CSC8499 Individual Project: Automated Brain Tumor Segmentation Using U-Net

Shubham Jakhmola

MSc in Advanced Computer Science,
School of Computing Science, Newcastle University.
s.jakhmola2@ncl.ac.uk

Abstract. Gliomas are the most prevalent and fatal kind of brain tumor, with an extremely low life expectancy in their most severe forms. Thus, a crucial step in enhancing the quality of life for cancer patients is treatment planning. Although magnetic resonance imaging (MRI) is a popular imaging method for evaluating these tumors, the volume of data it generates makes it difficult to manually segment the images in a reasonable amount of time, which restricts the use of precise quantitative assessments in clinical settings. The enormous spatial and structural heterogeneity among brain tumors makes automatic segmentation a difficult task, hence dependable and automatic segmentation methods are needed. This project focuses on developing deep learning models based on convolutional neural network to perform the automated semantic image segmentation of the MR images of brain. We explore the current state of the art autoencoder style U-Net architecture, also we use other prominent CNNs such as ResNet and VGG as a backbone in the U-Net architecture and evaluate them on the BraTS dataset. Different regularisation methods and hyperparameters are tested and optimised through a series of experiments. Finally, a web application is created so that the developed models can be used easily by medical practitioners.

Declaration: I declare that this dissertation represents my own work except where otherwise explicitly stated.

1 Introduction

A brain tumor is a development of abnormal cells in the brain that multiply uncontrollably. Since the human skull is a rigid and volume-limited structure, any unanticipated development may have an impact on a human function depending on the area of the brain involved. It also has the potential to spread to other bodily organs and have an impact on human functions [1]. Brain tumor can be benign (non-cancerous) or malignant (cancerous). As stated in [2], brain and other nervous system cancer is the tenth largest cause of death, with a five-year survival rate of 34% for males and 36% for women for those with cancer of the brain. The most common type of brain tumor

found in adults is glioma which starts from the glial cells [3]. According to WHO these tumors are categorized in 4 types ranging from I to IV in terms of severity [4]. Types III and IV gliomas are high-grade gliomas that nearly always result in death, whereas Types II and I low-grade gliomas grow more slowly and have a longer life expectancy, hence vigorous treatment is frequently postponed as long as possible. As a result, earlier diagnosis of brain tumors can substantially improve options for treatment and increase the likelihood of survival.

Image segmentation is the process of partitioning an image into well-defined regions or categories, each of which comprises pixels with comparable qualities and is designated to one of these categories [5]. Similarly, brain tumor segmentation is the process of separating the tumorous from the non-tumorous regions of the brain [6]. MRIs are the standard technique for brain tumor diagnosis as it is non-invasive and provides good soft tissue contrast with high spatial resolution [7]. Improved disease diagnosis, treatment planning, monitoring, and clinical trials all depend on the segmentation of brain tumors from neuroimaging modalities. To determine the location and size of the tumor, accurate brain tumor segmentation is necessary. However, the characteristics of brain tumors make accurate segmentation challenging. These tumors can develop in practically any area and come in a wide range of sizes and shapes. The intensity value of a tumor may overlap with the intensity value of healthy brain tissue, and they are typically poorly contrasted. As a result, it is difficult to tell healthy tissue from a tumor. Integrating data from various MR modalities, such as T1-weighted MRI (T1), T1-weighted MRI with contrast (T1c), and T2-weighted MRI, is a typical method to address this problem.

Depending on the degree of human interaction during segmentation of the scans, MRI segmentation can be divided into three classes. It can be classified into semi-automated approaches [8], completely automatic methods [9], and manual methods.

Precision and speed in treatment planning are critical for enhancing patient quality of life, however manual segmentation is time-consuming due to the enormous amount of data provided by MRI. As a result, approaches for automatic and reliable segmentation are necessary. However, developing automated brain tumor segmentation techniques is technically challenging and even professional raters' manual segmentations exhibit intra-operator variability [10] as tumors can be ill-defined with soft tissue boundaries and lesions deform surrounding normal tissues. Furthermore, the lack of a widely available brain tumor database containing ground-truth segmentations makes it difficult to assess the performance of different techniques in an unbiased manner. The BraTS (Brain Tumor Segmentation challenge), however, has made a concerted effort in this regard [11].

Artificial neural networks (ANN) and, in particular, convolutional neural networks have been shown to outclass humans in the task of image segmentation and classification. as illustrated by the classification of melanomas [12]. In 2015, Ronneberger et al. published U-Net [13], a network for segmenting biomedical images. This method outperformed all competing network structures in the ISBI challenge, achieving exceptional results. Since then, the U-Net architecture has been implemented in a variety of fields for segmentation, including the segmentation of brain tumors, where it has demonstrated increasing performance each year in the BraTS challenge [11,14,15].

The purpose of this project is to develop an automated brain tumor segmentation application, that uses multimodal MR images of patient's brain to generate the segmentation mask. The dataset employed for segmentation is BRATS'20, which comprises four distinct MRI modalities and one target mask file. The intention is to build on top of current state of the art, analyze and evaluate different medical image segmentation techniques, creating easy to use GUI for medical practitioners to perform automatic segmentation of gliomas.

1.1 Aim and Objectives

Aim: This dissertation aims to develop and implement an Artificial Neural Network that performs automatic brain tumor segmentation. In order to accomplish this goal, the basic structural components of the U-Net structure will be deciphered and subsequently implemented**.** Experiments will be conducted to evaluate the performance of various network architectures. Finally, we create a web application in which the developed models can be used efficiently.

Objectives:

- Research different brain tumor segments and the MRI modalities.
- Research the current semantic segmentation techniques and their limitations and advantages.
- Identify functional requirements for developing the model.
- Perform experiments to optimize the hyperparameters of deep learning models.
- Evaluate performance and compare metrics to find the best model
- Integrate the final model implementations with a GUI

1.2 Structure of Dissertation

Section 2 introduces the theory required to comprehend the project's methodologies. This chapter examines the labelling and segmentation of brain tumors, as well as the fundamental components of a convolutional neural network.

Section 3 gives some insight into the related work that has been done in brain tumor segmentation and also examines the U-Net architecture and how its building blocks are assembled for segmentation purposes.

Section 4 discusses the employed methods. First, the dataset and then the data pre - processing steps are examined. The Section then details the network architectures implemented and experimental design.

Section 5 explains the experiment settings and evaluation metrics that are used. This Section also presents the outcomes of the experiments described in Section 4. Predictions of the best models are shown. On the test dataset, the models are evaluated and compared. Finally, the developed graphical user interface is discussed.

In **Section 6**, the methods and outcomes are discussed, along with future work that can be performed to gain additional knowledge and enhance output. Additionally, issues uncovered during the writing of the dissertation are discussed.

Section 7 provides the conclusion of the completed work.

2 Theory

2.1 Gliomas and Brain Tumor Classes

There are numerous types of brain tumors. Malignant tumors account for fewer than one-third of all brain and other CNS (Central Nervous System) tumors identified in the United States, but they contribute the majority of deaths. Brain tumors are typically classified according to their main tumor site: primary brain tumors originate in the brain, whereas secondary, or metastatic, brain tumors spread from tumors that began elsewhere in the body [16].

The contour of these cells distinguishes them from the surrounding tissue. Brain tumors starts in the brain and only rarely spread to other parts of the body. Brain tumors are categorized into four types: gliomas, meningiomas, pituitary adenomas, and nerve sheath tumors. Glioma is a form of primary brain tumor that is common. They account for around 30% of all brain and central nervous system (CNS) cancers and approximately 80% of all malignant brain tumors [3].

Similarly, gliomas have been classed based on their aggressiveness. The WHO grading system for brain and CNS tumors [5] distinguishes four grades (grade I, II, III, and IV) based on histological characteristics and clinical outcome.

Low-grade gliomas (LGG) are primary brain tumors that develop in the brain or spinal cord from glial (supportive) cells. Gliomas of grades I and II are considered low-grade tumors, but the WHO restructured this type of tumor in 2016 [18]. Low grade gliomas might take years to form, but they normally progress to high grade gliomas over time. In the scan, a cyst can be observed, and it normally has little edema.

High-grade glioma is the most severe type of glial intra-axial tumor of the CNS. HGG also is responsible for the majority of malignant primary brain tumors and corresponds to WHO grade III and IV. Patients with HGG continue to have a bad prognosis. The HGG can be detected as a heterogeneous mass with surrounding edema in the MRI scans. There is commonly necrosis in the core region, as well as significant edema.

Brain tumor is divided into three distinct categories in the BraTS-dataset [11], which is the dataset used in this dissertation and is described in greater detail in Section 4.1. Necrosis, edema, and tumor enhancement are examples of frequently segmented tumor tissue types, which are described in greater detail below.

Enhancing Tumor

The portion of the tumor where cell division is active is classified as an enhancing tumor. This portion of the tumor is where tumor growth occurs; the rate of tumor growth gets to decide whether the tumor is benign or malignant. A faster rate of cell division results in a quicker growth rate and is regarded as more malignant.

Necrosis

The class necrosis is designated to the portion of the tumor where cells have died. This portion of the tumor has experienced unchecked cell death. Typically, this portion of the tumor is located in the tumor's center.

Edema

The portion of the tumor where fluid has accumulated is classified as having edema. Edema is also recognized as fluid retention or swelling, and it is most frequently observed in the vicinity of tumor enhancement and necrosis.

2.2 Semantic Segmentation

In semantic segmentation, every pixel of an image is assigned a class label. Formally, semantic segmentation is defined as finding the mapping between a real-valued tensor with integer-valued dimensions H, W, and C and a positive integer-valued tensor with integer-valued dimensions H and W, where the integer corresponds to a finite space of classes for the segmentation problem at hand.

This can be expressed mathematically as shown by Equation (1) below

$$\mathbb{R}^{H\times W\times C} \rightarrow \mathbb{Z}_{+}^{H\times W} \tag{1}$$

Segmentation of Brain Tumors

In brain tumor segmentation, labels are designated to tissues with similar characteristics. The brain can be roughly divided into tumor and non-tumor regions. Tissue within the tumor that shares the same features can then be subdivided into subcategories, with each subcategory being labelled separately. The labels are assigned to each MRI slice and utilized for network training and evaluation. Therefore, it is essential that these classifications are accurately delegated in order to produce a solid segmentation network. Commonly, multiple experts are employed to guarantee the accuracy of data labelling.

2.3 Deep learning

Deep learning is inspired by how biological systems process information. Biological brains are able to perform extremely complex tasks with ease, so it is reasonable to design systems that attempt to imitate their behavior. To learn multiple levels of representation and abstraction, deep learning entails conducting multiple layers of non-linear processing of information. The objective of deep learning is to comprehend

data such as images, audio, and text. Deep learning is also known as Artificial Neural Networks on occasion.

Supervised Learning

The most prevalent learning strategy in the field of machine learning is supervised learning, in which the mapping from x to y in Equation (2) is learned by adjusting the parameters w.

$$f(x; w) = y \tag{2}$$

Where x is an image, y is the segmented image in which each pixel has been labelled as belonging to a class, and f is the neural network with biases and weights w.

Artificial Neuron

The fundamental component of Artificial Neural Networks (ANN) is the Artificial Neuron, which is loosely based on how biological neurons function. The biological neuron receives multiple input signals from the dendrites and transmits only one electric signal to the axon. However, this isn't a linear process. There is no output from the neuron until the input signals reach a certain threshold, at which point a relatively powerful electrical impulse is generated. This is referred to as neuron activation.

