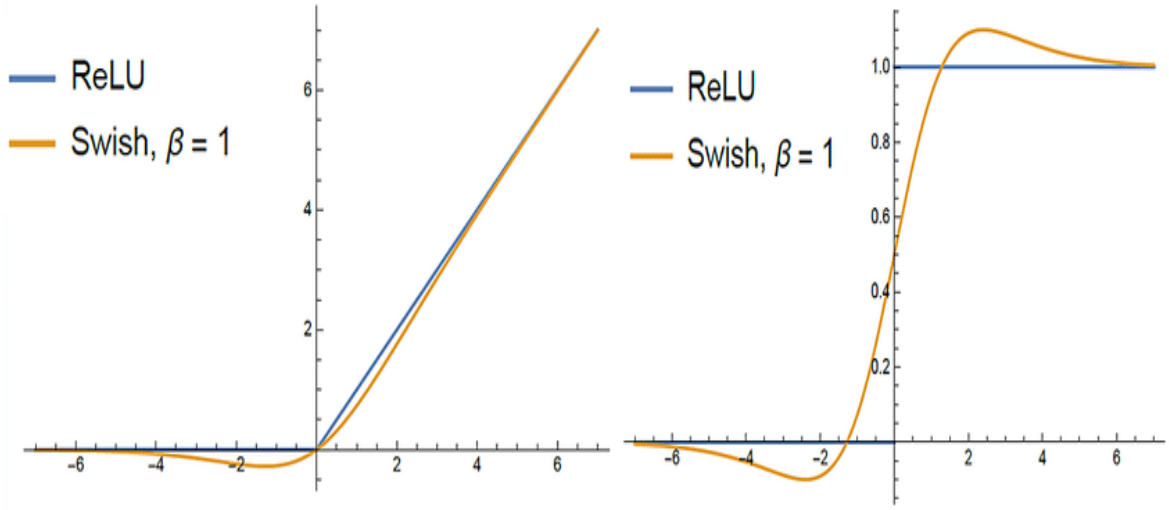


Figure 3. 4:(a) ReLU and Swish Functions (b)Derivative of ReLU and Swish

ReLU activation function is defined as:

$$f(x)=x^{+}=\max(0,x) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.2)$$

Where, x is the input to the neuron. ReLU has been the most popular activation function due to its gradient preserving property (it means that, it is having derivative of one for $x > 0$). The network extensively suffers from dying neuron problem for randomly initialized pre-activations where a high percentage of neurons are deactivated. Hence, deteriorating the network efficiency. ReLU is monotonous and smooth.

In 2018, Mina Basirat and Peter M. Roth further developed ELiSH and HardELiSH activation functions derived from the Swish activation function[35]. Equstions. 2 and 3 give the mathematical formulas of EliSH and HardEliSH activation functions respectively.

$$y(x)=\begin{cases} x/(1+e^{-x}), & x \geq 0 \\ (e^{x}-1)/(1+e^{-x}), & x < 0 \end{cases} \quad \ldots\ldots\ldots (3.3)$$

ELiSH activation function shares the properties of Swish, as its negative part is a multiplication of ELU and Sigmoid, while sharing the same positive part with Swish

$$y(x)=\begin{cases} x\max\left(0,\min\left(1,\frac{x+1}{2}\right)\right), & x \geq 0 \\ (e^{x}-1)\max\left(0,\min\left(1,\frac{x+1}{2}\right)\right), & x < 0 \end{cases} \quad \ldots\ldots\ldots (3.4)$$

HardELiSH activation function is introduced as a result of a multiplication of HardSigmoid and ELU in negative part and HardSigmoid and Linear in positive part.

HardELiSH and ELiSH activation functions provide a good flow of information during the training stage of the deep neural network. Therefore, they outperform the standard ReLU activation function dealing with the problem of the vanishing gradient [35].

Our contribution is based on using HardELiSH activation function in our project. It has shown to be competitive compared to existing approaches as well as very useful within the proposed framework. To demonstrate the benefits of our learned activation functions, we compared our approach with three activation functions and run it using U-Net network architectures. The results clearly demonstrate that using the proposed approach better results can be obtained.

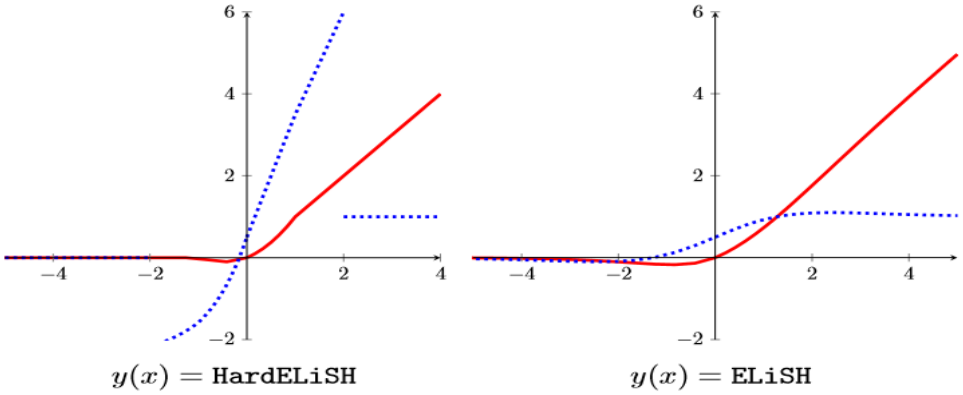


Figure 3. 5: The proposed new activation functions (red), HardELiSH and ELiSH, and their derivatives (blue dotted), respectively[35].

