

School of Computing

FACULTY OF ENGINEERING



UNIVERSITY OF LEEDS

Full Title of Project

Tao Wang

**Submitted in accordance with the requirements for the degree of
MSc in Advanced Computer Science (Artificial Intelligence)**

Session 2019/2020

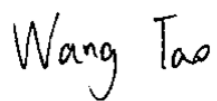
The candidate confirms that the following have been submitted:

Items	Format	Recipient(s) and Date
<i>Project Report</i>	<i>Report</i>	<i>SSO (25/08/20)</i>
<i>Code source</i>	<i>User manuals</i>	<i>SSO (25/08/20)</i>

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) 

Summary

This project covers design, implementation and evaluation of a neural network model which has ability to automatically segment brain tumor in multi-modal Magnetic Resonance Images. In this paper, firstly, the analysis and purpose of this project will be explained. Secondly, the survey with the relevant topic as this project will be analyzed, and then the techniques used in this study and the advantages of them and reasons why they are chosen will be shown. Next, the process of designing this system will be presented, mainly including data preprocessing and model building. Subsequently, I will give the details of model implementation and the code of it. Moreover, the evaluation metrics and the performance of this model will be displayed. At last, I will make a conclusion for this project, and describe my novel idea about segmenting brain tumor in in multi-modal Magnetic Resonance Images, and what I will do in the future.

Acknowledgements

Firstly, I would like to express my big thanks to my supervisor Ali Gooya, because he gives me this chance to complete this project and provides me with much guidance.

Also, I am grateful to Ammar Alsalka for his support so that I can know many details about how to complete my dissertation.

Finally, I would like to express my appreciation to my processor Dr Toni Lassila who spends time listening to my description of my Msc project.

Table of Contents

Summary	iii
Acknowledgements	iv
Table of Contents.....	v
Table of Figures	viii
Chapter 1 Introduction.....	1
1.1 Understanding the problem.....	1
1.2 The Project Aim.....	2
1.3 Objectives.....	3
1.4 Deliverables.....	3
1.5 Report Structure.....	4
Chapter 2 Background Research.....	5
2.1 Literature Survey.....	5
2.2 Overview of Important Background Information.....	6
2.2.1 Glioma.....	6
2.2.2 Magnetic Resonance Imaging (MRI).....	6
2.2.3 Deep Learning.....	7
2.2.4 Combination of Medical Imaging and Artificial Intelligence.....	7
2.2.5 Brats.....	8
2.2.6 Dataset.....	8
2.3 Methods and Techniques.....	9
2.3.1 Convolutional Neural Network (CNN)	9
2.3.2 Fully Convolutional Network.....	10
2.3.3 Back Propagation.....	10
2.3.4 Dropout.....	11
2.3.5 Batch Normalization.....	11
2.3.6 Python and Tensorflow.....	11

2.3.7 Adam Algorithm.....	12
2.3.8 Soft-max Function.....	12
2.3.9 ReLu Function.....	13
2.3.10 Skip-connection.....	14
2.4 Choice of Methods.....	14
2.4.1 Neural Network.....	14
2.4.2 3D U-Net.....	16
2.4.3 Choice Reasons of 3D U-Net.....	17
2.5 Summary.....	19
Chapter 3 Datasets and Experimental Design.....	20
3.1 Software Requirements.....	20
3.2 System Design.....	21
3.2.1 Initial analysis phase.....	21
3.2.2 Building Model.....	22
3.2.2.1 Encoder-Decoder.....	22
3.2.2.2 Encoding Path.....	23
3.2.2.3 Decoding Path.....	23
3.2.2.4 Shortcut.....	23
3.2.2.5 Layer Explaining.....	25
3.2.2.6 Three Modes of Convolution.....	26
3.3 Data Preprocessing.....	28
3.3.1 Cropping.....	29
3.3.2 Normalizing.....	30
3.3.3 Insert Slices and Patch Block Processing.....	31
3.3.4 Pickle.....	31
3.4 Summary.....	32
Chapter 4 System Implementation.....	33
4.1 Data Preparation.....	33
4.2 Model Building.....	35
4.3 Training Procedure.....	36

4.4 Testing Procedure.....	36
4.5 Implementation Details of System.....	37
4.6 Summary.....	41
Chapter 5 System Evaluation.....	42
5.1 Evaluation Metrics.....	42
5.1.1 Dice Coefficient (DSC).....	42
5.1.2 Hausdorff Distance (HD95).....	43
5.1.3 Intersection over Union (IoU).....	44
5.2 Evaluation of Generated Image vs. Truth Image.....	45
5.3 Evaluation from DSC and HD95.....	45
5.4 3D U-Net Performance vs. state-of-the-art Method.....	46
5.5 Summary.....	46
Chapter 6 Conclusion and Future Work.....	47
6.1 Conclusion.....	47
6.2 Future Work.....	48
6.3 Challenge.....	49
6.4 Personal Reflection.....	49
List of Reference.....	51

Table of Figures

Figure 1.1 sub-regions of glioma	3
Figure 2.1 the process of implementing FCN	10
Figure 2.2 the principle of soft-max function.....	13
Figure 2.3 the structure of simple neural network.....	15
Figure 2.4 Machine Learning vs. Deep Learning.....	15
Figure 2.5 The framework of cancer metastases detection.....	18
Figure 2.6 Volumetric Segmentation with the 3D U-Net.....	19
Figure 3.1 architecture of 3D U-Net.....	24
Figure 3.2 3D Convolution Layer.....	25
Figure 3.3 valid convolution.....	26
Figure 3.4 same convolution.....	27
Figure 3.5 full convolution.....	27
Figure 3.6 Maxing Pooling Layer.....	28
Figure 3.7 Up-Convolution Layer.....	28
Figure 3.8 four modalities of MRI.....	29
Figure 3.9 original image vs. normalized image.....	30
Figure 4.2 the segmentation process of 3D U-Net.....	38

List of Tables

Table 5.1 evaluation of 3D U-Net	47
Table 5.2 3D U-Net vs. state-of-the-art Methods.....	48

Chapter 1

Introduction

1.1 Understanding the problem

The task of medical image segmentation is usually defined as recognizing the outline or internal voxel set of brain tumors. It is the most common topic in deep learning researches applied to medical image analysis. The segmentation of organs and their sub-regions in medical images can be used to quantitatively analyze clinical parameters related to volume and shape, such as ventricular volume and contractile ejection rate of the heart. On the other hand, when using intelligent intensity-modulated radiotherapy technology to treat tumors, the delineation of the organ at risk is one of the very important steps in the formulation of a radiotherapy plan (Dan C et al., 2012). Deep learning is widely used in histopathological images and microscope image segmentation, or brain tissue structure segmentation and heart ventricular segmentation and other fields (Dan C et al., 2012).

There are thousands of people diagnosed with brain tumor in the world, in order to heal these patients, 3D magnetic resonance image is an important tool to analyze, treat and monitor tumor. However, it is really difficult for physician to process and analyze MR imaging data. Therefore, a robust deep learning-based technique is really urgent and essential. This technique can automatically segment tumors from magnetic resonance image, and compared with human experts, it will be more accurate and efficient. Using this technique, physician time will be saved, and it will provide an accurate reproducible solution for further tumor analysis and monitoring (Andriy, 2018).

A large number of researches have been proposed, and some of them get fruitful results, but the accuracy and performance of current technique are still not perfect. From the historical development process of deep learning methods, we can get that a sufficient amount of labeled data and appropriate model training methods are the key factor of successful use of deep learning (Xuan and Xiao-lin, 2019). As the extension of traditional machine learning algorithms, multiple layers and complexity are the significant features of deep learning model (Xuan and Xiao-lin, 2019). Although this kind of structure has a good representation ability, it also has many problems, such as network parameters, a large amount of data required for training, easy overfitting, etc, so this thesis is dedicated to creating a network with the function of segmenting brain tumor and trying to improve this technique to some extents. With the improvement of brain tumors segmentation algorithms, we will achieve extremely precise technique that can automatically segment multiple gliomas from multimodal MRI, which is the prerequisite of many clinical procedures (Yihui, 2018).

1.2 The Project Aim

The purpose of this project is to present a convolution neural network architecture that can automatically segment glioma in pre-operative MRI scans. In this paper, I will build and train a 3D u-net model which is able to automatically segment brain tumor from MRI into 3 sub-regions and try to reproduce the performance of 3D U-Net.

Figure 1 shows the three sub-regions of glioma, and the image patches from left to right are the whole tumor (yellow), the tumor core (red), and the enhancing tumor structures (light blue) (Brats, 2018).