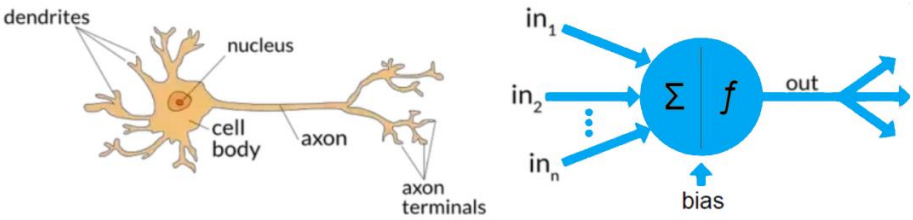

Figure 1. Figure illustrating biological neuron on the left and mathematical neuron on the right

This neuron behavior can be mathematically modelled as

$$O = \sigma\left(\sum_{i=1}^{n} w_i x_i - \theta\right) \tag{3}$$

Where O is the output of the artificial neuron, w_i is the proportion of input x_i that should make a contribution to the output, θ is the threshold, also known as the bias, and σ is a non-linear activation function. Figure 1 illustrates visual representations of the biological as well as mathematical neurons.

Multilayer Perceptron

Modeling a neural network is a natural progression from modelling a single neuron. Multi-Layer Perceptron (MLP) is the most elementary form of ANN. Which is comprised of many artificial neurons interconnected. This can be mathematically expressed as Equation (4)

$$V_i^{(l)} = \sigma\left(b_i^{(l)}\right), \qquad b_i^{(l)} = \sum_{j=1}^{N} w_{ij}^{(l)} V_j^{(l-1)} - \theta_i^{(l)} \tag{4}$$

Where $V_i^{(l)}$ represents the value of neuron i in layer l, σ is the non-linear activation function, $b_i^{(l)}$ is the local field of neuron i, $w_{ij}^{(l)}$ is the weight connecting neuron i to neuron $V_j^{(l)}$ in the previous layer, $l-1$, and θ_i is the bias.

In this simple case, all neurons are completely connected and feed forward, as shown in Figure 2, with a neuron in layer l having a connection to every neuron in layer $l-1$ and each neuron in layer l having connections with all neurons in layer $l+1$. "Input layer" represents l = 1, "Hidden layer" represents l = 2, and "Output layer" represents l = 3. Figure 2 depicts an illustration of this context. All layers other than input and output layers are referred to as hidden layers. As they are hidden from the network's interface, only the data or the loss function could be actively manipulated.

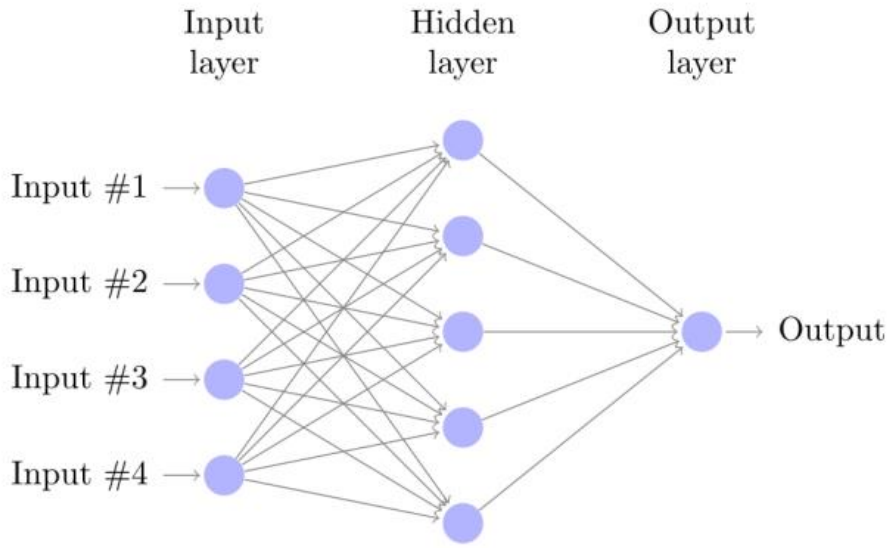

Figure 2. Fully connected feed forward MLP

Activation Functions

An activation function is really what gives the ANN its nonlinearity. ANNs could be reduced to one affine transformation in the absence of nonlinearities. Consequently, the activation function is from where the modelling ability of an ANN originates.

Rectified linear unit

The Rectified Linear Unit (ReLU) is a frequently employed activation function. This activation function was proposed by [19] on the basis that it qualitatively resembles modelling the dynamics of current through the neuron cell membrane. Equation (5) displays the formula for ReLU. Where b_i represents the local field in Equation (5).

$$\sigma(b_i) = \max(0, b_i) \tag{5}$$

The selection of ReLU is based on significant empirical evidence of its effect on enhancing learning [20]. It is assumed to be more effective because, unlike the conventional sigmoid and tanh activation functions, it does not experience saturation during forward feeding. ReLU also helps with the problem of vanishing gradients when performing backpropagation, as can be seen by differentiating the expressions in Equations (5) and (6). This assumes that the discontinuity in 0 can be ignored in numerical calculations. Since the gradient will be either 1 or 0 everywhere, updates will be more stable.

$$\sigma = \begin{cases} 1 \; if \; b_i > 0 \\ 0 \; if \; b_i < 0 \end{cases} \tag{6}$$

SoftMax layer

SoftMax is a specialized activation function that is frequently used in the final layer of ANNs for classification. As its name suggests, SoftMax is a soft version of the maximum function, meaning that the largest element of the function's real-valued input vector should tend toward unity and the other elements should tend toward zero. However, this should occur naturally with continuous values. Equation (7) displays the mathematical Equation for this activation function.

$$O_i = \frac{e^{\alpha b_i}}{\sum_j e^{\alpha b_j}} \tag{7}$$

Where b_i represents the local field described by Equation (7). The parameter determines the strength of the function's maximization effect and is frequently set to unity.

If, $\alpha \to \infty$ it is clear that $\mathrm{SoftMax}(b_i) \to \max(b_i)$.

SoftMax has three advantageous characteristics:

1. $0 < \mathrm{SoftMax}(b_i) < 1$, which indicates that each component can be perceived as a probability.
2. $\sum_i \mathrm{SoftMax}(b_i)$ equals 1, indicating that the output vector could be perceived as a probability distribution.
3. $\mathrm{SoftMax}(b_i)$ increases monotonically with b_i.

The combination of properties 1 and 2 is really what helps make SoftMax so famous for classification tasks, as the output of a neuron could be perceived as the probability that the neuron belongs to the class.

Loss Function

A loss function is used to determine the inaccuracy of an ANN and, consequently, the gradient when updating the ANN's parameters. A loss function ought to be positive. Depending on the type of problem at hand (regression, classification, etc.), various loss functions are necessary. The mathematical definition of a loss function is

given in Equation (8), where y represents the ground truth and y represents the model's predictions.

$$L(y, \hat{y}) \in \mathbb{R}_0^+ \tag{8}$$

Categorical Cross-entropy Loss

Categorical cross-entropy is a classification-related loss function [21]. Equation 9 displays the loss function.

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij}) \tag{9}$$

y represents the ground truth and $\hat{y}$ represents the prediction. N is the number of voxels, whereas C is the number of classes. Indices I and j denotes pixels and classes respectively. Since $\hat{y}_{ij} \in (0,1)$, Equation (9) is a well-defined loss function, but it must be combined with a SoftMax layer for the output to be perceived as probabilities.

Categorical Focal Loss

Focal loss (FL) [22] is also a form of Cross-Entropy. It reduces the contribution of simple examples and allows the model to concentrate more on learning challenging examples. It functions well in scenarios involving extreme class imbalance.

Using a modulating factor, $(1 - p_t)^\gamma$, Focal Loss proposes to de-emphasize simple examples and concentrate training on challenging negatives.

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma log(p_t) \tag{10}$$

Here, $\gamma > 0$ and when $\gamma = 1$, Focal Loss functions similarly to the Cross-Entropy loss function. Similarly, $\alpha \in [0,1]$ and can be tuned as a hyperparameter.

Dice Loss

In the computer vision community, the Dice coefficient is a popular metric for calculating the similarity between two images. Later in 2016, it was also implemented as the Dice Loss function [23].

$$DL(y, \hat{y}) = 1 - \frac{2y\hat{y} + 1}{y + \hat{y} + 1} \tag{11}$$

In this case, 1 is added to the denominator and numerator to prevent the function from being undefined in edge case situations, such as when $y = \hat{y} = 0$.

Convolutional Neural Network

A Convolutional Neural Network (CNN), which was introduced by [24] in 1999, is an ANN designed to utilize spatial features in data input. Conceptually, the idea is that earlier layers recognize simple shapes and features, while later layers can combine these relatively simple features to create more complex representations. This

concept is used to accomplish difficult tasks in the fields of computer vision and natural language processing, such as image classification and speech-to-text.

The central concept underlying a CNN is the incremental movement of a filter over the input data. Typically, the filter is a square matrix. Each filter element contains a real number. During the convolution, each filter element is multiplied by its corresponding input element. The result of a convolution is referred to as a feature map. Figure 3 depicts a visual representation of this procedure. Stride refers to the filter's step size, which determines the output size of the convolution. This can be expressed mathematically using the expression in Equation (12). V_{ijk} is the feature map for images, where i and j are the height and width of the feature map, and k is the quantity of filters in the convolutional layer. p and q represent the rows and columns of the filter, while r is the number of input channels.

This expression assumes that the stride in both dimensions is 1.

$$V_{ijk} = \sigma\left(\sum_{pqr} w_{pqkr} x_{p+i-1,q+j-1,r} - \theta_k\right) \tag{12}$$

Depending on the required output size of the convolution, padding can be used to shape the result by attaching empty entries to the convolution at the periphery of the input.

To further simplify the feature maps, pooling can be used to filter the feature map values so that a down sampled feature map contains only the maximum values from a $n \times n$ neighborhood in each pixel of the pooled feature map.

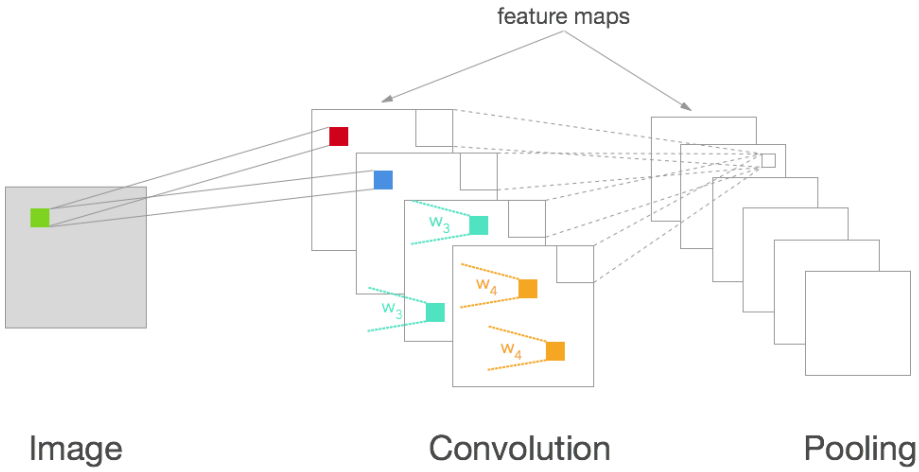

Figure 3. Image of Convolutional Neural Network by Will Whitney licensed under CC BY-SA 4.0 [25].

2.4 Batch Normalization

Batch normalization [26] is an adaptive method of reparameterization being used to overcome the difficulty of training very deep learning model. In practice, it greatly facilitates the optimization of deep neural networks.

The randomness from parameter initialization and input would accrue in each layer as training progresses, and the distribution of input for each layer would change as the previous layer changed its parameters via gradient descent. This effect, known as internal covariate shift, would make training the neural network more difficult. The batch normalization method helps reduce internal covariate shift. It performs a normalizing step on each layer's input to ensure a zero mean and unit variance.