3.4.2 Optimization Algorithm

We use optimization algorithms to train the neural network through optimizing the cost function J. Given an algorithm f(x), an optimization algorithm helps in either minimizing or maximizing the value of f(x). The cost function can be defined as:

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y'^{i}, y^{i}\right) \quad \text{.....(3.5)}$$

The value of cost function J is the mean of the loss L between the predicted value *y'* and actual value y^i. During the forward propagation step we can obtain the value of y' by using the weights W and biases b of the network. The optimization algorithms minimize the value of cost function J by updating the values of the trainable parameters W and b.

3.4.3 Learning rate

Learning rate is a parameter of optimization technique. The jumping of the optimization technique between each iteration is specified by learning rate. The optimization will need to be run a lot of times if the learning rate is too small, (taking a long time and potentially never reaching the optimum). The optimization may be unstable (bouncing around the optimum, and maybe even getting worse rather than better) If the learning rate is too big.

3.4.4 Gradient Descent

Usually we use gradient descent to optimize weights and other parameters of neural networks. Differentiable programming languages allowing us to use gradient descent to optimize any program parameter that would otherwise be hard-coded by a human.

When we train a network, the weights (w_i) are usually updated using optimization process called gradient descent. A multiple of the gradient of the loss relative to each weight, which is the partial derivative of the loss with respect to the weight is used to update the weight. In gradient descent methods, the parameters update by using a back-propagation algorithm.

The Weight Matrix W is initialized randomly. We use gradient descent to minimize the cost function J and obtain the optimal Weight Matrix W and Bias b. Mathematically it can be defined as:

$$\begin{aligned} &\text{Repeat } \{ \\ &W = W - \alpha \frac{\partial}{\partial W} J(W) \\ &b = b - \alpha \frac{\partial}{\partial b} J(b) \\ &\} \end{aligned} \qquad \ldots\ldots (3.6)$$

The first equation represents the change in Weight Matrix W, whereas the second equation represents the change in Bias b. The learning rate alpha and the derivatives of the cost function J with respect to the Weight W and Bias b determines the change in values. We repeat the updating the weight and bias until the cost function J has been minimized [36].

3.4.5 Stochastic Gradient Descent

In order to update the weights, we accumulate the error across all training samples. There are big datasets, which lead to slow training over iteration for just one update.

The solution to this problem is the stochastic gradient descent (SGD) algorithm, which works in the same way as regular gradient descent, but updates the weights after every

training sample. However, SGD is prone to noise in the data. SGD is used as the optimizer in all training process.

It is the most commonly used algorithm for solving such optimization problems. This approach is implemented by passing a set of mini-batches through the network at a time and computing the gradient descent for each mini-batch. Gradient change Δw_i is propagated back to update the weights on all layers to the entire network [37].

3.4.6 Mini-Batch Gradient Descent

One of the disadvantages of gradient descent is that it begins the Parameter updating only after it goes through the full training data. This poses a challenge when the training data is too big to fit in the computer memory. Mini-Batch gradient descent is a powerful tool that tackles the above problems of Gradient Descent.

In Mini-Batch gradient descent, we distribute the whole training data in small mini-batches of sizes 16,32,64, and so on depending on the use case. We then use these mini-batches to train the network iteratively. The use of mini-batch has two advantages: (a) training starts as soon as we traverse over the first mini-batch, i.e., from the first few training examples, and (b) We can train a neural network even when we have a large amount of training data that doesn't fit in the memory. The batch size now becomes a new hyper parameter for our model.

When the batch size equals number of training examples, it is called as Batch gradient descent. It faces the problem of beginning learning only after traversing the whole dataset. When the batch size equals one, it is called as Stochastic Gradient Descent. It does not make full use of vectorization, and the training becomes very slow. Therefore, the common choice is 64 or 128 or 256 or 512. However, it depends on the use case and the system memory, i.e., we should ensure that a single mini-batch should be able to fit in the system memory[36] .

3.4.7 Pooling layer

It is inserted between successive convolution layers. The function of pooling is to progressively reduce the spatial size of the input representation. It also Makes the input representations (feature dimension) smaller and more manageable. Furthermore, it reduces the number of parameters and computations in the network, therefore, controlling overfitting. It makes the network invariant to small transformations, distortions and translations in the input image [38].

Pooling are two types: Max pooling, in max Pooling, we define a spatial neighbourhood and take the largest element from the rectified feature map within that window, and average pooling: In average pooling we take the average or sum of all elements in that window as shown in the figure (3.5) below.

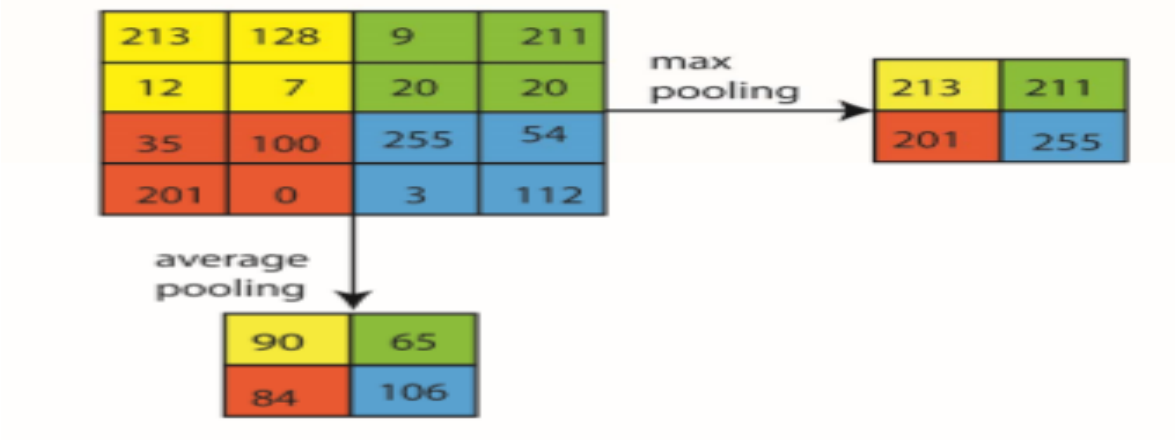


Figure 3. 6: types of pooling operations (max and average pooling operations).