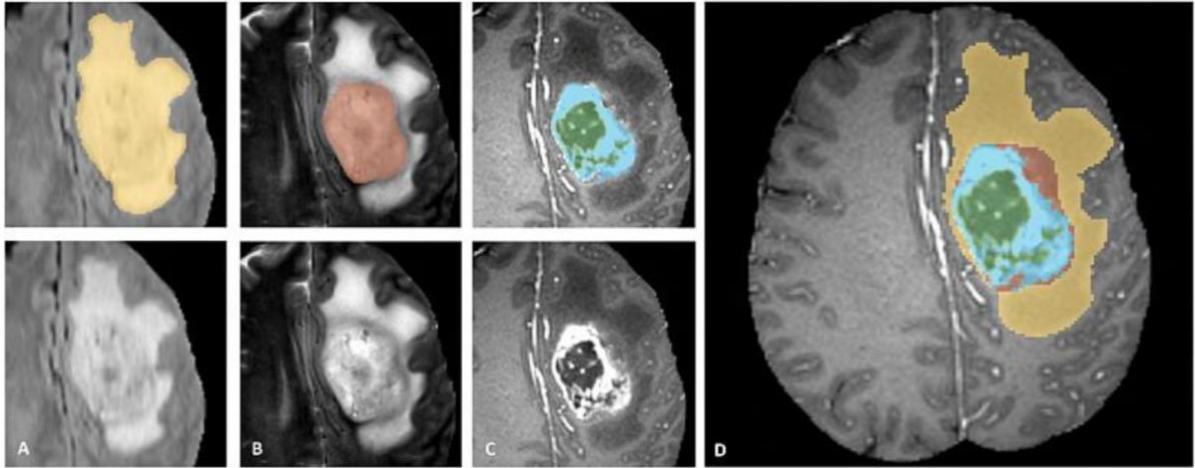


Figure 1.1 sub-regions of glioma (Brats, 2018)

1.3 Objectives

Objectives are essential for this project, and they are displayed below:

1. To analyze the problems of segmenting images
2. To find appropriate algorithms to deal with problems
3. To build a neural network architecture to segment glioma in pre-operative MRI scans
4. To process datasets
5. To train the network by training dataset
6. To validate the model using different image data
7. To evaluate the performance of this model by test data

1.4 Deliverables

After completing this project, a 3D u-net model which can be used to segment glioma in pre-operative MRI scans will be delivered.

Also, final project report will be submitted, and it includes the application of this model in some fields, and this report exhibits the process of creating this model in addition to the validation and evaluation results.

1.5 Report Structure

Chapter 1 introduces this system, and shows the context, aim, objectives and deliverables of this project.

Chapter 2 presents the researches with the similar topic as my project and describes the techniques used in the project, and the reasons why these techniques are chosen here.

Chapter 3 explains the social requirements of this project, and gives the details of building this project, including designing network, processing data.

Chapter 4 shows several codes of this system, and displays the process of implementing, and training this network.

Chapter 5 presents the evaluation and testing of this system

Chapter 6 makes the conclusion for this project and shows my novel idea that how to improve 3D U-Net and what I will do in the future.

Chapter 2

Background Research

2.1 Literature Survey

Segmenting brain tumors in multimodal MRI scans is one of the most challenging tasks in medical image analysis because of highly heterogeneous appearance and shape of it (Brats, 2018). As shown by Rupal and Mehul (2020) that they make the use of fully convolutional neural network, and this encoder-decoder architecture with three-layers deep is able to get the results of Dice Coefficient score 0.92, 0.90, and 0.79 respectively for whole tumor, tumor core, and enhancing tumor in training dataset, which is a very high performance. However, this method uses 2D slice as input, which means this model here is also 2D, but I focus on 3D model in my project, so I do not choose this method. Recently, most researches always focus on modifying 3D U-Net model to improve the performance of segmenting brain tumor. 3D U-Net was firstly proposed by Özgün et al. (2016). This network was born mainly to process some volumetric images, and the basic principle is not much different from U-Net architecture which has encoding path to analyze images and decoding path to make a full-resolution segmentation, and 3D U-Net just replaces 2D convolutions with 3D convolutions (Özgün, 2016). The model not only solves the problem of efficiency, but also requires only part of the slices in the data to be marked for the cutting of the block diagram (Özgün, 2016). The effect of 3D U-Net on the segmentation of a large number of medical images makes this semantic segmentation network architecture very popular. In recent years, 3D U-Net has also been seen everywhere in the championship schemes of some visual competitions. Thus, I think 3D U-Net is the most excellent model to segment medical imaging. Andriy (2018) who wins the first place in Brats 2018 also utilize the method like 3D U-Net, and the difference is that a VAE branch is added to the bottom of the down-sampling path to reconstruct the original image. I think VAE branch is the essence of Andriy's paper, but it is regrettable that I do not understand why this branch can improve the performance of his model, and the size of input image is $4 \times 160 \times 192 \times 128$, the memory of my GPU is not

sufficient to use so big input to train my model, so I do not put it into practice. In 2020, Po-Yu et al. modified 3D U-Net and DeepMedic and combined both two models and ensembled their segmentation predictions to get a high performance. Also, in their paper, they propose an state-of-the-art way that combines location information with the neural networks based on the latest patch to segment brain tumor (Po-Yu et al, 2020). Their method is a very good inspiration for me, but because of the limited time and equipment, I do not apply their idea in my project. At last, I decide to utilize original 3D U-Net to reproduce the performance of it in my project. I do not have so much time to compare the performance of each method, because it almost takes 24 hours to train the neural network model once. Therefore, I have to just choose a basic and crucial model to analyze and implement it.

2.2 Overview of Important Background Information

The overview of some important background information will be shown here, including glioma, MRI, deep learning, combination of Medical Imaging and Artificial Intelligence, Brats, and dataset.

2.2.1 Glioma

Glioma is the most common brain tumor in adults, and based on the invasiveness of glioma, it has two types which are high-grade and low-grade (Mahbubul et al., 2019). Low-grade gliomas always suggest that this patient has harmless tendencies and a better prognosis, but high-grade gliomas are uncontrollable and more aggressive, which means this kind of patients will have very limited survival time (Alberto et al., 2018). Depending on the tumor cells acuteness in MRI, gliomas can be classified into three kinds of sub-structures, whole tumor, the tumor core, and the enhancing tumor structures.

2.2.2 Magnetic Resonance Imaging (MRI)

MRI is a medical examination way, and it is the highest quality medical imaging for the recognition of soft tissue (Peter, 2018). Hydrogen nucleus signal collected by the computer system when putting human tissues, organs and lesion into the strong and uniform magnet space, which will be converted into an image by digital reconstruction technology, and this image is MRI (Judith, 2018).

2.2.3 Deep Learning

Deep learning is a new field based on traditional machine learning, and it is originated from the neural network inspired by the structure of human brain. Deep Neuron Networks (DNN) is a discriminative model which can use back propagation algorithm for training, weight update can be solved using method like stochastic gradient descent method (Jason, 2019). Deep neural network is the general term of deep learning, which contains various neural network, such as CNN, RNN, LSTM etc. (Jason, 2019). Currently medical imaging technology and computer hardware and software make a continuous progress, and the technique of medical image analysis also make a big development in order to contribute to clinical disease diagnosis and treatment. In recent years, deep learning, especially convolutional neural networks have rapidly become popular, and increasingly more researches focus on using deep learning to analyze medical image, which has become a mainstream research direction.

2.2.4 Combination of Medical Imaging and Artificial Intelligence

Utilizing Artificial Intelligence to analyze medical imaging is a relatively new research direction and industrial hot spot in the digital medical field. The analysis, recognition and labeling of medical images require the accumulation of long-term professional knowledge (Xiaoming, 2017). Doctors need to spend quite long time learning and mastering the relative knowledge. To many extents, the process of deep learning and doctors' learning are the similar. They both need to learn, understand and apply the massive knowledge.

However, artificial intelligence is able to be more efficient and accurate than professional doctors when detecting images (Xiaoming, 2017). The application of artificial intelligence in the medical field has become a trend. Applying artificial intelligence driven by big data to cancer diagnosis will undoubtedly provide patients with a front line of life (Xiaoming, 2017).

2.2.5 Brats

Although there are a large number of researches about segmentation of gliomas in pre-operative MRI scans, there is no unified standard to give a fair evaluation for the performance of these models. Brats is set to evaluate the novel strategies for the segmentation of brain tumors in multimodal magnetic resonance imaging (MRI) scans. What is more, before the appearance of Brats, algorithms always lack open datasets to train and test, and there were enormous differences between private datasets, which made compare different algorithms become difficult (Brats, 2018). Therefore, it is necessary to set a competition like Brats.