Batch normalization is a straightforward and efficient method for reparametrizing nearly any deep network. Reparameterization of a deep neural network can significantly simplify the coordination of updates across multiple layers. Batch normalization can be applied to any input or hidden neural network layer. This Section describes how batch normalization operates.

Consider an m-sized mini-batch $\mathcal{B}$. Since normalization is applied independently to each activation, we will concentrate on activation x. Therefore, there are m values of just this activation in the mini-batch, denoted by $\mathcal{B} = \{x_{i\ldots m}\}$. The steps involved in batch normalization are outlined below.

Normalize the batch of values $\mathcal{B} = \{x_{i\ldots m}\}$ utilizing batch's mean and variance calculated as:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{13}$$

and

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2, \tag{14}$$

where $\mu_{\mathcal{B}}$ is the mini-batch mean and $\sigma_{\mathcal{B}}^2$ is the mini-batch variance.

Calculate the normalized values $\hat{x}_{1\ldots m}$ as,

$$\hat{x}_{1\ldots m} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{15}$$

Where ϵ is a sufficiently small constant imposed to prevent the normalisation from producing an undefined denominator. The normalised values $\hat{x}_{1\ldots m}$ are then subjected to linear transformations.

$$y_i = \gamma\hat{x}_i + \beta \tag{16}$$

The variables γ and β are learned parameters that permit any standard deviation and mean for the new normalized values $y_{1\ldots m}$. Normalizing the value of a unit can diminish the expressive capability of a neural network containing that unit. It is com-

mon practise to replace the batch of hidden unit activations $x_{1\ldots m}$ with $y_{1\ldots m}$ in order to preserve the expressive power of the network. The new parametrization of the network can represent the same family of input functions as the old parametrization, but it is much simpler to train with gradient descent.

In convolutional networks, we must apply the same normalizing and at each location in a feature map to ensure that the feature map's statistics are identical throughout all spatial locations [27].

2.5 Weight Initialization

Initializing weights in a structured manner is one way to reduce the presence of vanishing gradients. [28] demonstrated that it is potentially justifiable to initialize weights according to Equation (17) while using ReLU as an activation function. This method will henceforth be known as he-normal.

$$w_l \sim \mathcal{N}(0, \sigma_{w_l}) \tag{17}$$

σ_l represents the standard deviation of the initial weights w_l in layer l of the ANN, while n_l represents the number of weights in layer l. σ_l is defined by the expression (18).

$$\sigma_{w_l} = \sqrt{\frac{2}{n_l}} \tag{18}$$

The rationale for this weight initialization is that if the weights are initialized with a constant variance, the expected value of each neuron will depend on the number of neurons in the preceding layer. Therefore, if there were numerous weights in the preceding layer, the variance will be tiny as well as the expected value of a local field in a given neuron will approach zero.

2.6 Regularization

Overfitting is a major issue while trying to train deep learning models. When a model begins to memorize the training data rather than generalizing, this is known as overfitting. Memorizing the training data would then decrease the model's performance when making predictions from out-of-sample data.

Regularization refers to all methodologies that aim to reduce model overfitting.

A few of these techniques that are employed in this dissertation are described in detail below.

Early stopping

Early stopping is a strategy wherein training is terminated because of an increase in the loss function on the validation data set over a predetermined number of epochs, also referred to as patience in the Keras software library [29]. This method is effective because a rise in the loss function assessed on the validation data suggests that the

model has ceased generalizing and begun to memorize the training data. It is recommended at this point to stop training the model. Figure 4 provides a visual depiction of the typical behavior observed during model training.

Figure 4. Figure illustrating the increase in validation loss on further training, meaning model is starting to overfit on training data

Dropout

Dropout is a training technique in which each neuron is dropped with a probability *p* during every forward pass through the network. As depicted in Figure 5, this can be viewed as training a combination of network architectures. This forces the network to discover alternative pathways through which information can transfer. Since there is no promise that a specified neuron will be in the network at a given forward pass, dropout frequently makes the network less susceptible to overfitting, as neurons cannot co-adapt as much.

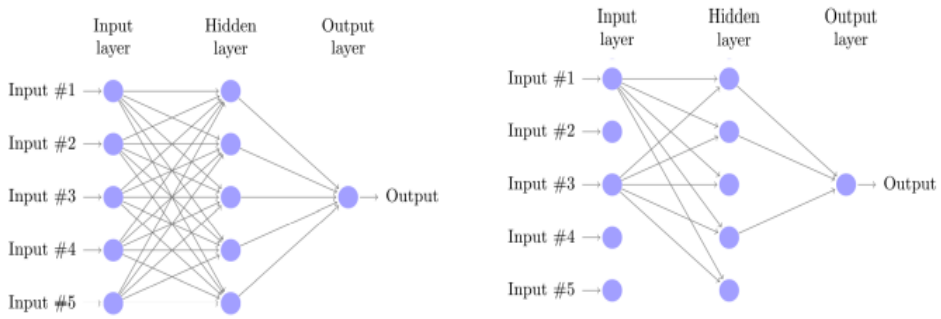

Figure 5. Figure illustrating difference in network structure upon introducing dropout. Left side is network with no dropout and right side shows how network changes with dropout implemented.

2.7 Optimizer

ADAM, short for adaptive learning rate optimization algorithm, is a well-known and widely used optimizer that was proposed in a paper by [30]. This is a variant of stochastic gradient descent. However, it utilizes moving averages of the initial two moments of gradients to make parameter updates more stable. The approximate formulae for these updates can be found in Equations (19) through (23).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{20}$$

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{21}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{22}$$

$$w_t = w_{t-1} - \eta \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{23}$$

m_t and v_t represent the first and second moving averages of the gradients, respectively. β1 and β2 are hyper parameters which dictate the influence of the gradients in the previous time step on the gradient updates in the current time step. To estimate these statistics, one employs the formula for unbiased estimators $\widehat{m}_t$ and $\hat{v}_t$, which is then utilized to update the ANN's weights. η represents the learning rate and ϵ is a small constant used to prevent division by zero.

2.8 Transfer Learning

For many tasks in the field of machine learning, cutting-edge models have been able to produce increasingly accurate results. However, a majority of these models rely on enormous quantities of accurate labelled data, which are both time-consuming and costly.

Transfer learning is a machine-learning technique that stores and then applies information learned in one domain to a separate but related domain. It allows the use of prior knowledge. In computer vision, low-level features, such as shapes as well as edges, can be employed in general tasks that transfer knowledge between tasks.

In each convolutional layer, deep neural networks capture different features. The lower convolution operation captures low-level features, whereas the higher convolution operation is capable of capturing more complex features. Typically, the final fully connected layer is utilized to determine task scores. When the pre-trained model is applied to a new task, it is crucial that the previously acquired features remain intact. As shown in Figure 6, the most common approach is to freeze layers from a pre-

trained model in order to maintain the general features. New trainable layers are then added to the model's top, which can assist in adapting the old features to new scenarios. Only newly added layers will undergo training.

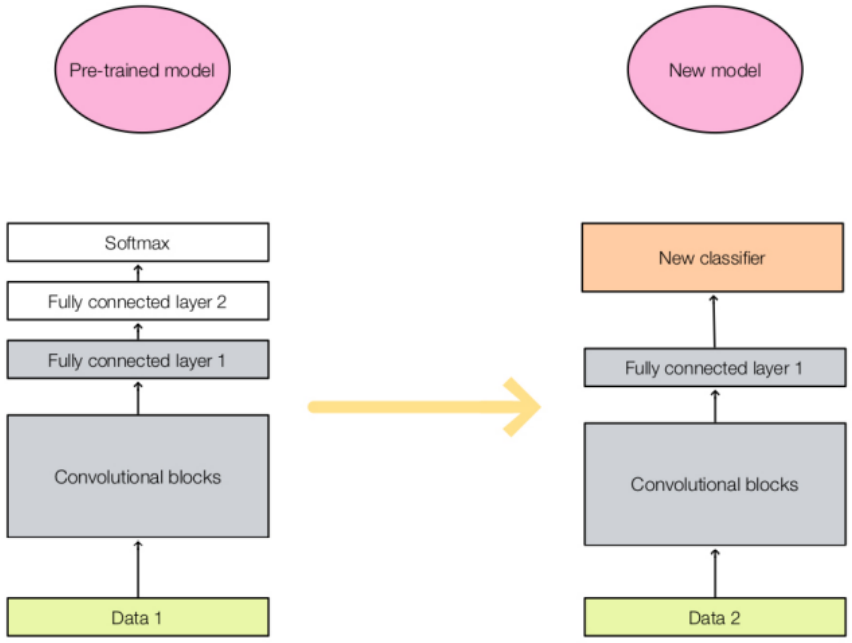

Figure 6. Figure illustrates transfer learning using a previously trained model. The grey blocks represent frozen layers during training.

In refined training, there are no fixed layers. Using the previously trained coefficients as initial values, the entire model is re - trained on new data. The parameters of the grey blocks in Figure 6 will indeed change during training as a result of back propagation.

3 Related work

3.1 Sliding Window Approach

The earliest ANN advances in semantic segmentation tasks employed regular CNNs in which each pixel represented a separate classification task. Therefore, there would be a sliding window in which a portion of the entire input image is provided as input and the central pixel of this portion is predicted. The sliding window was then shifted by one pixel, and a new patch was sent to the network. [31] provided an example of this type of architecture. Figure 7 displays the architectural layout. The input

to the network is a 4x33x33 tensor, which undergoes multiple convolutions before the feature maps become flattened and a SoftMax layer is used to classify the pixel value of the given input patch.

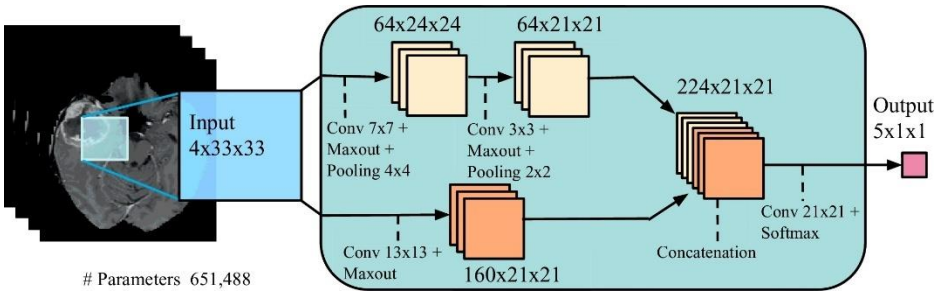

Figure 7. Image of architecture using sliding window approach by [31].

3.2 Fully Convolutional Network

In contrast to an MLP or a traditional CNN, a fully convolutional neural network (FCN) for classification tasks incorporates only convolutional layers. The advantage of a CNN over a Multi-Layer Perceptron is that it is less restrictive with regard to the size of the input, as the number of weights in the first layer of a MLP is a direct function of the size of the input, whereas the filters of a CNN are the same size and are independent of the size of the input.

The advantage of an FCN over a traditional CNN is that the feature maps are not required to be flattened and also that fully connected layers are present, which results in an FCN having fewer weights on average than a CNN. FCNs also enable it to output entire images rather than just one pixel at a time. In the case of images, this network type can accept any size of the image as input. [32] introduced the initial FCN. The task was sematic segmentation, where the input to the network was a complete image and the output was the image's mask.