Overfitting

It is one of the problems that occur during neural network training. It happens as a result of crush patterns in the training set that the network may wrongly assign to a specific class. It gets worse as the number of trainable weights increase in the neural network. Deep neural networks are susceptible to overfitting during training since they contain a large number of free parameter to be trained. This problem can be solved by using regularization to limit the parameters trained to capture only the features of interest in the image. Regulariztion uses dropout technique to solve the problem of overfitting. It drops some neurons during training phases to reduce the dependency of the final output on certain inputs.

3.4.8 Dropout technique

Hinton et al. [39] proposed termed named ”Dropout” which works based on randomly de-activate 50% of the nodes of a network on each training iteration. A disabled node would not participate in forward propagation (where they would output 0), and would block any error signal from propagating through the node during backpropagation. When training is done, all nodes are re-enabled, but all weights are halved to maintain the same output range [39]. Hence, we used dropout to avoid the overfitting problem.

3.4.9 Batch normalization

We used batch normalization in U-Net model to prevent overfitting problem. It is a super-powerful technique, which works between neural network hidden layers to ensure that at each layer the inputs are normalized, i.e. they are properly centred and scaled. As it is used to prevent overfitting problem it helps to speed up the training and hence, improve accuracy [37].

Training set is divided into batches. The CNN updates its weights after calculating the error of each batch. After each convolution layer the results of the convolution are normalized to have zero mean and standard deviation of one. The normalization is applied in over batches.

3.4.10 Loss function

In order to measure how well the model performs, the output is mapped to a single number. This loss function (error function) can be used to adjust the network to an desirable result. In image segmentations, a simple loss function is the binary categorical accuracy. It is defined as the percentage of the pixels. For medical segmentations, the large parts of the image which are belong to the background, the binary categorical accuracy can lead to false classification of the whole image to the background class. Therefore, a definition of the loss using the intersection and union of the segmentation with the ground truth is helpful. One loss function is the jaccard loss, also called jaccard distance, which is based on the jaccard index [40].

3.4.11 Momentum

In the optimisation technique we used momenteum to keep improving the model parameters towards the end of the optimisation process. It provides us with informations on how the parameters were changing over the last few iterations, and use these informations to keep moving in the same direction. The number of prior iterations changes depends on the initial learning stabilises.

3.5 Evaluation parameter

The evaluation parameter is the dice score, which is the standard parameter for the segmentation procedure. We used the following equation to obtain the dice score:

DSC = $\frac{2TP}{2TP+FP+FN}$ With TP: true positives, FP: false positives and FN: false negatives.

CHAPTER FOUR

Results and Discussions

4.1 Results

In this chapter we have shown and have discussed the results obtained by this proposed study.

Figure. 4.1 illustrates the results of HardEliSH brain tumour segmentation. The figure compares the ground truth (pink colour) with U-Net results (green colour). It can be noticed that the U-Net results are visually closed.

HardEliSH activation function:

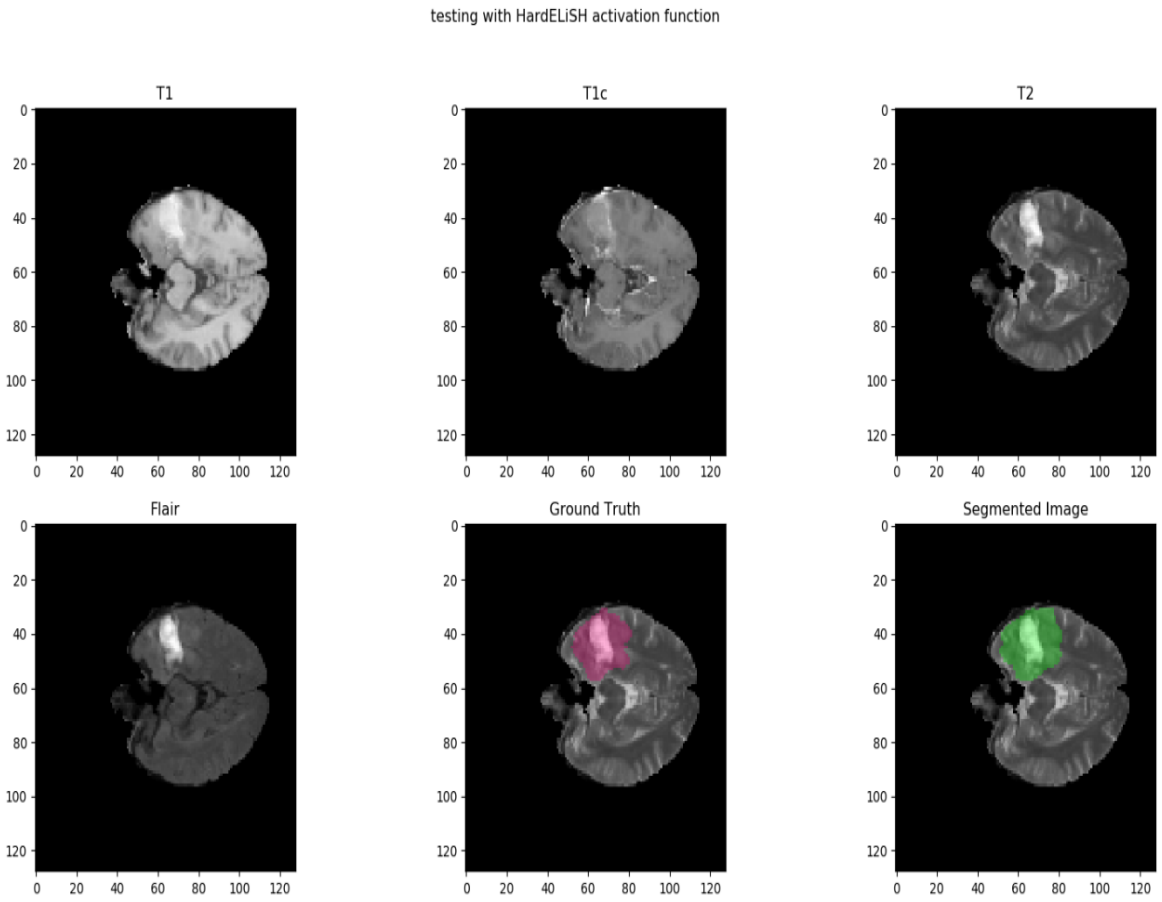


Figure 4. 1: Patient 2 brain tumour segmentation The results of U-Net using HardEliSH activation function. (a) T1. (b) T1c. (c) T2. (d) Flair. (e) Ground Truth (pin). (f) U-Net segmentation (green)

ReLU activation function result:

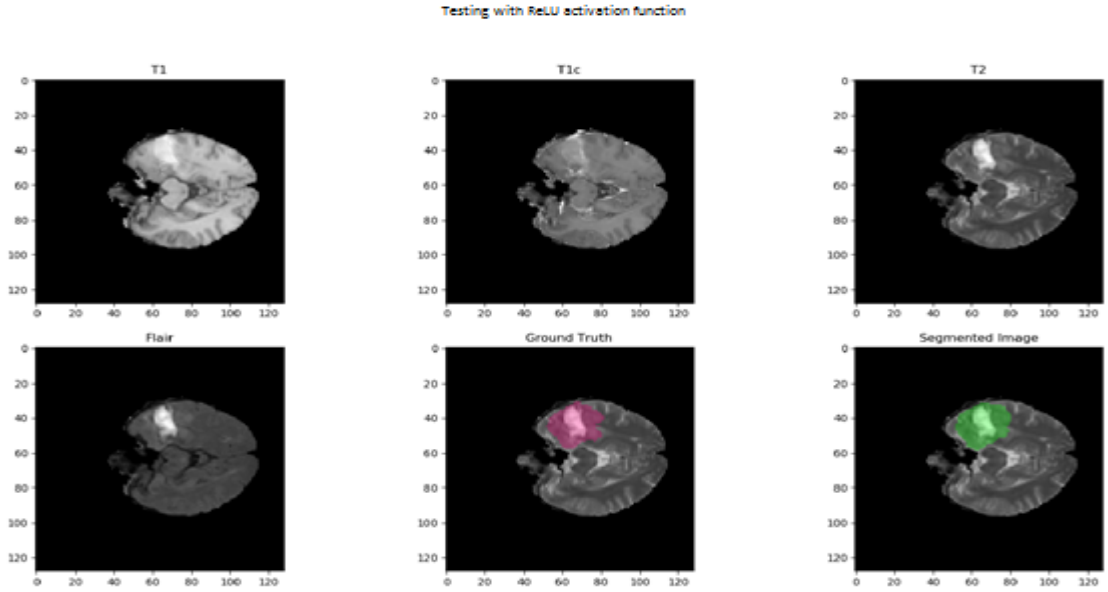


Figure 4. 2: Patient 2 brain tumour segmentation The results of U-Net using ReLU activation function. (a) T1. (b) T1c. (c) T2. (d) Flair. (e) Ground Truth (pin). (f) U-Net segmentation (green)

ELiSH activation function result:

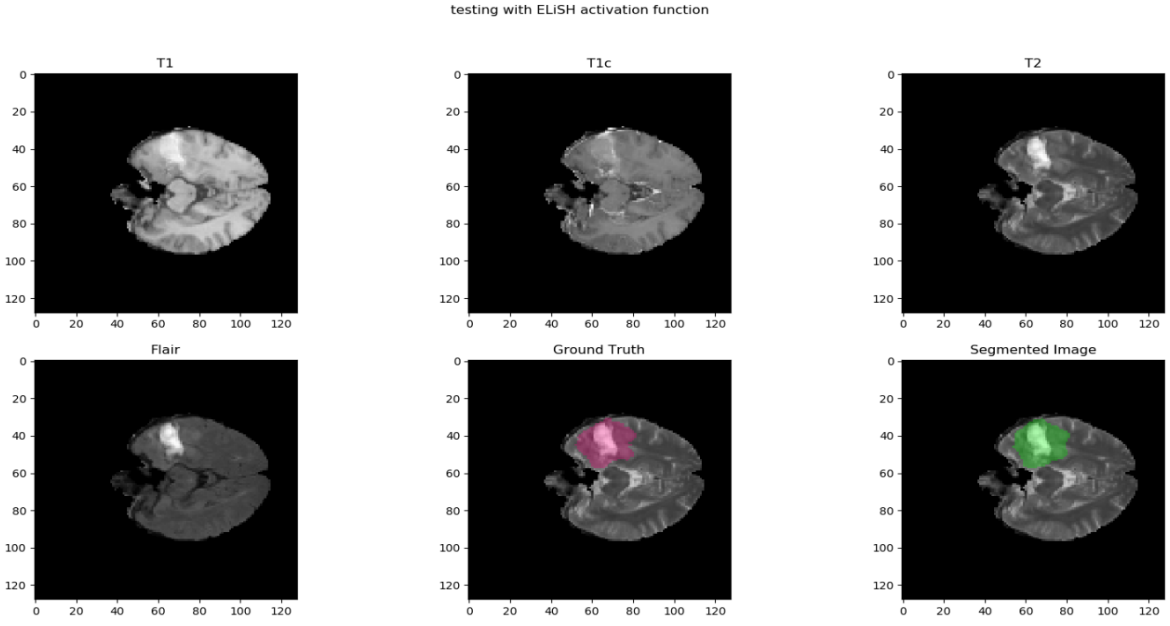


Figure 4. 3: The results of U-Net using EliSH activation function. (a) T1. (b) T1c. (c) T2. (d) Flair. (e) Ground Truth (pin). (f) U-Net segmentation (green).