2.2.6 Dataset

Dataset in 2018 Brats has 285 cases, including 210 HGG and 75 LGG, all the images in the dataset have been segmented manually by experts, which will be the ground truth (labeled images) when training network (Rupal and Mehul, 2020). MRI of each case contains 4 modalities which are T1, slice resolution is 1-6 mm, T1ce, 3D resolution is 1 mm, T2, slice resolution is 2-6 mm, and FLAIR with a slice resolution of 2-6 mm (Brats, 2018). MRI has multi-modal characteristics, because single mode cannot fully subdivide the tumor in the relevant area, which will often cause failure or deficiency, multiple MRI modes are used to effectively make up for the above weaknesses (Wolfgang et al., 2018). Multi-modal images can effectively complement each other and can effectively improve the accuracy of segmentation, but it will also increase the difficulty of segmentation problems to some extent (Wolfgang et al., 2018). Although the image information input from multiple modes increases

the necessary information for segmentation, at the same time, a lot of unnecessary information will increase the difficulty of segmentation (Wolfgang et al., 2018).

2.3 Methods and Techniques

This project is written in Python, and the framework used here is tensorflow. As for the architecture of the model, I find DeepMedic and 3D U-Net both perform well about segmentation of brain tumor in multimodal MRI scans. Po-Yu et al.(2020) made use of two deep learning models which are DeepMedic and 3D U-Net respectively to segment glioma in pre-operative MRI scans. DeepMedic was proposed by Konstantinos et al. (2017) to solve the challenge of brain lesion segmentation, and this neural network architecture has a dual pathway, 11 three-dimensional Convolutional layers deep. I think the most excellent point of this model is that it utilizes dense-training method to solve the challenge of uneven distribution of segmentation targets in medical images that are often small. Moreover, 3D U-Net is 3D version of U-Net, which means 3D U-Net uses 3D convolutions, and the details of this network I will describe in chapter 3 (Po-Yu et al., 2020). In the end, I decide to use 3D U-Net in this study. 3D U-Net is a complicated structure which consists of multiple techniques, including Fully Convolutional Network, Back-propagation, Dropout, Batch Normalization, Soft-max function, Adam algorithm, skip-connection, etc.

2.3.1 Convolutional Neural Network (CNN)

CNN is different from conventional neural networks, not fully connected structure. The neurons in each layer of CNN are set as three dimensions: width, height, and depth. The convolution itself is a two-dimensional template, so width and height represent the size of convolution, and the depth here represents the number of channel, in addition the channel of input layer depends on the type of image, the number of other channels is based on convolution kernel number (Sumit, 2018). This structure can lessen the number of

parameters, such as weights and bias, reduce computational overhead, and prevent overfitting due to too many parameters, because convolution has the characteristics of "weight sharing" (Sumit, 2018).

2.3.2 Fully Convolutional Network (FCN)

Fully Convolution Network is always used for semantic segmentation, and semantic segmentation means we need to classify each pixel in one image (Sik-Ho, 2018). FCN can input any size of images, and when input passing through every convolution layer, deconvolution layer will be applied to up-sample the feature map in order restore the output of last convolution layer to the same size of the input image, and then FCN will have ability to make a prediction for each pixel, and make sure the spatial information in the output data is similar to the original input image at the same time, and finally up-sampled feature map will be classified pixel-by-pixel (Sik-Ho, 2018).

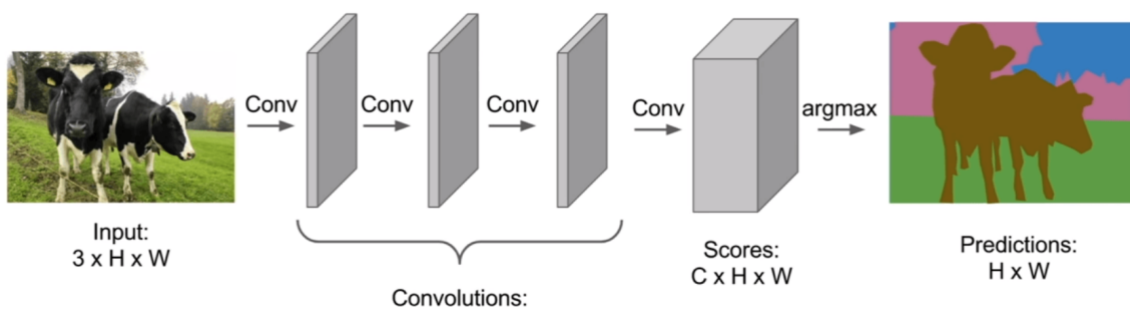


Figure 2.1 the process of implementing FCN (Sergios, 2019)

2.3.3 Back propagation

In my opinion, back-propagation is used to compute the partial derivative of weights and bias in the cost function, and neural network will update the parameters depending on it. If we

use traditional way compute the partial derivative, the neural network will consume too much time and resource. Anas (2019) claims that back propagation is a method to transfer total loss back to neural network in order to update the weight to reduce the weights by minimizing the weight in a way that brings higher errors to the node. Generally accepted by the industry that back-propagation is the essence of neural network, almost every kind of neural network will make use of it during the process of training.

2.3.4 Dropout

Dropout is a method to deal with the problem of overfitting. Dropout is performed during the process of training a neural network, it randomly ignores a part of layer neurons, which makes one network be treat as many different networks during the process of training because different nodes are used every time (Jason, 2019). Therefore, the output of the missing layer is randomly subsampled, it has the effect of reducing capacity or making the network thinner during training, and it can solve the problem of overfitting effectively (Jason, 2019).

2.3.5 Batch Normalization (BN)

Batch normalization is used to standardize the inputs when training a very deep neural network, and this has a big effect on stabilizing the learning process and allows fewer number of training epochs to train a deep network (Jason, 2019). In conclusion, BN is a method to unify the scatted data and optimize the neural network, and it is always set between fully connected layer and activation layer.

2.3.6 Python and Tensorflow

Python is very popular and simple programming languages in the world, it mainly has three applications, Web Development, machine learning, data analysis and visualization, and

Scripting (YK, 2018). The application of Python in this study is certain to be machine learning, and I utilize framework of tensorflow which is created by the Google Brain team, and it is the most popular open source machine learning framework in the world because it is suitable for numerical computation and large-scale machine learning (Serdar, 2019). Python is used by tensorflow to provide a convenient front-end API for building applications (Serdar, 2019). A large number of algorithms are encapsulated by tensorflow so that users can call these algorithms to build neural network or finish other applications straightforwardly, and users can just use several lines of code to build a model.

2.3.7 Adam Algorithm

Adam is an algorithm for performing the first derivative optimization on a random objective function, which is based on adaptive low-order moment estimation, and it is straightforward to accomplish this algorithm, and it just needs lower memory to get very high computational efficiency (Jason, 2017). It is extremely effective on sparse gradients and has excellent performance on non-steady state and online problems (Jason, 2017).

2.3.8 Soft-max Function

Soft-max is used to make a classification for multiple different types, it can predict the probability of this input belonging to each classification and give the range from 0 to 1, for example, the soft-max layer has 3 neurons which represents 3 types and the value of each neuron is the classification possibility (Uniqtech, 2018). It obtains the exponent of each output, and then normalizes the sum of these exponents to each number, thereby converting logit into probability (Uniqtech, 2018).

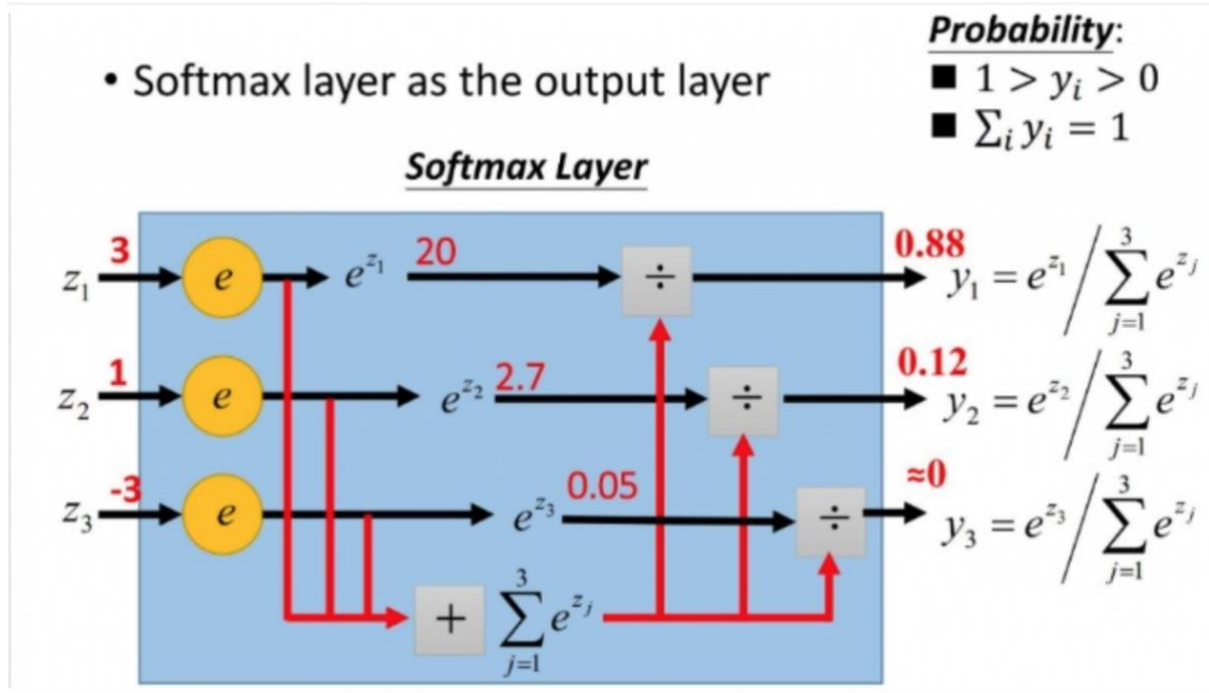


Figure 2.2 the principle of soft-max function

2.3.9 ReLU function

Rectified Linear Unit (ReLU) is one of activation functions which is set between two layers, after passing the value of neurons in the previous layer through ReLU, and then the output will be the input of the next layer (Dishashree, 2020).