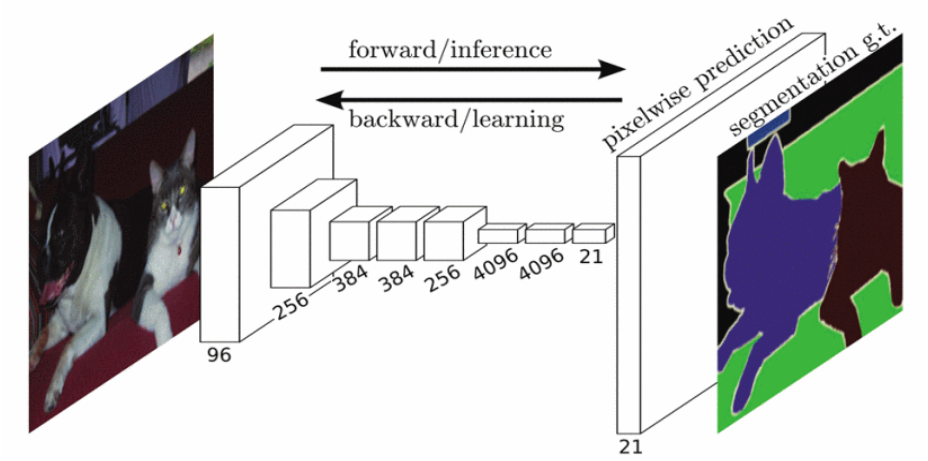

Figure 8. Image of architecture used by [32].

3.3 U-Net

A common architecture for segmentation problems is the encoder-decoder architecture. Encoder-decoder architecture is depicted in Figure 9 as a simplified diagram. The encoder can be a pre-trained classification network such as VGG, and the decoder's job is to up sample the input features and generate pixel-level predictions.

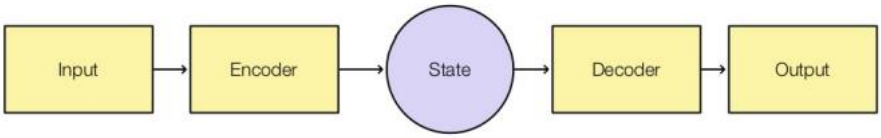

Figure 9. Simplified representation of encoder-decoder architecture.

[13] introduced the U-Net fully convolutional neural network (FCN). The network follows an encoder- decoder architecture. Which implies that it accepts an input and produces an object with a similar shape to the input. Regarding semantic segmentation, U-Net accepts a $H \times W \times C_{in}$ input and returns a $H \times W \times C_{out}$ output. If the input is an image, then H is the height, W is the width, C_{in} is the number of input channels, and C_{out} is the number of output channels.

In the semantic segmentation problem, the C_{out} equals the number of distinct classes among all the labels.

Segmentation begins with two layers of convolutions, followed by down sampling through max-pooling. This process is repeated four times before being followed by a new set of convolutions. The final feature maps are then subjected to up sampling through the use of transposed convolutions. Figure 10 depicts the concatenation of these up-sampled feature maps with the feature maps of the convolution operation before the final down sampling. This combination is what made the U-Net architec-

ture novel in comparison to previous works. This reuse of earlier outputs aids in restoring fine detail when up sampling, as this information would be lost in smaller feature maps without the skip connections "over" the down sampling blocks.

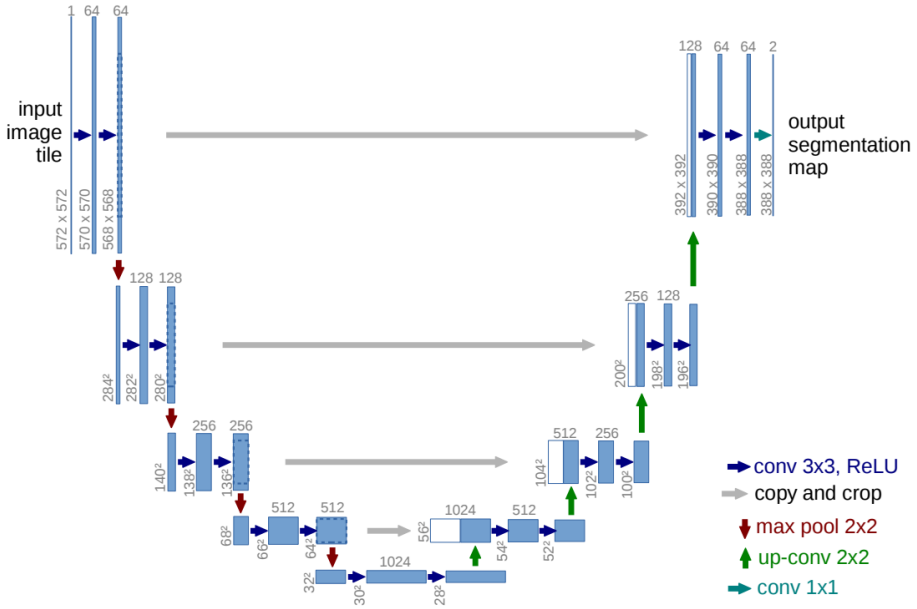

Figure 10. U-Net architecture proposed by [13].

4 Methods and Data

4.1 Dataset

MICCAI BraTS Dataset

The 2020 MICCAI BraTS training dataset features 369 patients with four different modalities: T1, T2, T1ce, and FLAIR. All scans are in NIfTI format [33] and were obtained using distinct clinical protocols and scanners from 19 different institutions. All images were manually segmented by one to four raters who all followed the same annotation process, and the annotations were approved by expert neuro-radiologists. The notations are for the enhancing tumor (label 4), peritumoral edema (label 2), necrotic and non-enhancing tumor core (label 1), and everything else in the image (label 0). The data set includes both low-grade glioma (LGG) and high-grade glioma (HGG) cases, with tumor classifications determined by specialists in the field. The data have been co-registered to the same anatomical template, interpolated to the same resolution ($1mm^3$), and skull-stripped [11,14,15]. Each patient is represented by 155 images referred to as slices for each modality. Only three modalities (T2, T1ce and FLAIR) are used as inputs for our networks, as they provide sufficient context for brain tumor

segmentation and also reduce training time significantly. As a result, this arrangement was chosen for our work.

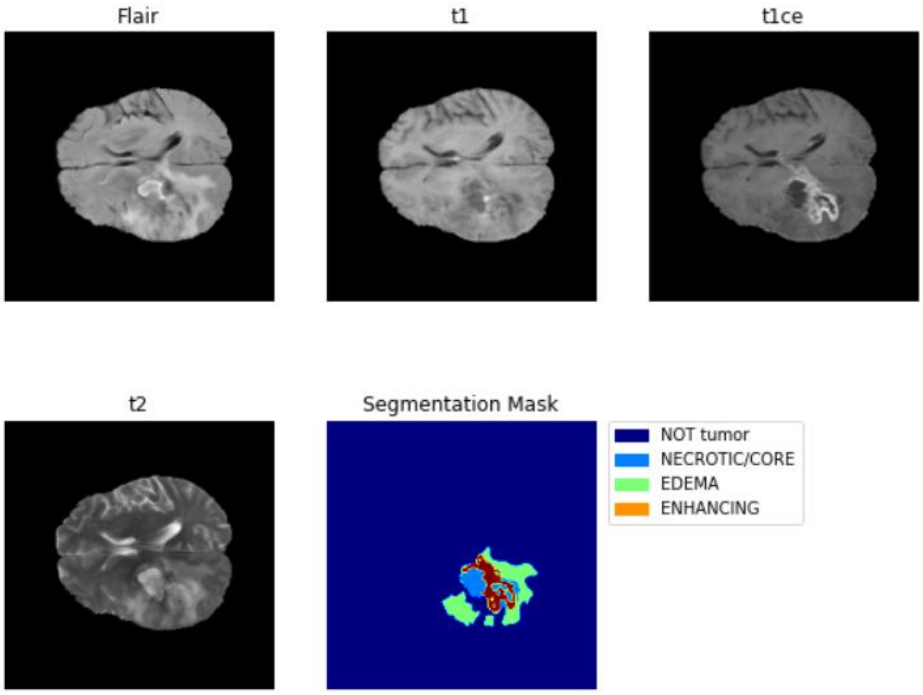

Figure 11. Illustrates the different MRI modalities in the dataset and also the ground truth mask. Shown layer is of patient number 5, slice is 80 along the axial plane.

Training, Validation and Testing

The dataset is split into 70% for training, 20% for validation and 10% for testing as shown in table below.

Data portion	Explanation	Chosen split	Number of patients
Training	To train the models	70%	240
Validation	Used for hyperparameter optimizations and access training performance.	20%	68
Testing	Evaluate the performance of model on unseen data	10%	36

Table 1. Table depicting chosen split sizes and explanation of chosen splits

Imbalanced Data

Unbalanced classes are one of the key factors that contribute to the difficulty of brain tumor segmentation. Typically, healthy tissues make up the majority of the

brain, with tumors taking up a minor portion. In the semantic segmentation task, the neural network will tend to classify more pixels as background in order to achieve a better loss function outcome. To deal with this problem of unbalanced classes, we include sample weighting in the loss function as discussed in later Section. We use preprocessing steps such as image cropping to mitigate this issue and we employ loss functions that are specifically tuned to handle such tasks.

4.2 Data Preprocessing

Preprocessing is defined in this dissertation as any approach or technique applied to image data prior to further analysis, such as image segmentation and image classification. Data pre-processing is necessary because methods used for further analysis make assumptions about the incoming image data. It may be important, for example, for images to have a specific image dimension, or for the image dimensions of images within a multi-modal image collection to be the same. General brain MR preprocessing pipeline typically comprises of the following steps: 1) converting image format, 2) registration and co-registration, 3) bias field correction, 4) skull stripping, and 5) intensity normalization. In our case, as we are using the Brats 2020 dataset which is already skull stripped and all MR image modalities are co-registered to the same anatomical template, interpolated to 1 mm^3 resolution. The next paragraphs go over each of the processes that have been applied to our dataset in detail.

Image Format Conversion

Each MR image file is originally present in NIfTI file format, with .nii extension. Out of many visualization and imaging toolkits available for medical imaging files we chose to use SimpleITK and nibabel libraries to read as well as convert each NIfTI file into a NumPy array so that it can be efficiently processed.

Cropping

The data size of each MR image is 240x240x155, but the vast majority of pixels in this correspond to background class. The tumor region usually accounts for less than 2% of all labelled data available. To solve this heavy class imbalance problem, we employ use of radiomics library which provides us with the coordinates of the centre of mass of the tumor region. Using this knowledge, we have cropped the images into size of 128x128x128 keeping the tumor as a centre. 128 is chosen as we need image size to be a power of 2 as our U-Net architecture will half the image size at every stage during encoding or contraction path.

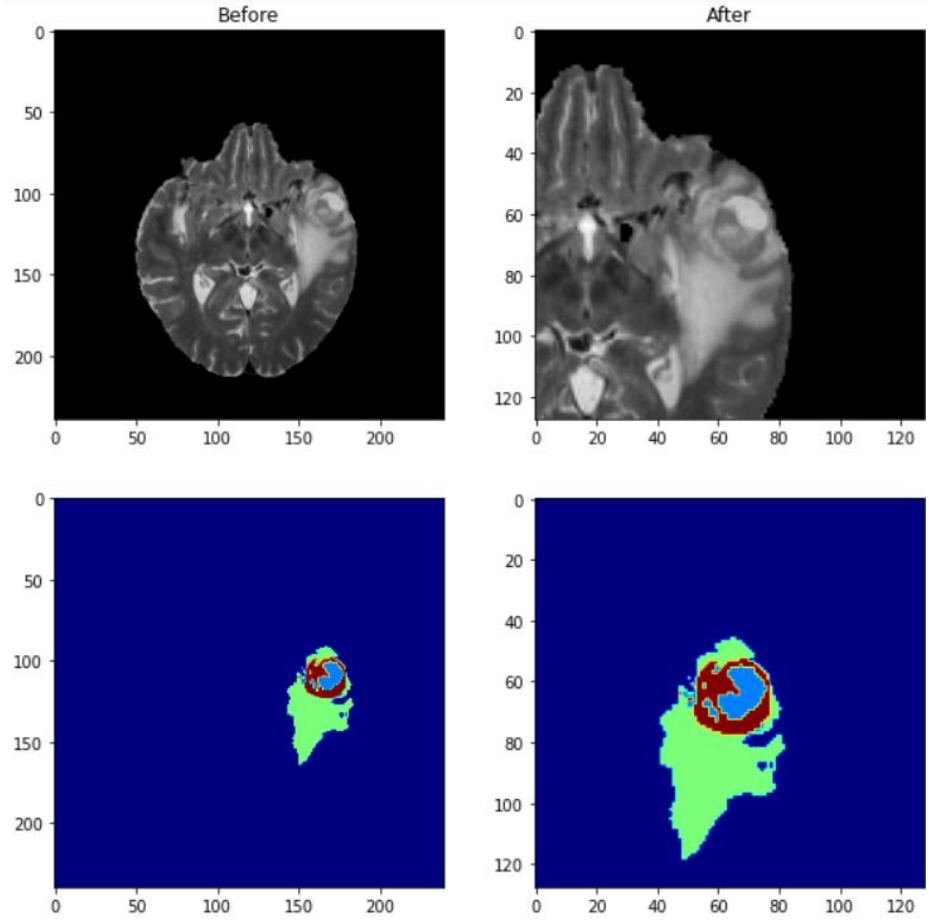

Figure 12. Figure showing MRI FLAIR image and associated mask before and after applying cropping.