Swish activation function result:

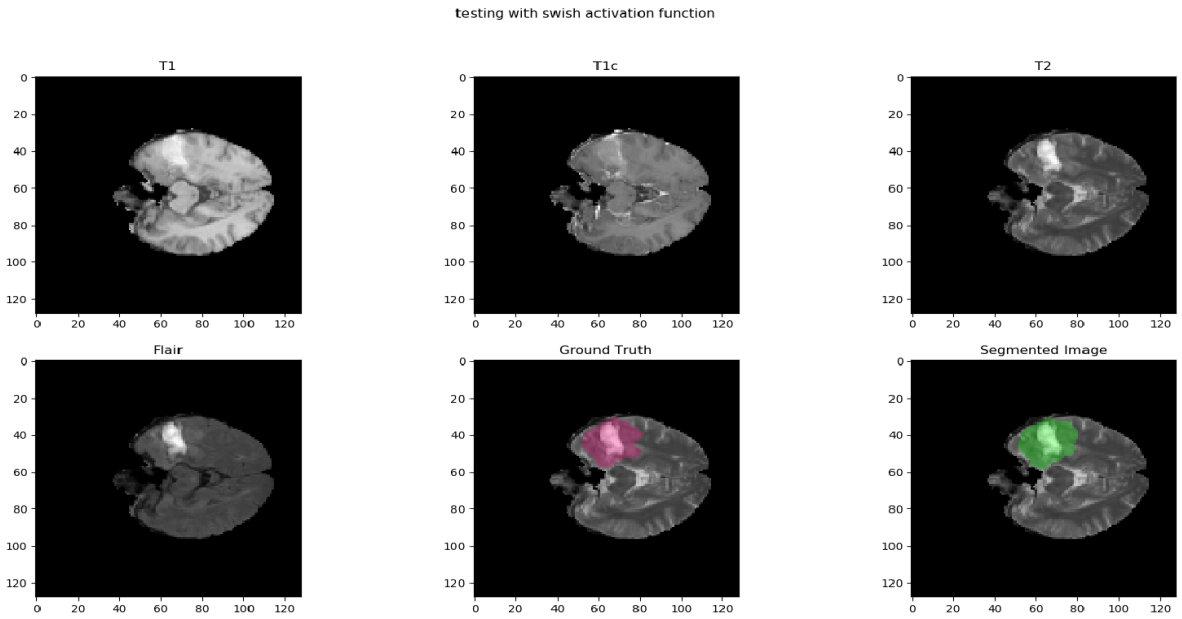


Figure 4. 4: The results of U-Net using swish activation function. (a) T1. (b) T1c. (c) T2. (d) Flair. (e) Ground Truth (pin). (f) U-Net segmentation (green).

Table 4.I compares the dice score resulting from the different activation functions. It can be observed that the new activation function HardELiSH obtained the best results and outperforms the most used ReLU in the task of MRI brain tumour segmentation.

Type of activation function	Number of epochs	Train dice score	Test dice score	Time Consumed
HardELiSH	**30**	**93.7%**	**88.4%**	6 hours
ReLU	**30**	**93.1%**	**88.2%**	6 hours and 8 minutes
ELiSH	**30**	**92.6%**	**87.8%**	6 hours and 13 minutes
Swish	**30**	**92.1%**	**88.8%**	7 hours

Table 4. 1 Results for training and testing images for different activation function showing that 'HardELiSH' activation function is the highest.

From the above table, we can say that the increasing number of epochs plays an important role in the increasing of the accuracy of the detection of brain tumour during the training. As we increase the number of epochs the accuracy increases. We started training with twenty epochs and we saw that the accuracy of both training dice score and testing dice score was good but when we increased the number of epochs to thirty, the accuracy increases. Hence, the accuracy depends on the increasing the number of epochs and the appropriate selection of activation function. We applied and measured different activation functions and compared between them as shown in the table 4.1. ReLU activation function is the common use among researchers in the field of brain tumour detection by deep learning. Hence, when we compared the new introduced activation functions HardELiSH with most standard one (ReLU) we found that, the HardELiSH activation function outperformed the ReLU. Therefore, the accuracy is increased. Furthermore, the time consumed in the training was less compared with ReLU.

4.2. Discussion

In this project we worked on the BraTS2015 data. The database was in the form of (Mha) files which we converted it to (png images). We implemented our network by using Pytorch software and trained it with standard computer with NVIDIA 2GB and single GPU. The data is a collection of several patients with low-grade gliomas (LGG) and High-grade gliomas (HGG). It is categorized by four modalities, T1, T1c, T2 and FLAIR MR images. .

The total number of images used for the training is 5700. However, there are 30 epochs each has 190 images. The training time of the U-Net network with each activation function in standard computer took around 6 hours (30 epochs). The optimizer, stochastic gradient descent, was implemented with the following parameters: learning rate = 0.0001, momentum = 0.99, and dropout = 0.2. The activation functions used were, ReLU, Swish, EliSH, and HardEliSH.