ReLU function is defined as:

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

The advantages of ReLU are that, for linear functions, ReLU has stronger expressive power, especially in deep networks, and for nonlinear functions, ReLU does not have the problem of gradient disappearance because the gradient of this activation function will not become less with the increase of layers in neural network, which can keep the convergence rate of the model in a stable state (Sagar, 2017).

2.3.10 Skip-connection

Skip-connection is widely used in convolutional architectures. It provides an additional path for gradient with back propagation, and this path conducive to the convergence of network (Nikolas, 2020). The principle of skip-connection is that it skips some layers and inputs the output of one layer to another layer (Nikolas, 2020). This method can solve the problems of gradient vanish effectively.

2.4 Choice of Methods

2.4.1 Neural Network

Neural networks are derived from machine learning algorithms, and due to the powerful ability, it gradually replaces the position of traditional machine learning technique. It is inspired by human brain or other biological systems, and creates a mathematical model depending on the structure and function of these systems and uses various functions to imitate the process of implementation (Larry, 2017). Neural network consists of a large number of neurons, and the implementation process of neural network is just the calculation process of neurons. In the most situations, the structure and function of artificial neural can be changed by itself based on the types of input, so it is an adaptive system (Larry, 2017). Current neural network is a non-linear statistical data modeling tool, and typical neural network is made up of three parts, architecture, activation function, and learning rate, where architecture provides the parameters of the network and give the relationship of adjacent nodes, and activation function is used to add nonlinear factor, for example, if no linear line can classify 4 points, activation function can be used to find a curve or circle to classify them, and learning rule defines the step size of each update of weights or bias (Larry, 2017). In general, a neural network contains input layer, hidden layer and output layer.

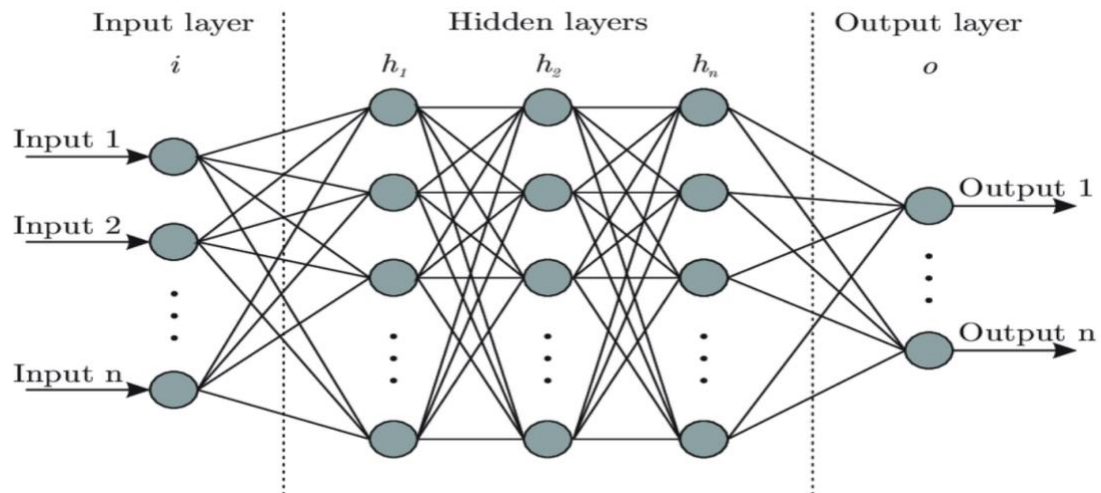


Figure 2.3 the structure of simple neural network (Lavanya, 2019)

Compared with traditional machine learning technique, I think the biggest advantage of deep learning is that deep learning algorithms can extract features of objects automatically, while machine learning always require expert to spend much time identifying features of objects manually (Sambit, 2018).

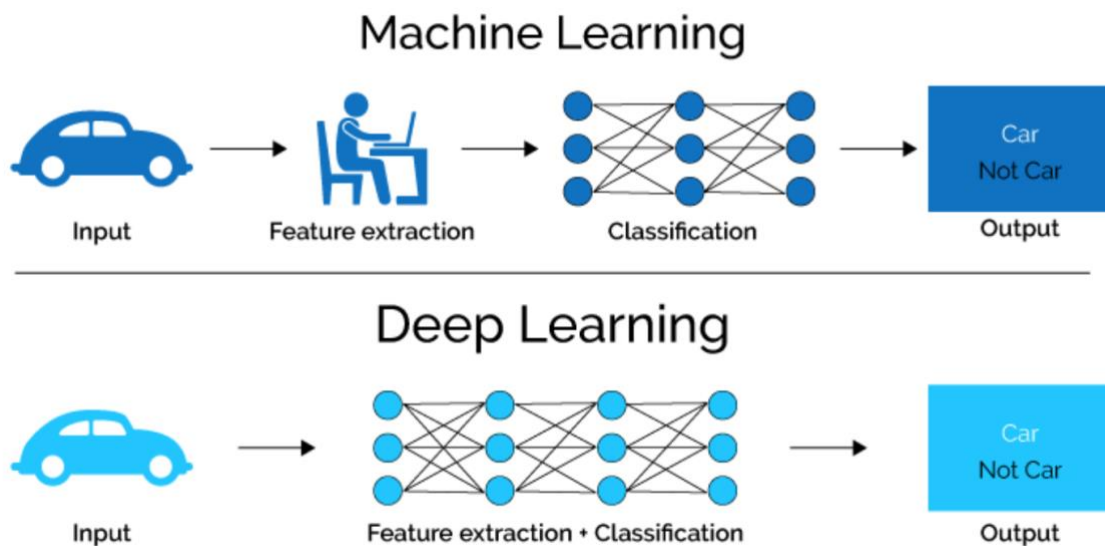


Figure 2.4 Machine Learning vs. Deep Learning (Sambit, 2018)

Also, deep learning is an end to end problem solving approach, in contrast, machine learning needs to break down one problem into several parts, solve them in order, and combine the results in the end (Sambit, 2018). For example, in this study, we just need input the MRI of

brain tumor and labels into deep learning model, and this model will provide the 3 sub-regions at output, but in traditional machine learning algorithms, multiple complicated steps need to be pre-prepared for final output. Therefore, compared with machine learning techniques, deep learning algorithms should be a better choice in this study.

2.4.2 3D U-Net

This study will focus on 3D U-Net, it is a variant of FCN, but it is different from FCN, firstly the structure of U-Net is completely symmetrical, which means the left side and the right side are very similar, while the FCN decoder is relatively simple, using only a deconvolution operation, and there is no convolution structure in decoder. Also, 3D U-Net has a feature called skip-connection, while FCN uses summation instead of skip-connection. Obviously, U-Net is a very popular model in the field of medical imaging segmentation. This method was proposed at the MICCAI conference in 2015 and has been cited more than 4000 times. The original purpose of U-Net is to solve the problem of biomedical images (Olaf, 2015).

Because of the excellent performance, it cannot only be used in medical imaging segmentation, but also it has been widely used in other semantic segmentation tasks, such as satellite image segmentation. I think the biggest features of 3D U-Net are U-shaped structure and skip-connection. The encoding path of U-Net performs 4 times of down-sampling, and the feature map is the one sixteenth of the original image. Symmetrically, the decoding path also up-sample 4 times, and it restores the high-level semantic feature map to the resolution of the original image (Sik, 2019). Skip connection is the main point in 3D U-Net. This structure use skip connection at the same stage, which ensures that the feature maps finally obtained will incorporates more low-level features and makes the feature maps from different scale incorporate so that this model can perform multi-scale prediction and Deep-Supervision, and 4 times up-sampling can also make the segmentation image recover more refined information such as edges (Sik, 2019).