Scaling

Unlike radiography or CT, absolute voxel intensities in MR images do not represent any physical quantities and lack unity. As a result, the voxel intensities in an MR picture are only relevant in respect to the MR image's intensity distribution. The intensity distributions of MR images are affected by the scanning process as well as other scanner-specific characteristics. As a result, MR pictures acquired from similar objects using different scanners or scanning conditions frequently show significant differences in intensity distributions.

We use scikit-learns min-max scaler to linearly scale the voxel-values to be between 0 and 255.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{24}$$

, where X' and X are rescaled and original images respectively and X_{min} and X_{max} are minimum and maximum intensity values of the image.

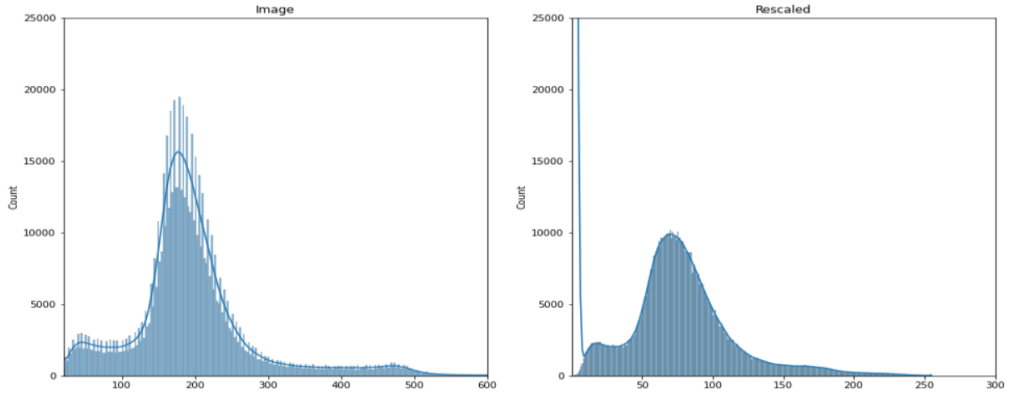

Figure 13. Figure illustrates the distribution of an original MR image and after applying rescaling.

Mean Normalization

We apply z-transformation to change the distribution such that the intensity distributions of an image will be having zero mean and unit standard deviation.

$$Z = \frac{X - \mu}{\sigma} \tag{25}$$

, where Z and X are the z-transformed and the original image respectively, and μ and σ are the mean and standard deviation of the original image X.

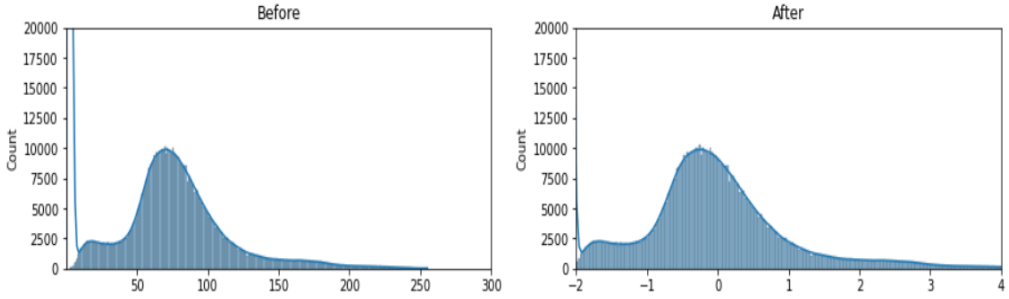

Figure 14. Figure illustrates the outcome of applying the z-transformation to an image.

N4 Bias Field Correction

MR pictures can suffer from bias field disturbances due to the magnetic field heterogeneities of MRI scanners. A bias field is a low-frequency noise signal that can cause the high-frequency contents of an MR picture to become less prominent [34].

The authors in [35] developed a N4 bias field correction, publicly available for example via the N4BiasFieldCorrectionImageFilter in SimpleITK 5.0, to correct the effects of MR bias fields.

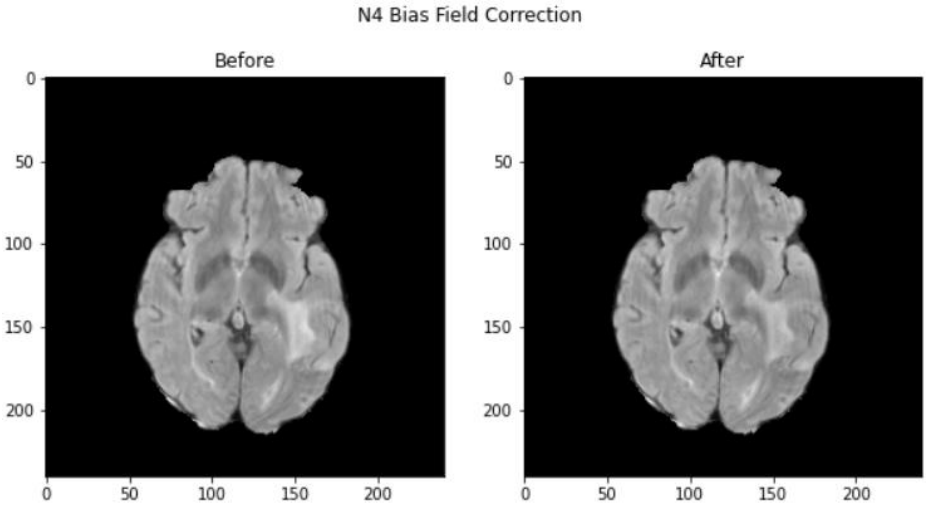

Figure 15. Figure illustrating the FLAIR MR image before and after its bias field correction.

4.3 CNNs Implemented

U-Net

This dissertation's deep learning model is built on the 2D U-net architecture as explained in Section 3.3. Instead of using 2D patch-based approach for classification as discussed before, we use the entire 3D MR volumes stacked together as an input, so that the context is not lost and out model uses this extra contextual information for better results.

For tumor segmentation, we modify the fundamental U-Net. The original U-Net only utilizes the viable portion of every convolutional layer, i.e., the segmentation map hardly consists of pixels with complete context in the input image. Because of this, the resolution of the final segmentation map is lower compared to the original image. To anticipate the labels of each pixel within the input image, the original U-Net must employ an overlap-tile technique. By reflecting the input image to estimate the labels of the pixels in the image's border region, the missing context is extrapolated. However, this is inconvenient, and the overlap-tile technique is ineffective, particularly when the tumors are close to the boundary, so we use zero padding around the input boundary before performing the convolutional operation, ensuring that the output size is identical to the input size. Consequently, the output resolution is identical to the original image size. We can directly anticipate all pixel labels using a single forward propagation without mirroring the input image, rendering the over-lap-tile technique obsolete. In addition, to prevent overfitting, we use dropout [36] and early stopping as regularization techniques, as described in Section 2.6.

Also, we added two more convolutional layers in encoder and consequently two more transposed convolutional layers in decoder paths to improve efficiency. We added batch normalization layers explained in Section 2.4 to better train our U-Net model.

Also, we have proposed use of weighted loss function as explained later in Section 4.4 to deal with the unbalanced class problem in brain tumor segmentation.

The proposed network receives three modalities as inputs in three channels and outputs four channels, each carrying the probability of a pixel belonging to a class using SoftMax as last layer activation function.

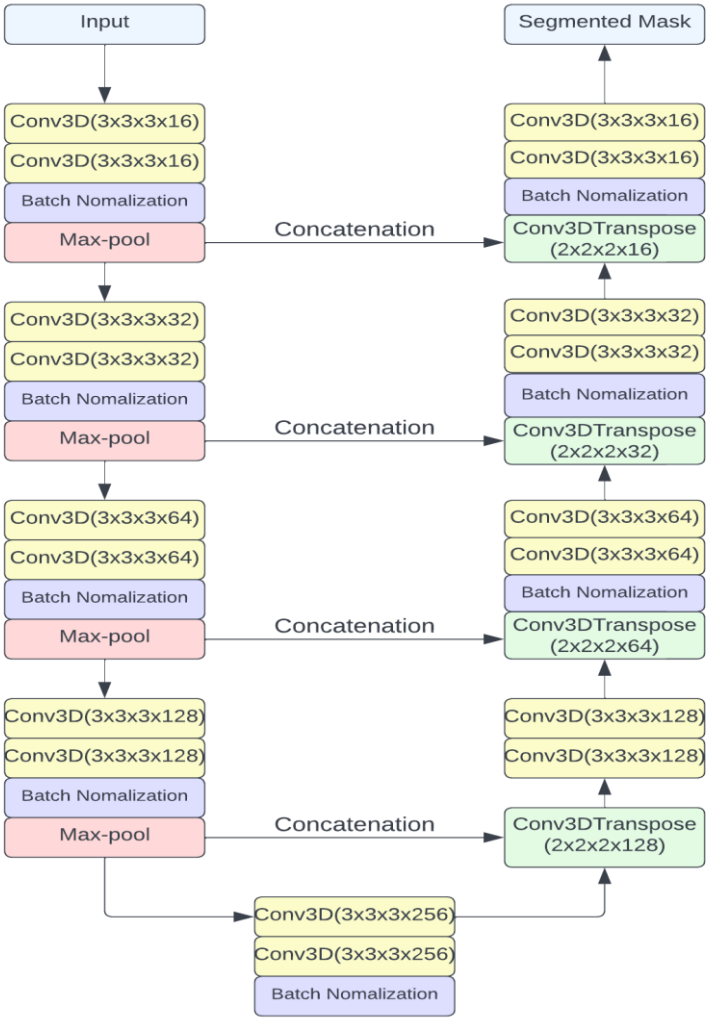

Figure 16. Structure and settings details for U-Net.

U-Net with modified encoder

Segmentation model 3D (Python) GitHub repository is a very useful and efficient toolkit for image segmentation tasks. For binary and multiclass picture segmentation, many model architectures are offered, including but not limited to U-Net, PSPNet, and FPN, with 99 encoders pre-trained on the ImageNet dataset. When compared to training a model from scratch, pre-trained models give better and faster convergence, requiring fewer resources. In this project, we have used well-known networks such as ResNet, and VGG as encoder path in the U-Net architecture.

ResNet. For image classification, residual networks are the cutting-edge network architecture [37]. Figure 17 depicts a residual network building block. A shortcut connection and element-wise addition are used to perform the operation $\mathcal{F} + x$. Batch normalization, as detailed in Section 2.4, is used immediately after each convolving layer and before the Rectified Linear Unit (ReLU) activation. The network does not employ dropout.