We used gradient descent to optimize weights and other parameters of neural networks. However, the optimization techniques often work based on learning rate. The learning rate specifies how aggressively the optimization technique should jump between each iteration.
We used the stochastic gradient descent (SGD) to solve the optimization problems which implemented by passing a set of mini-batches through the network at a time and computing the gradient descent for each mini-batch. We reduced the spatial size of the input representation by using pooling operation. Furthermore, reduced the number of parameters and computations in the network, therefore, controlling over fitting.

However, due to the large dataset in the training, the training will be slow. Hence, we have come over this problem by using batch normalization which works between layers through dividing the training set into batches, each batch size is 16. This makes CNN updates its weights after calculating the error of each batch.

Similarly, we used dropout as a regularization technique to solve the problem of overfitting. It works based on dropping some neurons during training phases to reduce the dependency of the final output on certain inputs. Adding momentum to the optimisation technique helped to keep improving the model parameters towards the end of the optimisation

process. It provides us with informations on how the parameters were changing over the last few iterations.

We have calculated the dice score of training and testing which is the standard parameter for the segmentation procedure for each activation function and compared between them. We have got that the HardELiSH activation function is the best one among the others for detection of brain tumour.

CHAPTER FIVE
Conclusion and Recommendation

5.1 Conclusion

This study aims to validate the feasibility of improvement of the performance of deep learning network using a suitable activation function. The findings of the study are promising and are expected to have a great impact on the task of brain tumour segmentation by increasing the performance of the existing already top-ranking deep learning models. Selecting inappropriate activation function leads to loss the information of the input during forward propagation and the exponential vanishing of gradients during back-propagation, this leads to inefficient and less accurate results. So that, our proposed study modified U-Net network to be trained and tested with different activation functions and compared between them. However, we have found that the Hard ELiSH activation function outperforms the others activation functions. Hence, we got 93.7% score in training and 88.4% in testing compared with most standard one (ReLU) activation function as shown in the table 4.1. Furthermore, it solves the vanishing gradient problem which faced by other activation functions. Hence, we have increased the accuracy of the brain tumour segmentation by the appropriate selection of activation function.

Limitations of the study

Due to the limitation in computation power and memory in our standard computer we couldn't increase the number of epochs. Moreover, we used only HGGs database for training. To use both LGGs and HGGs dataset and to increase number of epochs we require super-computing machine to train which is not available here in Sudan. These the limitations we faced during the work of this study.

5.2 Recommendations

a). Applying 3D data with HGGs and LGGs with 3D U-Net and training the network with using machine learning computers.

b). Develop a new framework for training models to detect brain tumors which will improve the speed or accuracy of detection and to be able to predict the future behavior of the patient tumor.

5.3 Published work

Mushtaq Salih, Musab Salih, and Mohammed Ahmed, Enhancement of U-Net Performance in MRI Brain Tumour Segmentation using HardELiSH Activation Function. International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) 2019. Accepted for publication in ICCCEEE 2019.

References

[1] H. Dong, G. Yang, F. Liu, Y. Mo, and Y. Guo, *Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks*, 2017.

[2] Z. Shboul, L. Vidyaratne, M. Alam, S. M. S. Reza, and K. M. Iftekharuddin, "Glioblastoma and Survival Prediction," *Brainlesion : glioma, multiple sclerosis, stroke and traumatic brain injuries : third International Workshop, BrainLes 2017, held in conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised selected papers. Bra...* vol. 10670, pp. 358-368, 2018.

[3] H. Dong, G. Yang, F. Liu, Y. Mo, and Y. Guo, "Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks," 2017.

[4] A. Albiol, A. Albiol, and F. Albiol, "Extending 2D Deep Learning Architectures to 3D Image Segmentation Problems," 2019.

[5] N. Shahid, T. Rappon, and W. Berta, "Applications of artificial neural networks in health care organizational decision-making: A scoping review," *PloS one,* vol. 14, pp. e0212356-e0212356, 2019.

[6] J. A. Cruz and D. S. Wishart, "Applications of machine learning in cancer prediction and prognosis," *Cancer informatics,* vol. 2, pp. 59-77, 2007.

[7] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *PROCEEDINGS- IEEE,* vol. 105, pp. 2295-2329, 2017.

[8] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby*, et al.*, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," *IEEE transactions on medical imaging,* vol. 34, pp. 1993-2024, 2015.

[9] M. Cabezas, S. Valverde, S. González-Villà, A. Clérigues, M. Salem, K. Kushibar*, et al.* (2018, October 01, 2018). Survival prediction using ensemble tumor segmentation and transfer learning. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv181004274C

[10] R. L. Buckner and D. C. Carroll, "Self-projection and the brain," *Trends Cogn Sci,* vol. 11, pp. 49-57, Feb 2007.

[11] U. Ilhan and A. Ilhan, "Brain tumor segmentation based on a new threshold approach," *Procedia Computer Science,* vol. 120, pp. 580-587, 01/01 2017.

[12] A. A. o. N. Surgeons, "Classification of Brain Tumors."

[13] O. Gonzalez-Perez and A. Quiñones-Hinojosa, "Astrocytes as neural stem cells in the adult brain," *Journal of stem cells,* vol. 7, pp. 181-188, 2012.

[14] J. Estelrich, M. J. Sánchez-Martín, and M. A. Busquets, "Nanoparticles in magnetic resonance imaging: from simple to dual contrast agents," *International journal of nanomedicine,* vol. 10, pp. 1727-1741, 2015.