The other techniques that I mentioned in section 2.2, fully convolutional network, back-propagation, dropout, Adam algorithm, soft-max function and batch normalization all play

vital roles in 3D U-Net. Convolutional networks are used to extract feature maps in the model, and back-propagation is used to compute the partial derivative of parameters of network in order to use Adam algorithm to update the them during the process of training, and dropout is used to avoid overfitting, batch normalization can accelerate the convergence speed and optimize the model, soft-max function is used to calculate the error from the prediction value to the labeled sample, and ReLU function is set to reduce the interdependence of parameters and alleviate the occurrence of over-fitting problems.

2.4.3 Choice Reasons of 3D U-Net

As for the reasons that I select 3D U-Net, firstly, as shown above, many researches have used 3D U-Net to segment brain tumor in pre-operative MRI scans and gotten high accuracy and excellent performance.

Also, 3D U-Net is really suitable to segment medical imaging. With regard to medical imaging, this kind of images always possess relatively simple image semantics and relatively fixed the structure because we just need the imaging of a fixed organ, not the whole body. For example, we need to segment brain tumor, the medical imaging should certainly be brain MRI (Sik, 2019). Both high-level semantic information and low-level features are important due to the fixed structure of the organ and the simply semantic information. U-shaped structure and skip-connection of 3D U-Net can solve this kind of problems perfectly, so this is the key point why I choose 3D U-Net.

Another reason is that it is relatively difficult to obtain medical imaging data, so the dataset is limited, which cause the model should not be too big, if the number of parameters is too high, overfitting will happen, although I apply dropout here. The number of parameters in original U-Net is roughly 28 MB, and if the number of channels can be reduced, the scale of this model will be smaller, so 3D U-Net is really suitable for brain tumors segmentation in multimodal MRI scans.

What is more, compared with natural images, medical images are different in that they have multiple modalities, such as Brats competition which provides us with multi-modal data flair, T1, T1CE, T2 and the details I will show below. 3D U-Net is able to extract features of different modalities.

Last but not least, interpretability is extremely crucial for medical images. Obviously, Medical imaging is ultimately to assist doctors in clinical diagnosis, so it is far from enough for the network to tell the doctor whether a 3D CT has a disease or not (Bolei, 2015). The doctor also wants to know which layer the lesion is on, and where on which layer (Bolei, 2015). 3D U-Net plays well in this point. We just need to display the feature map of 3D U-Net, doctors can get the information explicitly.

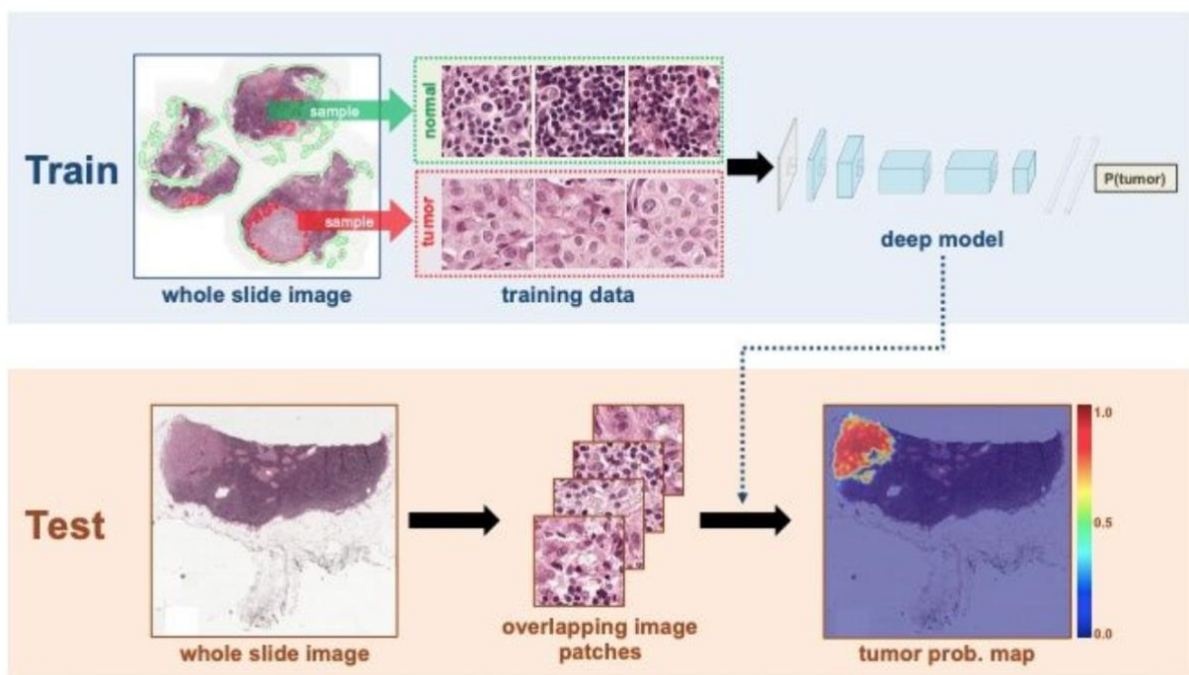


Figure 2.5 The framework of cancer metastases detection (Shawn, 2015)

Actually, 2D U-Net can also be used to segment biomedical images because this kind of images are often blocky, which means that a whole image is composed of many slices (Ozgun, 2016). However, there is a problem that a slice of a slice of a biomedical image, including training data and labeled data has to be sent to the designed model for training,

which will cause the efficiency to be low, and make images pre-processing become tedious (Ozgun, 2016). Thus, 3D U-Net is a better choice.

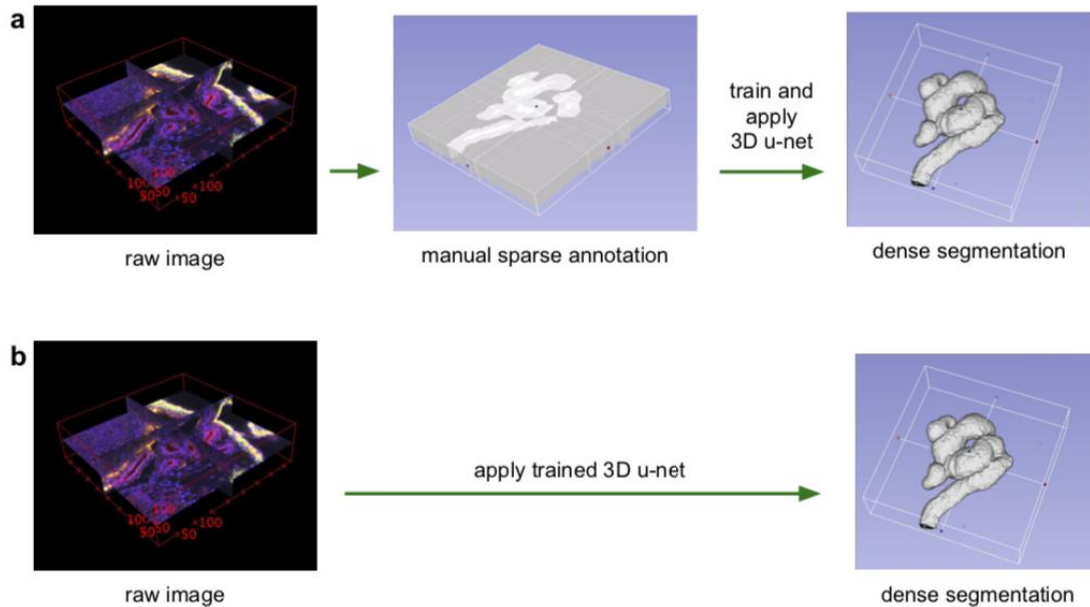


Figure 2.6 Volumetric Segmentation with the 3D U-Net (Ozgun, 2016)

In conclusion, the reasons why I choose 3D U-Net are that good performance in previous researches, skip-connection and U-shaped structure, limited dataset of medical imaging, good interpretability, and high efficiency.

2.5 Summary

This chapter shows the previous survey about segmenting brain tumor from MRI, and then describes the techniques used in this project. At last, I gives the reasons why I choose these techniques and the advantages of them.

Chapter 3

Datasets and Experimental Design

3.1 Software Requirements

In the field of medical imaging, when doctors or researchers perform quantitative analysis, real-time monitoring, and treatment planning of a specific internal tissue and organ, in order to make correct treatment decisions, they usually need to know some detailed information about this tissue and organ. Therefore, medical imaging has become a necessary tool to diagnose and treat diseases, and it plays increasingly significant role in this field. Magnetic resonance image is one of biomedical imaging techniques, and it has been widely used in clinical examination, diagnosis, treatment and decision-making currently. However, there are multiple difficulties when doctors are processing and analyzing MR imaging data. The current clinical situations are that, firstly, when experts face the areas where changes are not obvious, it is very difficult to distinguish different organizations. Secondly, there are big differences about the characteristics of tumor tissues, such as location and morphology. Also, multiple Magnetic Resonance images (MRI) modalities are available, and different characteristics of tissues can be captured by different modalities (Brats, 2018). Therefore, automated brain tumor segmentation technology is urgently required by the industry. How to make the most of deep learning methods to analyze and process these ultra-large-scale medical image big data and provide scientific methods for the screening, diagnosis, and efficacy evaluation of various major diseases in clinical medicine are main problems that urgently needs to be solved in the field of medical image analysis.