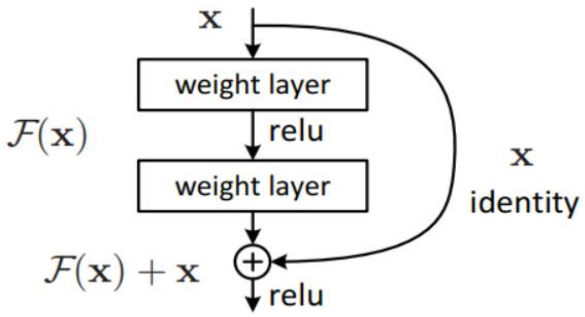

Figure 17. Residual learning: a building block.

VGG. VGG stands for Visual Geometry Group, and it is a multi-layered deep Convolutional Neural Network (CNN) architecture. The "deep" refers to the number of layers, with VGG-16 or VGG-19 having 16 or 19 convolutional layers, respectively. The encoder part of the U-Net is same as the VGG architecture but without the last fully connected layers. Batch normalization is applied only in the decoder part, after each convolving layer and before the Rectified Linear Unit (ReLU) activation. The network does not employ dropout.

4.4 Experimental Design

U-Net

The U-Net, as described in Section 4.3, is implemented and trained using a variety of experimental tests observing loss functions, dropout rates, and weight initialization described in this Section. The training and validation loss and IOU scores as a function of epochs for each of the U-Net experiments will be presented. The best perform-

ing experiment will be used to evaluate the test dataset in terms of IOU scores for each class.

Loss Function

We experiment with different loss functions as described in Section 2.3. To deal with the class imbalance problem we use sample weighting in our loss function. This method involves assigning a greater weight to minority classes and a lesser weight to majority classes, allowing the neural network to focus more on the minority classes. The weights are proportional to the number of samples within each class, but have a greater value. The formula for estimating weight is given by Equation (26)

$$w_j = \frac{n_{samples}}{n_{classes} \times n_{samples_j}} \tag{26}$$

where w_j represents the weight for the jth class, $n_{samples}$ represents the total number of samples in the dataset, and $n_{samples_j}$ represents the number of samples in the jth class. We utilize a combination of weighted dice loss and categorical focal loss as shown by code below, where class weights are calculated using Equation (26).

```
dice_loss =
sm.losses.DiceLoss(class_weights=np.array([wt0, wt1, wt2,
wt3]))
cfc_loss = sm.losses.CategoricalFocalLoss()
overall_loss = dice_loss + (1 * cfc_loss)
```

Dropout

This experiment looks at the effects of utilizing different rates in the dropout layers, which is explained in Section 5.4.

Weight Initialization

In this experiment, the weight initialization scheme utilizing he-normal, as described in Section 2.5, he-uniform and random-uniform are examined for their effects on network training. Best loss function is used from previous experiment.

U-Net with encoder as ResNet

We experiment with ResNet34 and ResNet50 as U-Net backbones, architectures of which are explained in Section 4.3. We investigate the differences between transfer learning and refined learning for this case.

U-Net with encoder as VGG

We experiment with VGG16 and VGG19 as U-Net backbones, architectures of which are explained in Section 4.3. We examine the differences between transfer learning and refined learning for this case.

5 Experiments and Results

5.1 Experiment Settings

All of the deep neural networks are trained by Adam [30], which is an adaptive first-order gradient-based optimization algorithm described in Section 2.7. The networks are trained on 40 epochs with an initial learning rate of 0.0001. The batch size for training is set to 1. Keras callback method is used to decrease the learning rate by multiplying the learning rate by 0.2 to slow it down, if the validation loss does not improve after two epochs. We also use early stopping, which means that if the loss does not get better after 3 iterations, the training process will end. We implement the deep CNNs in Python using the open-source machine learning libraries Tensorflow and Keras. Our experiments are done on computer with 16 GB RAM and 6 GB GPU memory (main memory).

5.2 Evaluation Metrics

Accuracy:

Accuracy is one of the most straightforward evaluation metrics for classification tasks; it is simply the ratio of correctly classified samples to the total number of samples. This is mathematically expressed in Equation (27).

$$Accuracy = \frac{Correctly\ classified\ samples}{Total\ number\ of\ samples} \tag{27}$$

In cases of class imbalance, it can be difficult to interpret how successfully the model is performing using only this metric. This is the case with this dissertation, as discussed in Section 4.1.

Intersection Over Union:

Intersection over union (IoU) is the commonly used evaluation metrics in semantic image segmentation. It is also called the Jaccard index or the Jaccard similarity coefficient. The IoU is computed by dividing the overlap between the predicted and ground-truth annotations by their union. Let n_{ij} represent the number of pixels of class i predicted to be of class j. Assuming that there are N_c distinct classes, let $t_i = \sum_j n_{ij}$. Mean IoU represents the average IoU across all classes:

$$IOU_{mean} = (1/N_c) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii} \right) \tag{28}$$

5.3 Experiment with different loss functions

In this Section, we compare the metrics of the various loss functions used to train our U-Net model. The best IOU score and accuracy is achieved by combining weighted dice and categorical focal loss, as shown in Table 2. Utilizing dice loss alone results in the second-best scenario, but as we will see in Section 5.9, even though the metrics appear comparable, the weighted loss function results in a greater improvement in IOU scores for minority classes.

Loss Function	Validation Accuracy	Validation IOU Score
Categorical cross-entropy	0.9573	0.4741
Categorical focal loss	0.9810	0.5994
Dice loss only	0.9820	0.6698
Weighted dice + categorical focal loss	0.9874	0.6759

Table 2. Table showing validation accuracy and validation IOU score for each loss function.

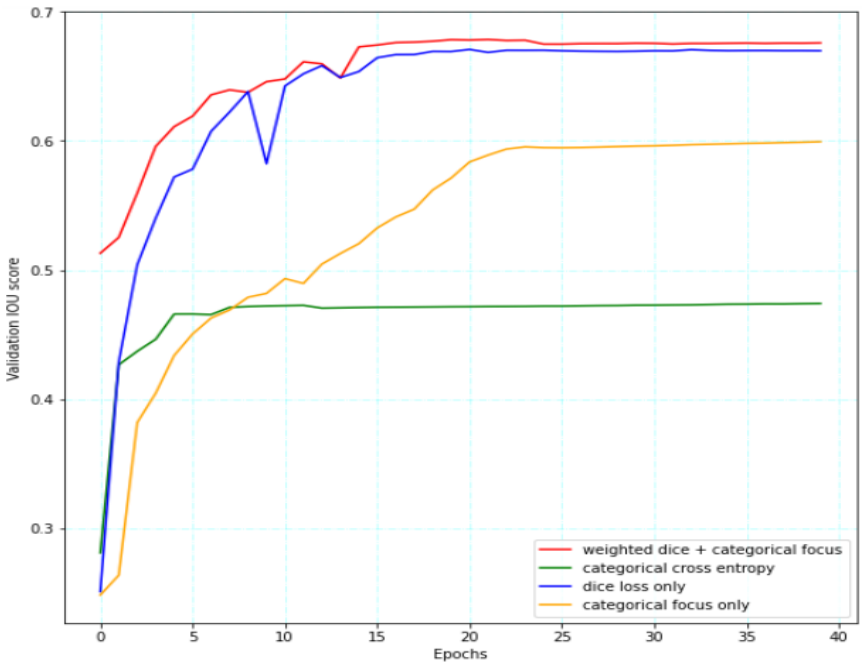

Figure 18. Graph illustrating validation IOU score for different loss functions.

5.4 Experiment with different dropout rate

We experiment with various dropout rates for our U-Net model in this Section. From table 3, we can conclude that a dropout rate of 0.1 yields the best results. 0.2 also appears to perform comparably, whereas 0.3 performs the worst of the three. Based on our experiments, we can conclude that for a dropout rate between 0.1 and 0.2, our network trains efficiently on the training data and leads to improved generalization. Previous best loss function is used.

Dropout rate	Validation accuracy	Validation IOU score
0.1	0.9874	0.6759
0.2	0.9833	0.6727
0.3	0.9795	0.6354

Table 3. Table showing validation accuracy and validation IOU score for different dropout rates.

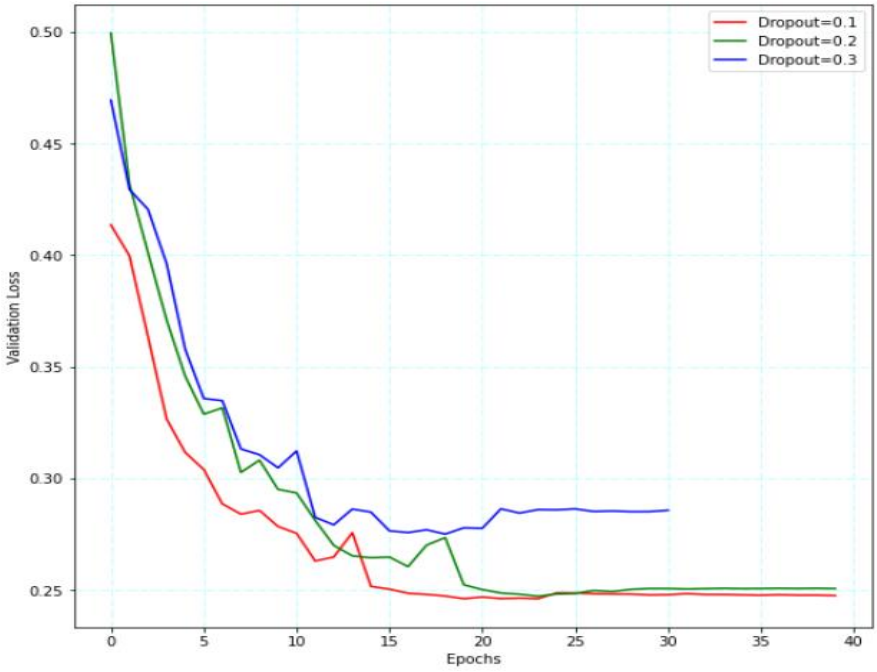

Figure 19. Graph illustrating validation loss curves for different dropout rates.

5.5 Experiment with different weight initializations

The results of our experiments with three distinct weight initializers for our kernels are shown in Table 4. Using random-uniform, our network does not appear to learn with increasing epochs, so model training was terminated after 24 epochs using early stopping. We obtain the best results by using he-uniform as the weight initializer, as it converges more quickly and with much less loss, as shown in the Figure below. Previous best dropout rate is used.

Weight Initializer	Validation Accuracy	Validation IOU score
he-normal	0.9811	0.6572
he-uniform	0.9874	0.6759
random-uniform	0.9811	0.6570

Table 4. Table showing validation accuracy and validation IOU score for different weight initializations.

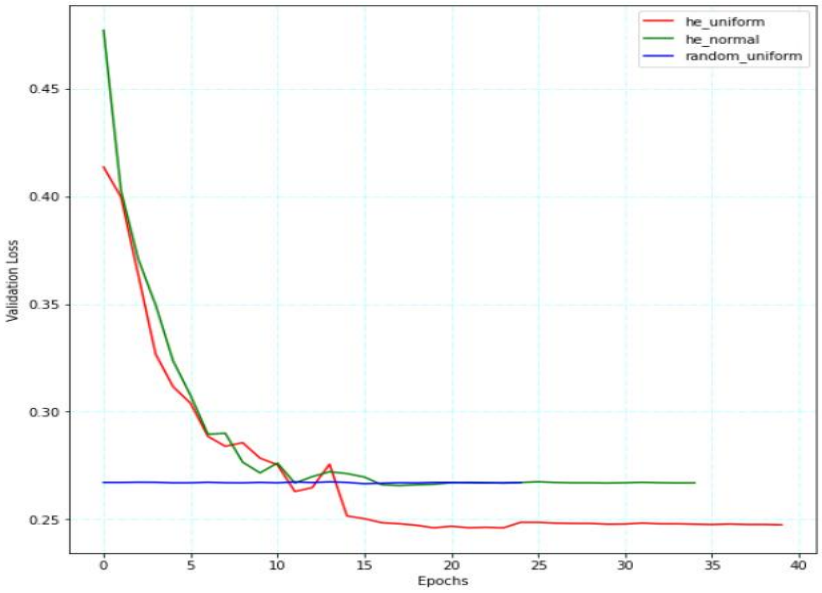

Figure 20. Graph illustrating validation loss curves for different weight initializations.