[15] M. B. naceur, R. Saouli, M. Akil, and R. Kachouri, "Fully Automatic Brain Tumor Segmentation using End-To-End Incremental Deep Neural Networks in MRI images," *Computer Methods and Programs in Biomedicine,* vol. 166, pp. 39-49, 2018/11/01/ 2018.

[16] A. R. Omobolaji, "PREDICTION OF RECURRENCE AND MORTALITY OF ORAL TONGUE CANCER USING ARTIFICIAL NEURAL NETWORK," December 8, 2017 2017

[17] M. Paliwal and U. Kumar, "Kumar, U.A.: Neural networks and statistical techniques: A review of applications. Expert Systems with Applications 36(1), 2-17," *Expert Systems with Applications,* vol. 36, pp. 2-17, 01/31 2009.

[18] J. Schmidhuber. (2014, April 01, 2014). Deep Learning in Neural Networks: An Overview. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1404.7828S
[19] D. Shen, G. Wu, and H.-I. Suk, "Deep Learning in Medical Image Analysis," *Annual review of biomedical engineering,* vol. 19, pp. 221-248, 2017.
[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, pp. 436-44, May 28 2015.
[21] R. BERNARD, "Deep Learning to the Rescue," MARCH 25 2019.
[22] T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," *Array,* vol. 3-4, p. 100004, 2019/09/01/ 2019.
[23] Wikipedia, "Feedforward neural network."
[24] Saurabh, "Backpropagation – Algorithm For Training A Neural Network," May 22, 2019 2019.
[25] Ra\, \#250, and l. Rojas, *Neural networks: a systematic introduction*: Springer-Verlag, 1996.
[26] O. Ronneberger, P. Fischer, and T. Brox. (2015, May 01, 2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150504597R
[27] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski*, et al.* (2017, November 01, 2017). Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv171101468K
[28] A. Myronenko. (2018, October 01, 2018). 3D MRI brain tumor segmentation using autoencoder regularization. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv181011654M
[29] X. Feng, N. Tustison, and C. Meyer. (2018, December 01, 2018). Brain Tumor Segmentation using an Ensemble of 3D U-Nets and Overall Survival Prediction using Radiomic Features. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv181201049F
[30] A. Kermi, I. Mahmoudi, and M. T. Khadir, "Deep Convolutional Neural Networks Using U-Net for Automatic Brain Tumor Segmentation in Multimodal MRI Volumes," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Cham, 2019, pp. 37-48.
[31] J. Long, E. Shelhamer, and T. Darrell. (2014, November 01, 2014). Fully Convolutional Networks for Semantic Segmentation. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1411.4038L
[32] F. Manessi and A. Rozza. (2018, January 01, 2018). Learning Combinations of Activation Functions. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180109403M
[33] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. (2018, November 01, 2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv181103378N
[34] P. Ramachandran, B. Zoph, and Q. V. Le. (2017, October 01, 2017). Searching for Activation Functions. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv171005941R
[35] M. Basirat and P. M. Roth. (2018, August 01, 2018). The Quest for the Golden Activation Function. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180800783B
[36] R. Agrawal, "Optimization Algorithms for Deep Learning," Jule 23, 2019 2019.
[37] S. Ioffe and C. Szegedy. (2015, February 01, 2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150203167I

[38] M. M. Lopez, *Deep Learning for Brain Tumor Segmentation*: University of Colorado Colorado Springs, 2017.

[39] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (2012, July 01, 2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*. Available: https://ui.adsabs.harvard.edu/abs/2012arXiv1207.0580H

[40] M. Levandowsky and D. Winter, "Distance between Sets," *Nature,* vol. 234, pp. 34-35, 1971/11/01 1971.