Deep learning is a data-driven analysis task, which can automatically learn related model features and data characteristics when inputting massive dataset of a specific problem during the training period. Different from manually designing models for specific problems, deep learning methods can implicitly and automatically learn medical image features directly

from data samples. The learning process is essentially an optimization problem solving process. Through learning, the model will recognize and get the accurate features of dataset so that it can make a correct predict for new dataset.

In this study, the model should have ability to segment the correct brain tumor from MRI after learning features of a large-scale dataset and can label the correct sub-regions of tumor in test dataset.

3.2 System Design

3.2.1 Initial analysis phase

In this phase, the literature about segmentation of brain tumor is reviewed comprehensively. Also, several deep learning models are analyzed, and comparing the performance and accuracy of these models.

In these proposed researches, we can get that the techniques of segmenting medical images are quite mature, and the accuracy rate of intelligent imaging diagnosis has surpassed clinicians in many diseases. Therefore, currently, most of researches are committed to improving the existing models so that these models can perform better. The principle is to build a multi-hidden-layer machine learning model, use massive sample data for training, learn more accurate features, and ultimately improve the accuracy of classification or prediction.

Michael et al. (1996) claimed that early pathological classification is usually divided into 3 steps. The first step is to manually label the target area in the image, the second step is to identify and classify the segmented area, and the third step is to make a macro judgment on the entire diagnosis result. With the continuous development of CNN, classifiers are becoming more and more powerful, and new algorithms can directly classify images and detect objects end-to-end, such as 3D U-Net.

Therefore, the challenge of this paper is to find and enhance an end-to-end model in order to improve the accuracy of segmenting brain tumor.

This phase is extremely time-consuming due to the massive amounts of researches, and I also need to spend a great deal of time analyzing and understanding these materials so that I can pinpoint approaches to the problem.

3.2.2 Building Model

This system model is designed as 3D U-Net. This network uses data augmentation methods, mainly by rotation, scaling, and setting the image to gray. At the same time, smooth dense deformation fields are used on the training data and the real labeled data, mainly by selecting a grid with a standard deviation of 4 from a sample of normally distributed random vectors, with a spacing of 32 voxels in each direction, and then applying B-spline interpolation (Sik-Ho, 2019). Subsequently, training the network, and the process of the training uses a weighted cross-entropy loss function to reduce the weight of the background and increase the weight of the labeled image data to achieve a balanced effect on the tubules and background voxels Loss (Viraj, 2019). This network has an encoding path and a decoding path, and each path has 4 resolution levels.

3.2.2.1 Encoder-Decoder

The structure of encoder-decoder was proposed by Hinton et al. (2016), at that time, the main function of encoder-decoder structure is not segmentation, but image compression and noise removal. The input is an image, and through the encoding of down-sampling, feature map that is smaller than original image will be gained, which is equivalent to compression, and then through decoding, which is used to recover feature map to original image, so we just need to store a feature and a decoder, when we want to save a image. In the same way, this idea can also be used to denoise the original image (Hinton et al, 2016). This way is to

manually add noise to the original image during the training phase, and then put it into decoder, then we can restore it to original image (Hinton et al, 2016). Later, this idea was used in the problem of image segmentation, which is the FCN or U-Net structure.

3.2.2.2 Encoding Path

The theoretical significance of encoding path is that it can make some small disturbances become robust, prevent overfitting, make the amount of calculation be less, and rise the size of the receptive field, but the main function of it is to extract feature map (Olaf, 2015).

Encoding path has 4 layers and each one comprises of two convolutions with the size of $3*3*3$, and after each convolution, there is an activation layer (ReLU), and then followed by a $2*2*2$ maximum pooling layer with stride of 2 (Olaf, 2015).

3.2.2.3 Decoding Path

The second half is decoding, that is, the process of using the previously encoded abstract features to restore the original image size, and finally get the segmentation result (Olaf, 2015). There are 4 layers in this path as well, each layer has a $2*2*2$ deconvolution layer with stride size of 2, followed by 2 convolution layers with size of $3*3*3$, and after convolutions, there is a ReLU layer (Olaf, 2015). The last layer is a $1*1*1$ convolution layer, which is used to lessen the number of channels so that the number of channels is the same as the number of labels (Sik-Ho, 2019). In the last layer, weighted soft-max loss function is set so that the network can be trained with sparsely annotated data (Sik-Ho, 2019). The weights of unlabeled pixels are set to zero by soft-max, so the network can learn from only labeled pixels and generalize to the entire stereo data.

3.2.2.4 Shortcut

However, when decoding to recover data, feature scale will change, and information will inevitably be lost. Note that there is concept of shortcut in this model, and shortcut is able to

pass the layer with the same resolution in the encoding path to the decoding path to provide it with original high-resolution features (Sik-Ho, 2019), and this method is called skip-connection, it is the structure of feature fusion, which means it can concatenate the low-level feature with high-level feature (Matthew and Rob, 2013). The underlying features are basic units, such as color, texture or edge contours, and as for high level features, these features are more abstract, which are more like a region (Matthew and Rob, 2013). Also, it supplements information, and allows model to rely on more information, and improves the problem of insufficient information during up-sampling, thereby improving the accuracy of segmentation (Matthew and Rob, 2013).

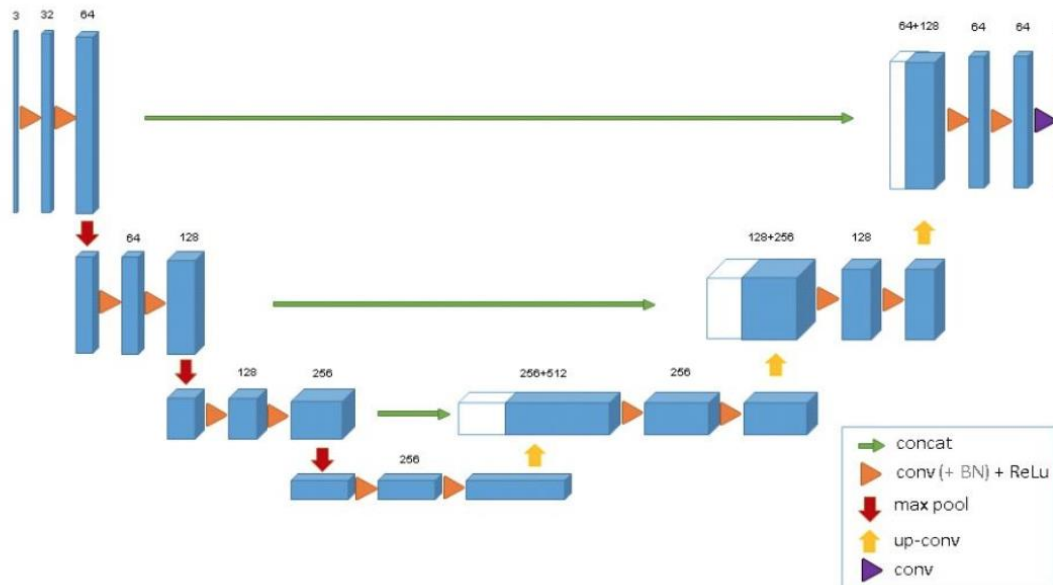


Figure 3.1 architecture of 3D U-Net (Sik-Ho, 2019)

Figure 3.2.2.1 displays the structure of my model clearly. The blue box represents the feature map, 3x3 convolution is displayed by the orange arrow, which is able to extract features of images, skip-connection is displayed by the green arrow, which is used to combine low-level and high-level features, the pooling layer is represented by red arrow, which is used to reduce dimensionality, the yellow arrow represents up-sampling, which can restore the feature map to original image, 1x1x1 convolution is shown by purple arrow, which is used to give the final segmentation image.

3.2.2.5 Layer Explaining

Convolutional layer in both encoding and decoding path is used to extract the feature map of images, and the deconvolutional layer in the decoding path can up-sample the feature map and restore the size of feature map to the original image. Max pooling layer is used to keep the main features while reducing the parameters and the feature map dimension in order to prevent overfitting and improve the generalization ability of the model.

What is more, between each 3D Convolutional layer and activation layer, I set batch normalization to unify the scatted data. Also, I set dropout between the second and third up-sampling and down-sampling layer respectively to reduce overfitting.