5.6 U-Net with VGG as encoder

Using VGG as an encoder in the U-Net architecture does not produce satisfactory results, as the validation IOU scores are significantly lower than those of the previous U-Net model. The outcomes of using VGG16 and VGG19 with and without the pre-trained weights are shown in Table 5. VGG19 takes advantage of transfer learning and converges more quickly and achieves higher metric scores than VGG19 without pretrained weights. As evidenced by the metric scores, the additional convolutional layers in VGG19 provide a distinct learning advantage over VGG16 for complex features. The loss function used is weighted dice loss.

Modified encoder type	Pretrained Weights	Validation accuracy	Validation IOU score
VGG16	ImageNet	0.8657	0.2458
VGG16	None	0.8651	0.2506
VGG19	ImageNet	0.9491	0.4069
VGG19	None	0.9491	0.3890

Table 5. Table showing validation accuracy and validation IOU score for different encoder types used and the associated weights.

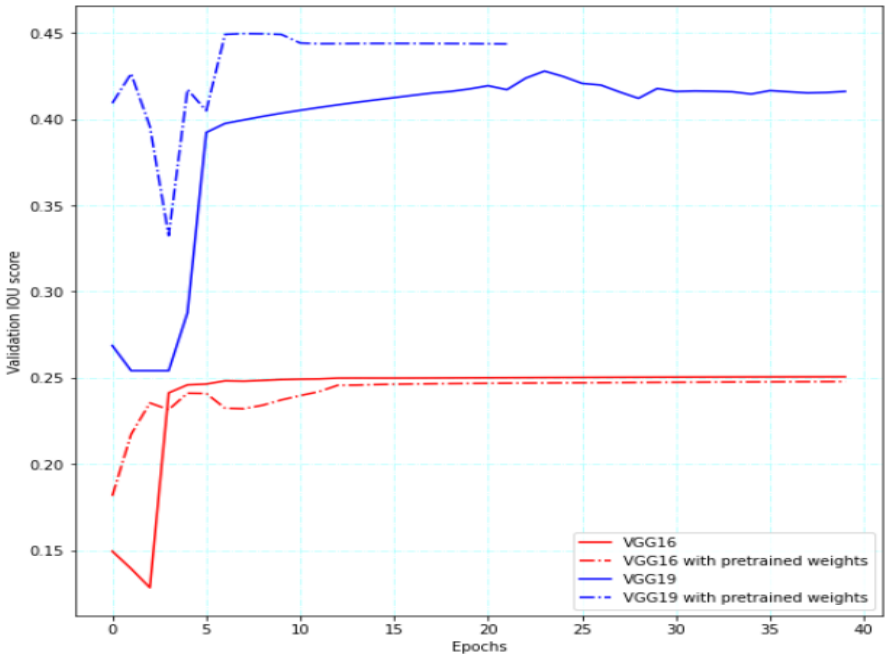

Figure 21. Graph illustrating validation IOU scores for different VGG models as encoder.

5.7 U-Net with ResNet as encoder

Table 6 indicates that when ResNet is used as the encoder for U-Net, the ResNet50 network provides comparable metric scores to our previous U-Net model. In addition, the additional convolutional layers in ResNet produce significantly higher training and validation scores. In addition, models without pretrained weights perform marginally better than "ImageNet" weights. The loss function used is weighted dice loss.

Modified encoder type	Pretrained Weights	Validation accuracy	Validation IOU score
ResNet34	ImageNet	0.8812	0.2447
ResNet34	None	0.9251	0.2460
ResNet50	ImageNet	0.9764	0.6310
ResNet50	None	0.9798	0.6454

Table 6. Table showing validation accuracy and validation IOU score for different encoder types used and the associated weights.

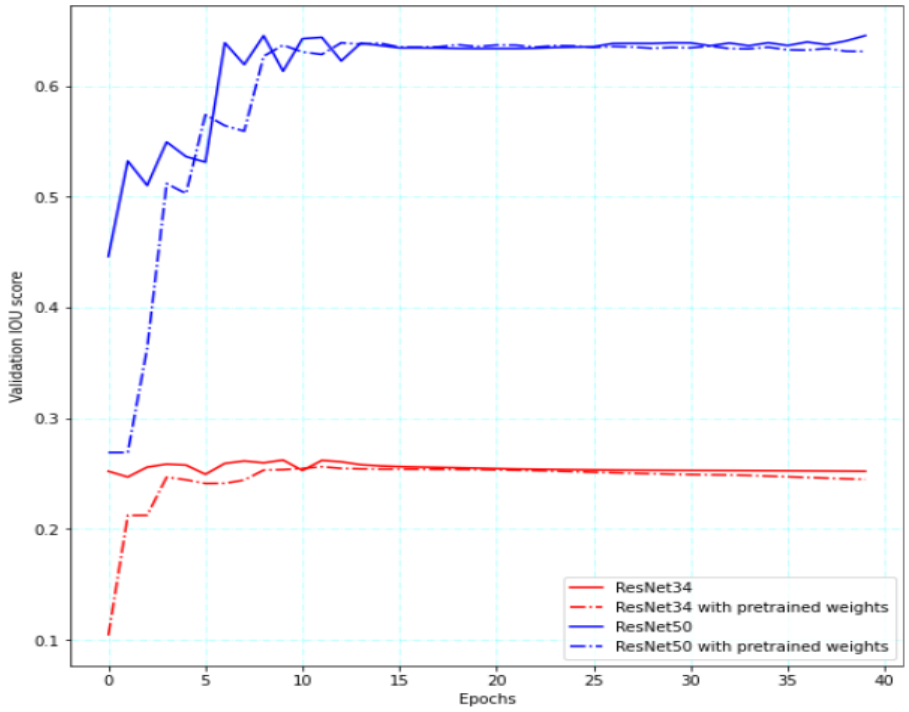

Figure 22. Graph illustrating validation IOU scores for different ResNet models as encoder.

5.8 Prediction Of Models

The predictions made by the U-Net model and the ResNet50 model is shown in Figure 24 below. The depicted MRI slice is from patient number 211, axial slice number 72 from the testing dataset. According to the predictions, our models perform exceptionally well in terms of tumor core.

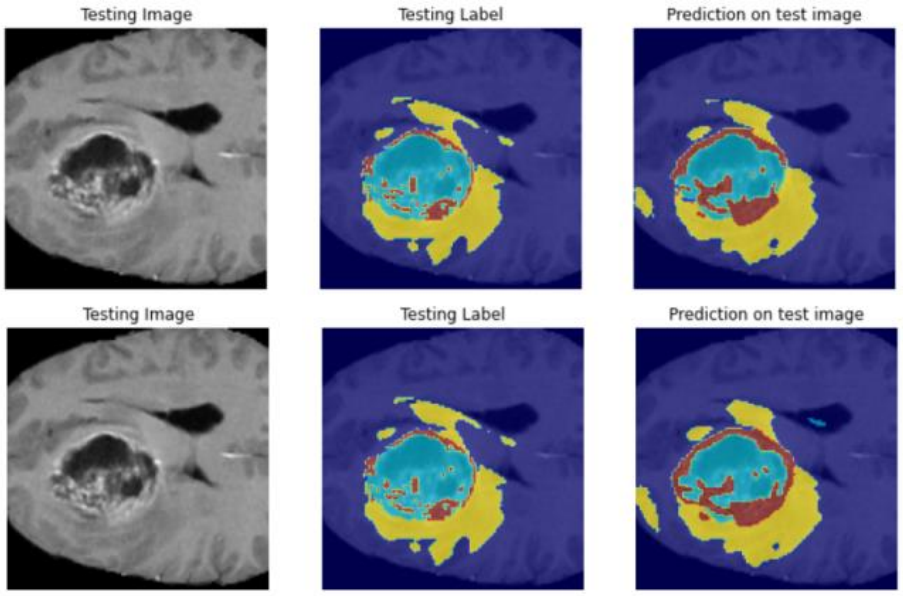

Figure 23. Figure depicting the test FLAIR MR image, the ground truth and the predicted segmentation mask. Top row are results by U-Net and bottom row by ResNet50 as encoder. Tumor segmentation contains necrotic core (blue), edema (yellow) and enhancing tumor (orange).

5.9 Model Evaluation and Comparison

Using the testing dataset, we conduct a final evaluation of the top-performing models. The results are shown in the table 7. As can be seen, the U-Net model with the weighted loss function performs the best, and while the mean IOU score is comparable to that of the U-Net model trained purely on dice loss, the IOU score for minority classes such as necrotic core is significantly improved from 0.5490 to 0.6145.

Model	Mean IOU	Non tumor	Edema	Necrotic	Enhancing
U-Net with dice only	0.7115	0.9886	0.6354	0.5490	0.6734
U-Net with weighted loss	0.7309	0.9898	0.6482	0.6145	0.6713
ResNet50 without pretrained weights	0.6686	0.9852	0.5742	0.4897	0.6255

Table 7. Models used for evaluation, the evaluated mean IOU score followed by IOU scores for each class.

5.10 Graphical User Interface

A web-application has been developed using the Flask micro-web framework in Python. The application is based on a RESTful API that enables simple HTTP requests to access our resources. It is a two-webpage application that is fairly straightforward. The first web page is the homepage where the NumPy file containing the preprocessed MRI scans can be uploaded using the POST method upon clicking the "Predict" button.

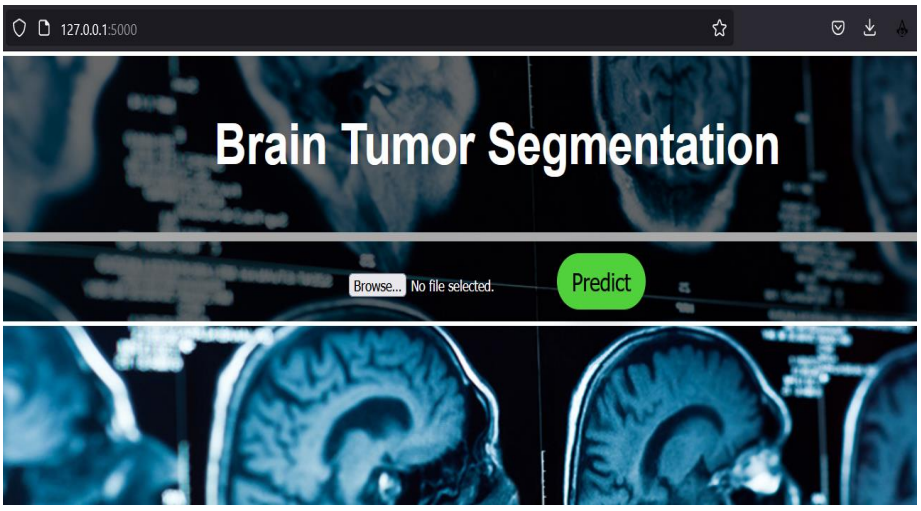

Figure 24. Screenshot of the homepage. Browse button does file uploading and predict button submits the file for prediction.

Now there is a delay as the model is loaded and predictions are. Each segmentation map layer is persisted in system memory. After completing the predictions, the user is

redirected to a second web page as shown in Figure 11 where all 128 layers of the brain MRI scan can be viewed using a slider that allows the user to view the desired slice by adjusting the slider. For each slice, the FLAIR MR image and the segmentation mask prediction are displayed.