APPENDIX

Implementation code for 2D U-Net Model

```
import torch
from torch import nn
from torch.nn import functional as F
class UNet(nn.Module):
    def __init__(self, in_channels=1, n_classes=2, depth=5, wf=6, padding=False,
    batch_norm=False, up_mode='upconv', use_HardELiSH = False):
    super(UNet, self).__init__()
     assert up_mode in ('upconv', 'upsample')
     self.padding = padding
      self.depth = depth
      prev_channels = in_channels
      self.down_path = nn.ModuleList()
     for i in range(depth):
self.down_path.append(UNetConvBlock( prev_channels, 2**(wf+i), padding, batch_norm, use_HardELiSH ))
      prev_channels = 2**(wf+i)
        self.up_path = nn.ModuleList()
        for i in reversed(range(depth - 1)):
  self.up_path.append(UNetUpBlock(prev_channels, 2**(wf+i), up_mode, padding, batch_norm, use_HardELiSH ))
  prev_channels = 2**(wf+i)
    self.last = nn.Conv2d(prev_channels, n_classes, kernel_size=1)
    def forward(self, x):
        blocks = []
        for i, down in enumerate(self.down_path):
            x = down(x)
            if i != len(self.down_path)-1:
                blocks.append(x)
                x = F.avg_pool2d(x, 2
```

```
 for i, up in enumerate(self.up_path):
            x = up(x, blocks[-i-1])
        return self.last(x)
class HardELiSH (nn.Module):
    def forward(self, x):
        return 1.67653251702 * x * torch.sigmoid(x)
class UNetConvBlock(nn.Module):
    def __init__(self, in_size, out_size, padding, batch_norm, use_swish):
        super(UNetConvBlock, self).__init__(   )
        block = []
        self.dropout = False
        block.append(nn.Conv2d(in_size, out_size, kernel_size=3,
                               padding=int(padding)))
        if use_ HardELiSH:
            block.append(HardELiSH ())
        else:
            block.append(nn.ReLU())
        if batch_norm:
            block.append(nn.BatchNorm2d(out_size))
        if self.dropout:
            block.append(nn.Dropout(p = 0.2))
        block.append(nn.Conv2d(out_size, out_size, kernel_size=3,
                               padding=int(padding)))
        if use_ HardELiSH:
            block.append(HardELiSH ())
        else:
            block.append(nn.ReLU())
        if batch_norm:
            block.append(nn.BatchNorm2d(out_size))
        if self.dropout:
            block.append(nn.Dropout(p = 0.2))
```

```
self.block = nn.Sequential(*block)
    def forward(self, x):
        out = self.block(x)
        return out
class UNetUpBlock(nn.Module):
    def __init__(self, in_size, out_size, up_mode, padding, batch_norm, use_ReLU):
        super(UNetUpBlock, self).__init__()
        if up_mode == 'upconv':
            self.up = nn.ConvTranspose2d(in_size, out_size, kernel_size=2,
                                         stride=2)
        elif up_mode == 'upsample':
            self.up = nn.Sequential(nn.Upsample(mode='bilinear', scale_factor=2),
                                    nn.Conv2d(in_size, out_size, kernel_size=1))
    self.conv_block = UNetConvBlock(in_size, out_size, padding, batch_norm, use_ReLU)
    def center_crop(self, layer, target_size):
        _, _, layer_height, layer_width = layer.size()
        diff_y = (layer_height - target_size[0]) // 2
        diff_x = (layer_width - target_size[1]) // 2
        return layer[:, :, diff_y:(diff_y + target_size[0]), diff_x:(diff_x + target_size[1])]
    def forward(self, x, bridge):
        up = self.up(x)
        crop1 = self.center_crop(bridge, up.shape[2:])
        out = torch.cat([up, crop1], 1)
        out = self.conv_block(out)
        return out
# model = UNet(depth = 5, wf = 3, padding = True, n_classes = 1, use_swish = True, batch_norm =
True)
# summary(model, (1,128,128)) import torch
from torch import nn
from torch.nn import functional as F
class UNet(nn.Module):
def __init__(self, in_channels=1, n_classes=2, depth=5, wf=6, padding=False,
```

```
    batch_norm=False, up_mode='upconv', use_HardELiSH = False):
     super(UNet, self).__init__()
     assert up_mode in ('upconv', 'upsample')
     self.padding = padding
     self.depth = depth
     prev_channels = in_channels
     self.down_path = nn.ModuleList()
     for i in range(depth):
     self.down_path.append(UNetConvBlock( prev_channels, 2**(wf+i),
      padding, batch_norm, use_HardELiSH ))
     prev_channels = 2**(wf+i)
     self.up_path = nn.ModuleList()
     for i in reversed(range(depth - 1)):
      self.up_path.append(UNetUpBlock(prev_channels, 2**(wf+i), up_mode,
       padding, batch_norm, use_HardELiSH ))
      prev_channels = 2**(wf+i)
     self.last = nn.Conv2d(prev_channels, n_classes, kernel_size=1)
      def forward(self, x):
     blocks = []
    for i, down in enumerate(self.down_path):
        x = down(x)
        if i != len(self.down_path)-1:
            blocks.append(x)
            x = F.avg_pool2d(x, 2)
 for i, up in enumerate(self.up_path):
x = up(x, blocks[-i-1])
     return self.last(x)
 class HardELiSH (nn.Module):
 def forward(self, x):
     return 1.67653251702 * x * torch.sigmoid(x)
 class UNetConvBlock(nn.Module):
```

```
    def __init__(self, in_size, out_size, padding, batch_norm, use_swish):
        super(UNetConvBlock, self).__init__()
        block = []
        self.dropout = False
        block.append(nn.Conv2d(in_size, out_size, kernel_size=3, padding=int(padding)))
        if use_swish:
            block.append(SWISH())
        else:
            block.append(nn.ReLU())
        if batch_norm:
            block.append(nn.BatchNorm2d(out_size))
        if self.dropout:
            block.append(nn.Dropout(p = 0.2))
        block.append(nn.Conv2d(out_size, out_size, kernel_size=3, padding=int(padding)))
        if use_ HardELiSH:
            block.append(SWISH())
        else:
            block.append(nn.ReLU())
        if batch_norm:
            block.append(nn.BatchNorm2d(out_size))
        if self.dropout:
            block.append(nn.Dropout(p = 0.2))
        self.block = nn.Sequential(*block)
    def forward(self, x):
        out = self.block(x)
        return out
class UNetUpBlock(nn.Module):
    def __init__(self, in_size, out_size, up_mode, padding, batch_norm, use_ReLU):
        super(UNetUpBlock, self).__init__()
        if up_mode == 'upconv':
            self.up = nn.ConvTranspose2d(in_size, out_size, kernel_size=2,   stride=2)
```

```
        elif up_mode == 'upsample':
            self.up = nn.Sequential(nn.Upsample(mode='bilinear', scale_factor=2), nn.Conv2d(in_size, out_size, kernel_size=1))
        self.conv_block = UNetConvBlock(in_size, out_size, padding, batch_norm, use_ReLU)

    def center_crop(self, layer, target_size):
        _, _, layer_height, layer_width = layer.size()
        diff_y = (layer_height - target_size[0]) // 2
        diff_x = (layer_width - target_size[1]) // 2
        return layer[:, :, diff_y:(diff_y + target_size[0]), diff_x:(diff_x + target_size[1])]

    def forward(self, x, bridge):
        up = self.up(x)
        crop1 = self.center_crop(bridge, up.shape[2:])
        out = torch.cat([up, crop1], 1)
        out = self.conv_block(out)
        return out

# model = UNet(depth = 5, wf = 3, padding = True, n_classes = 1, use_ HardELiSH = True, batch_norm = True)
# summary(model, (1,128,128))
```