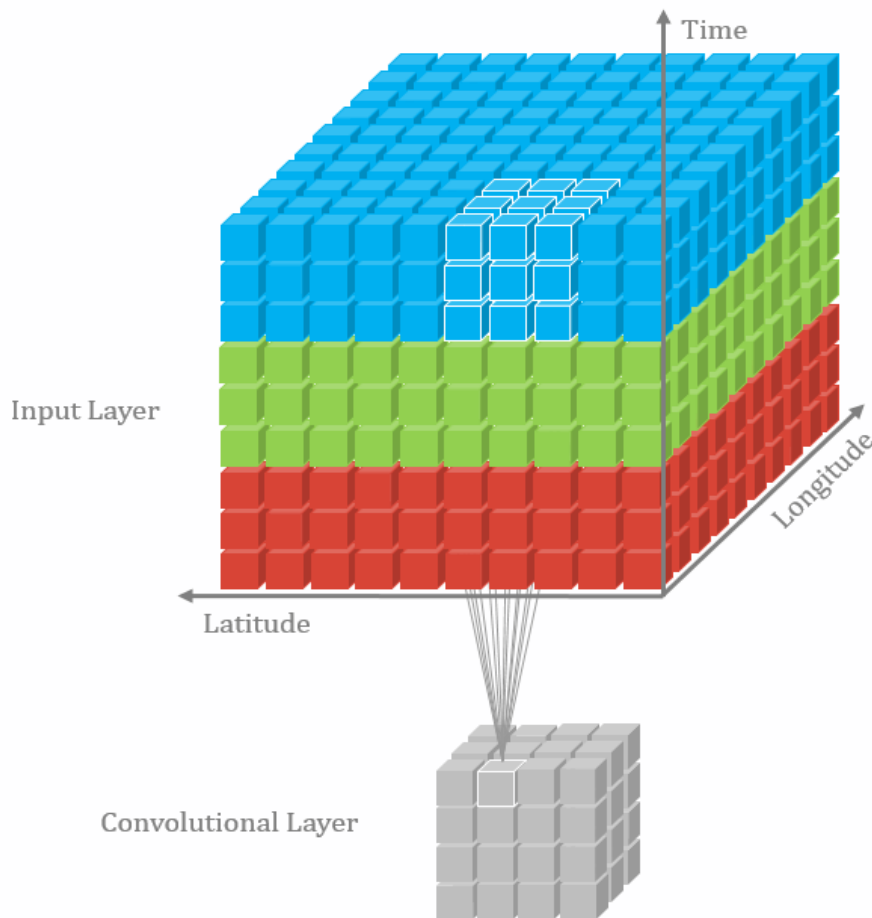


Figure 3.2 3D Convolution Layer (Resuly, 2018)

3.2.2.6 Three Modes of Convolution

There are 3 modes of padding about convolution layer, “valid”, “same”, and “full”.

“Valid” represents only using the valid part of each convolution, which means padding is 0. If image size is $D1 \times D1$, stride is S , the size of convolution kernel is $D2 \times D2$, and after passing through the convolution, the image size will be $((D1-D2)/S+1)^2$ (Olaf, 2015).

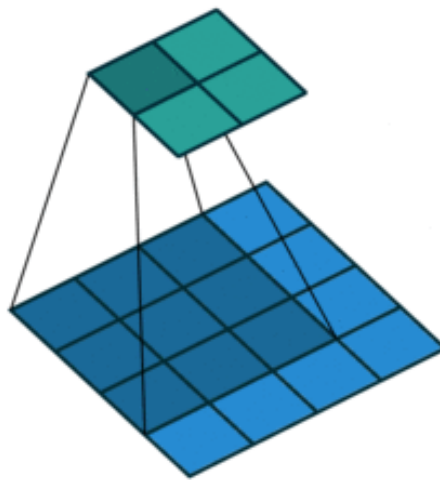


Figure 3.3 valid convolution (Jiulin, 2020)

“Same” means if image size is $D1 \times D1$, stride is S , the size of convolution kernel is $D2 \times D2$, and after passing through the convolution, the image size will still be $D1 \times D1$ (Olaf, 2015).

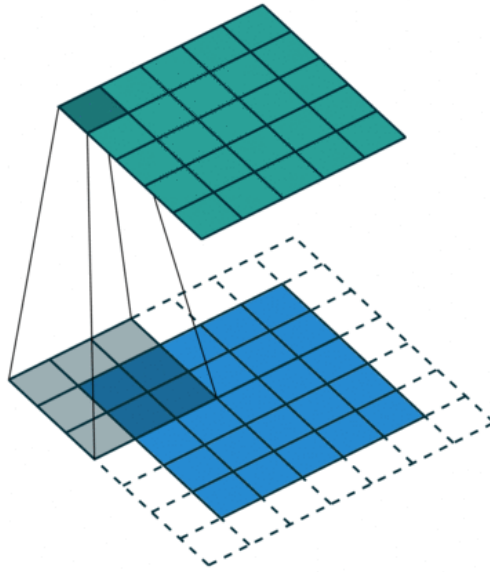


Figure 3.4 same convolution (Jiulin, 2020)

“Full” means stride equals 1, and if image size is $D1 \times D1$, the size of convolution kernel is $D2 \times D2$, and after passing through the convolution, the image size will be $(D1 + D2 - 1)^2$ (Olaf, 2015).

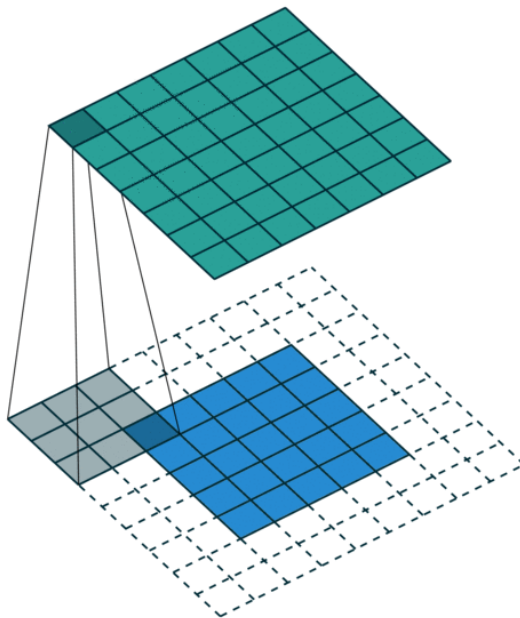


Figure 3.5 full convolution (Jiulin, 2020)

In 3D U-Net, pooling is 2×2 , and it uses “valid” strategy, so it is necessary to ensure that the input image has an even number of side lengths before pooling layer.

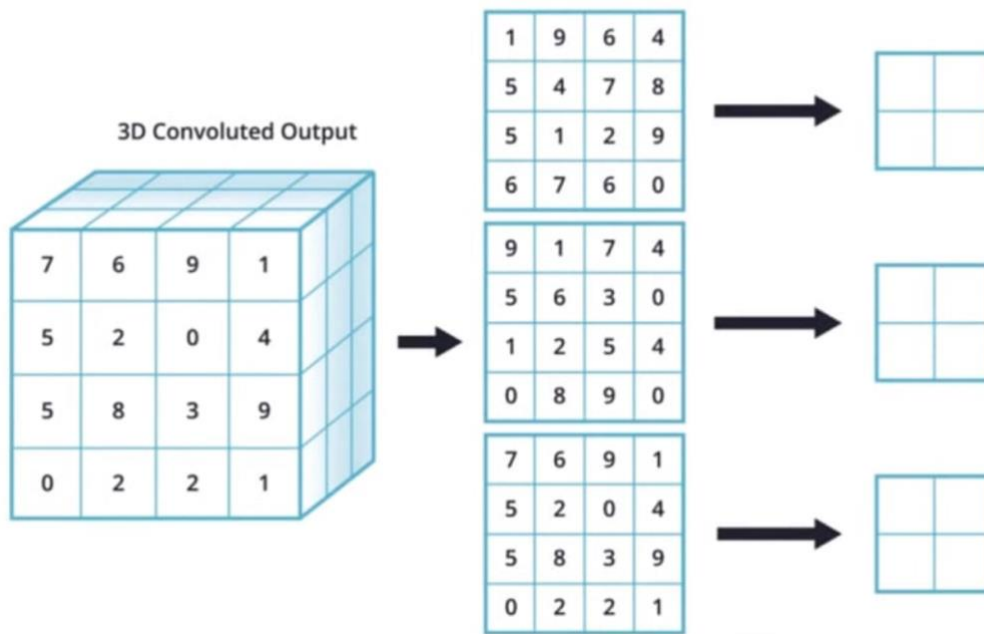


Figure 3.6 Maxing Pooling Layer (Sandeep, 2020)

In-Network upsampling: “Max Unpooling”