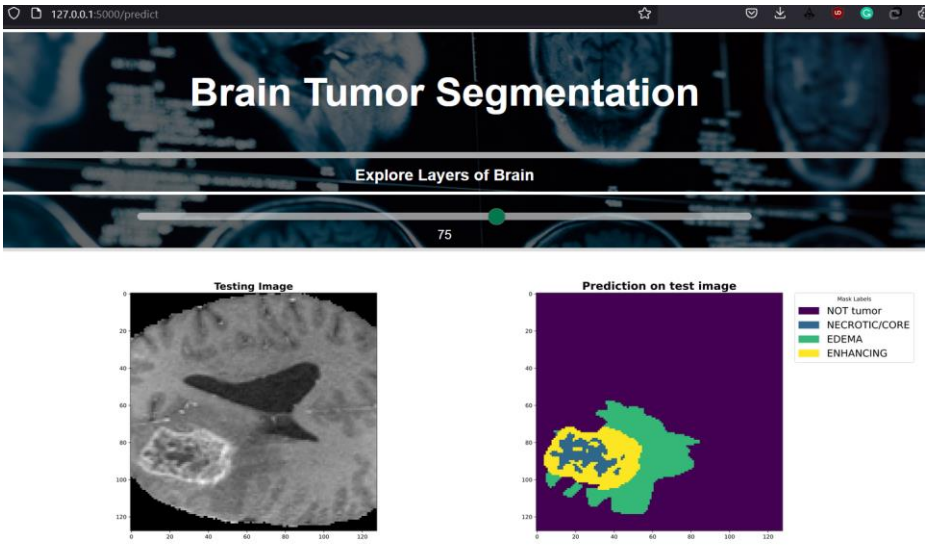

Figure 25. Webpage showing the input FLAIR image and the predicted segmentation mask with color coded legend. The slider controls the slice of brain viewed in this webpage.

6 Discussion

This dissertation explores the U-Net architecture, the variations that can be introduced by combining other popular FCN architectures with U-Net and the influence of different loss functions on the training of the U-Net. This study also investigates the effect of transfer learning on training the U-Net with modified encoders in terms of performance.

Due to limited computational resources available during training the model, the time spent on training has been significantly higher. There are numerous regularization techniques and hyperparameters that can be tested and tuned when training these networks. Each new regularization technique and hyperparameter requires the training of a new network. A more rigorous approach would be to try all combinations of regularization techniques and hyperparameters, but this quickly becomes impractical due to the time-consuming task of training a network and the combinatorial explosion of combinations of regularization techniques and hyperparameters.

The experimental setup for the U-Net was to add batch-normalization layers before performing non-linear activations. It is unknown whether moving same layers to after non-linearity would yield different outcome. In addition, four convolution blocks are used in the encoder path of U-Net, which produced better results than five, but adding or removing layers has not been tested.

Due to patient confidentiality, the datasets for medical images in general and brain tumor images in particular are relatively small. A larger dataset means more examples from which to learn, which improves the network's performance. The MICCAI BraTS dataset is one of the largest brain tumor datasets, but it is still relatively small in terms of deep learning. The dataset serves as a benchmark for the segmentation of brain tumors. By incorporating additional brain tumor datasets, a larger dataset might have been procured for training the network, which would likely have improved its performance.

The batch size used for this project was 1, as GPU memory was limited. A larger batch size could have led to different outcomes. If additional computational resources were available, experiments could be conducted to determine how batch size affects the training and performance of networks.

ResNet50 and VGG19 as the modified U-Net encoder outperformed ResNet34 and VGG16 respectively, demonstrating that deeper neural networks with greater capacity to learn the training data can handle such challenging classification tasks more effectively. Also, the pretrained "ImageNet" weights are not significantly superior to random weights, as the deeper neural network models do not appear to benefit from the transfer learning. Consequently, dedicated medical pretrained networks are required.

During the acquisition of T1ce MR images, a contrast fluid containing gadolinium is injected into the patient to enhance the contrast of the tumor and blood vessels. In some instances, the network segments these high-contrast blood vessels as tumors based on the predicted segmentations. Thus, it appears that the networks learn that areas with high contrast are associated with tumors. To address the problem of segmenting high contrast non-tumor tissue as tumor tissue, one may be able to perform image preprocessing in the vessel regions. These regions are situated in the same area of the head, making this preprocessing theoretically possible. By decreasing the image contrast in these regions, better segmentation results may be obtained.

A more nuanced and computationally inexpensive approach would be to only attempt to segment the classes that are clearly visible inside a given modality, but the authors lack sufficient domain knowledge and therefore do not take this approach.

Limitations and Future Work

The application developed for this dissertation requires knowledge of Python programming and is not intended for use in a clinical environment. The obtained segmentation results are promising, but not on par with those of an experienced radiologist.

A potential next step for our programme could be the expansion of the user interface to allow physicians to also edit the segmentations.

Further complicating the clinical application of the proposed architectures is the fact that just one or a few of the four modalities utilized in this dissertation are captured when patients are screened. Therefore, the implementations presented in this dissertation cannot be utilized without data synthesis or re-training of said network on a particular modality.

Also, the network struggles with finer details in tumor borders; this could be remedied by adding additional weight to the border pixels of the various ground truth clas-

ses. Adding additional weight to border pixels would penalize the network for incorrectly classifying border pixels, which would hopefully motivate it to complete this task with greater success.

In some cases, when examining the predicted segmentation results, only one or a few pixels are classified as tumors when there is no tumor in the ground truth. In some instances, these false positives also appear dispersed. In cases of brain tumors, a few pixels dispersed across the brain are extremely rare. These false positives could be eliminated through image post-processing. This could be accomplished by establishing a minimum number of pixels that must be present and have boundaries with one another. This simple postprocessing step would further improve the segmentation performance.

7 Conclusion

Segmentation of brain tumors is an important aspect of medical image analysis. For supervised learning, deep learning techniques for brain tumor segmentation typically require voluminous amounts of annotated ground truth data. Typically, tumor annotation requires medical professionals and is time-intensive. To overcome this problem, this dissertation explores the U-Net architecture and employs different techniques to handle the class imbalance problem.

Cropping the MR images with tumor in the center reduces the number of pixels, thereby accelerated training whilst also maintaining competitive segmentation outcomes because of higher number of tumorous labelled pixels.

Adding weights to the imbalanced classes during training was also investigated as a means of improving the segmentation outcomes. Experiments conducted on loss functions revealed that our weighted loss function performed 12% better than dice loss for the class with the fewest available samples, necrotic core class. The evaluation of the model revealed that the mean IOU score is 0.7309, indicating an outstanding performance.

Acknowledgments

I would like to thank Dr. Sergiy Bogomolov and Paulius Stankaitis for supervising my project and providing invaluable guidance throughout its duration. Also, thanks to family and friends for their continued understanding and support.

References

1. DeAngelis LM (2001) Brain Tumors. *N Engl J Med* 344: 114–123.
2. Miller KD, Ostrom QT, Kruchko C, et al. (2021) Brain and other central nervous system tumor statistics, 2021. *CA Cancer J Clin* 71: 381–406.
3. Goodenberger ML, Jenkins RB (2012) Genetics of adult glioma. *Cancer Genet* 205: 613–621.

4. Kleihues P, Burger PC, Scheithauer BW (1993) The New WHO Classification of Brain Tumours. *Brain Pathol* 3: 255–268.
5. Minaee S, Boykov Y, Porikli F, et al. (2022) Image Segmentation Using Deep Learning: A Survey. *IEEE Trans Pattern Anal Mach Intell* 44: 3523–3542.
6. Tiwari A, Srivastava S, Pant M (2020) Brain tumor segmentation and classification from magnetic resonance images: Review of selected methods from 2014 to 2019. *Pattern Recognit Lett* 131: 244–260.
7. Bauer S, Wiest R, Nolte L-P, et al. (2013) A survey of MRI-based medical image analysis for brain tumor studies. *Phys Med Biol* 58: R97–R129.
8. Liu J, Li M, Wang J, et al. (2014) A survey of MRI-based brain tumor segmentation methods. *Tsinghua Sci Technol* 19: 578–595.
9. Zeineldin RA, Karar ME, Coburger J, et al. (2020) DeepSeg: deep neural network framework for automatic brain tumor segmentation using magnetic resonance FLAIR images. *Int J Comput Assist Radiol Surg* 15: 909–920.
10. Ghosh A, Thakur S (2022) Review of Brain Tumor MRI Image Segmentation Methods for BraTS Challenge Dataset, *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 405–410.
11. Menze BH, Jakab A, Bauer S, et al. (2015) The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Trans Med Imaging* 34: 1993–2024.
12. Haenssle HA, Fink C, Schneiderbauer R, et al. (2018) Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Ann Oncol* 29: 1836–1842.
13. Ronneberger O, Fischer P, Brox T (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation.
14. Bakas S, Akbari H, Sotiras A, et al. (2017) Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Sci Data* 4: 170117.
15. Bakas S, Reyes M, Jakab A, et al. (2019) Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge.
16. Percy C, Holten V van, Muir CS, et al. (1990) International classification of diseases for oncology / editors, Constance Percy, Valerie Van Holten, Calum Muir. Portuguese and Spanish versions (1ed) issued as: Publicación científica / Organización Panamericana de la Salud; no. 345.
17. Louis DN, Ohgaki H, Wiestler OD, et al. (2007) The 2007 WHO classification of tumours of the central nervous system. *Acta Neuropathol (Berl)* 114: 547.
18. Louis DN, Perry A, Reifenberger G, et al. (2016) The 2016 World Health Organization Classification of Tumors of the Central Nervous System: a summary. *Acta Neuropathol (Berl)* 131: 803–820.
19. Glorot X, Bordes A, Bengio Y (2011) Deep Sparse Rectifier Neural Networks, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 315–323.

20. Xu B, Wang N, Chen T, et al. (2015) Empirical Evaluation of Rectified Activations in Convolutional Network.
21. Yi-de M, Qing L, Zhi-bai Q (2004) Automated image segmentation using improved PCNN model based on cross-entropy, *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, 743–746.
22. Lin T-Y, Goyal P, Girshick R, et al. (2018) Focal Loss for Dense Object Detection.
23. Sudre CH, Li W, Vercauteren T, et al. (2017) Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations, 240–248.
24. LeCun Y, Haffner P, Bottou L, et al. (1999) Object Recognition with Gradient-Based Learning, In: Forsyth DA, Mundy JL, di Gesú V, et al. (Eds.), *Shape, Contour and Grouping in Computer Vision*, Berlin, Heidelberg, Springer, 319–345.
25. Whitney W (2014) English: A diagram of a convolution layer and a pooling layer applied to an image.
26. Ioffe S, Szegedy C (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.
27. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning, Cambridge, Massachusetts, The MIT Press.
28. He K, Zhang X, Ren S, et al. (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.
29. Keras: the Python deep learning API Available from: https://keras.io/.
30. Kingma DP, Ba J (2017) Adam: A Method for Stochastic Optimization.
31. Havaei M, Davy A, Warde-Farley D, et al. (2017) Brain tumor segmentation with Deep Neural Networks. *Med Image Anal* 35: 18–31.
32. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440.
33. Larobina M, Murino L (2014) Medical Image File Formats. *J Digit Imaging* 27: 200–206.
34. Juntu J, Sijbers J, Dyck D, et al. (2005) Bias Field Correction for MRI Images, 543–551.
35. Tustison NJ, Avants BB, Cook PA, et al. (2010) N4ITK: Improved N3 Bias Correction. *IEEE Trans Med Imaging* 29: 1310–1320.
36. Srivastava N, Hinton G, Krizhevsky A, et al. (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15: 1929–1958.
37. He K, Zhang X, Ren S, et al. (2015) Deep Residual Learning for Image Recognition.