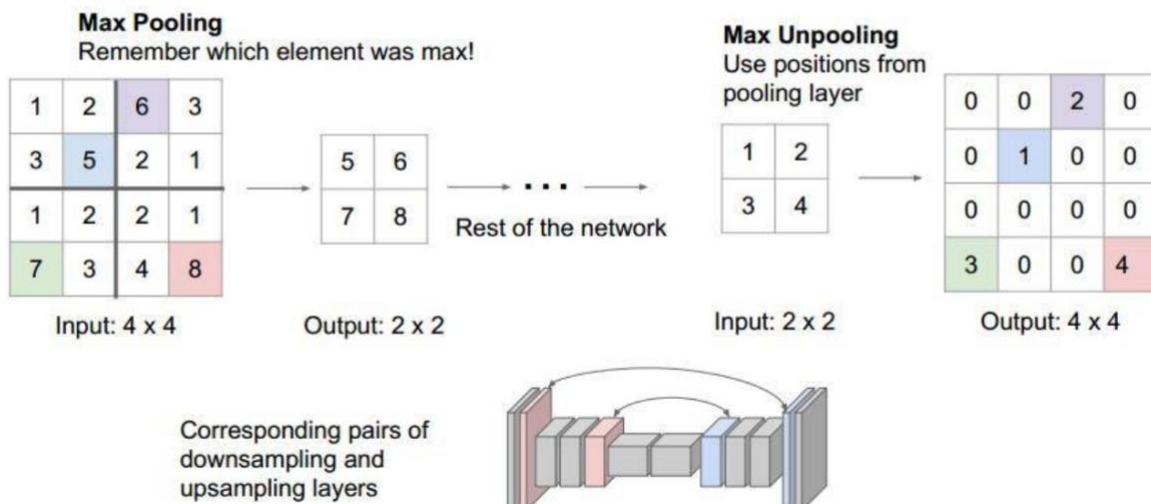


Figure 3.7 Up-Convolution Layer (Oldpan, 2018).

3.3 Data Preprocessing

As shown above, the datasets are collected by Brats 2018, and glioma in MRI scan is multi-modal data, classified into flair, t1, t1ce, and t2, these 4 types of images can be considered

as 4 different latitude information of MRI images. In the medical world, things like t1, t2, t1ce, flair are called sequences. A case can have multiple sequences, each sequence is composed of many slices, in addition, the way to obtain each sequence is different (Brats, 2018). The flair sequence is a commonly used sequence of nuclear magnetic resonance. The full name is the fluid attenuation inversion recovery sequence. In this sequence, the cerebrospinal fluid presents a low signal, and the substantive lesions and the lesions containing bound water are obvious High signal (Brats, 2018). T1 and T2 are used to measure the physical quantities of electromagnetic waves. They can be used as imaging data. The imaging based on T1 is called "T1-weighted-imaging", and it is often referred to as "T1" and T2 in clinical work (Brats, 2018). T2 is similar and the signal of it is related to the water content (Brats, 2018). The t1ce sequence needs to be used to create a contrast agent in the blood when doing MR (Brats, 2018). The blood supply is abundant in the bright place, and the enhancement clearly indicates that the blood flow is very rich (Brats, 2018). We should process the datasets before feeding the data images into the model because the changes of different modalities used for mapping tumor-induced tissue changes cause changes in the intensity range (Po-Yu et al., 2019).

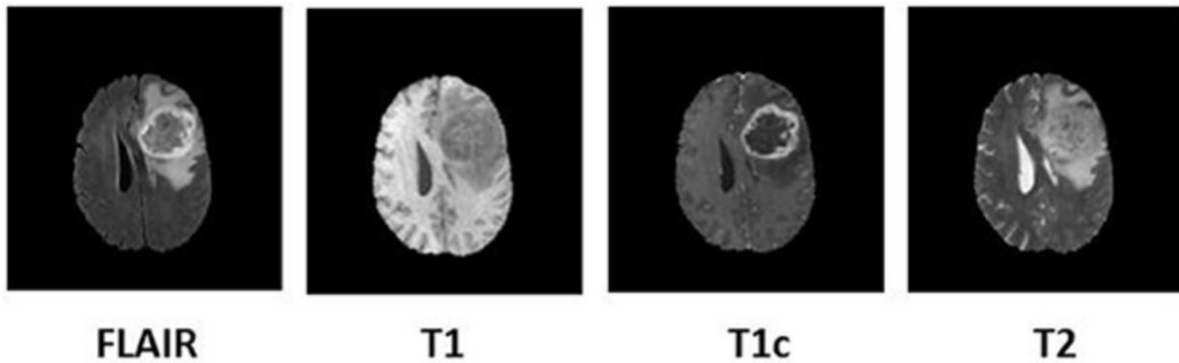


Figure 3.8 four modalities of MRI (Ujjwal et al., 2020)

3.3.1 Cropping

I store the data of 4 modalities of MRI into different matrices which represent fair, t1, t1ce, and t2. Mask image where brain tumor in MRI has been segmented into 3 sub-regions, whole tumor, the tumor core, and the enhancing tumor structures manually also is saved in a

matrix. Then, these matrices should be cropped because the grey part of Brats MRI is the brain area, and the black part is background which account for a large scale of the whole imaging, while background is useless for segmentation (Po-Yu et al., 2019). From the doctor's point of view, background information in MRI will be automatically filtered out (Po-Yu et al., 2019). Thus, it is essential to remove the background information. Simultaneously, cropping will reduce the scale of network and improve the quality of network.

3.3.2 Normalizing

T1, T2, flair, and T1ce represents 4 different modalities of one image, which means the image contrast is different, so the z-score method is used to separate each modal image. Each mode into a standard value range is standardized, and then Each MR image is preprocessed by first cropping it at non-zero voxels (1 percentile, 99 percentile) to remove outliers (Po-Yu et al., 2019). Next, use $\bar{x}_i = (x_i - \mu) / \sigma$ to normalize each mode separately, where i is the index of the voxel inside the brain, \bar{x}_i is the normalized voxel, and x_i is the corresponding original voxel, And μ and σ are the mean and standard deviation of primitive voxels inside the brain (Po-Yu et al., 2019).

Image normalization is the processing of data centralization by de-averaging. According to the convex optimization theory and the related knowledge of data probability distribution, data centralization conforms to the law of data distribution, and it is easier to obtain generalization effects after training. Data standardization is one of the common methods about data preprocessing.

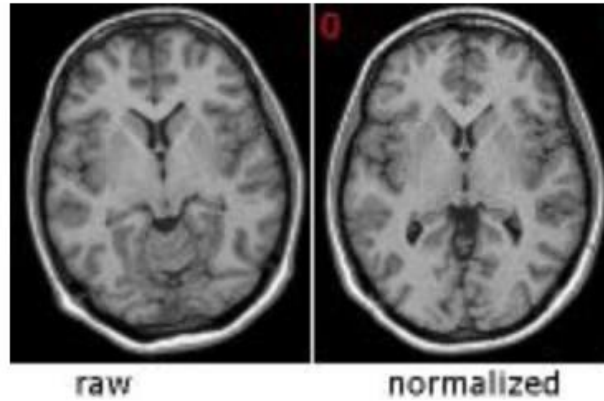


Figure 3.9 original image vs. normalized image (Chris, 2002)

3.3.3 Insert Slices and Patch Block Processing

In the beginning, I set the input size as $160 \times 192 \times 169 \times 4$, however, during the process of training this network, the RAM always broke down, which means due to the limitation of the size of the memory and graphics card resources, the image cannot be completely input to the network. Therefore, I need to insert slice into each input and block the image and the corresponding mask. The size of the cropped image and mask is $(160, 160, 160)$, where my block size is $(32, 160, 160)$, that is, 5 blocks of $(32, 160, 160)$ size are divided from the Axial direction. At last, the input size is $160 \times 32 \times 160 \times 4$. Although this approach will influence the performance of network, at least I can run the code in this terrible GPU. Moreover, the resolution of the block is generally an even number so that it can be divisible when pooling and down-sampling.

3.3.4 Pickle

The four modes after normalization and blocking are merged into four channels, and the saved shape is $160 \times 32 \times 160 \times 4$. Subsequently, we should merge 3 labels of mask into 3 nested sub-regions, and finally merge them into three channels which are WT, TC, ET respectively, so the shape of final mask is $160 \times 32 \times 160 \times 3$.

During the process of my study, I use Google Colaboratory to run code, but when I store all data into list or tensor, I find the RAM is not enough. Therefore, after normalizing and cropping, I make use of pickle which is a standard module of the python language so that I can save the information of image data in the program to the file, and the file can be loaded when I need datasets to train my network.

3.4 Summary

This chapter presents the urgent requirement of this project in the medical field, and the vital role this project plays in processing and analyzing MR imaging data. Then, the details of building 3D U-Net model are described. Next, the procedures of preprocessing are shown.

Chapter 4

System Implementation

The implementation of this system includes 4 steps, preparing and preprocessing data, building the architecture of neural network, feeding data into the network for training, and evaluating the performance of it.

4.1 Data preparation

As shown above, the data of four modalities image is stored in a matrix, and the data of label image is stored in another matrix, and then insert the slices into the original image data. After normalizing and cropping these matrices, I use pickle package to save these two matrices into two 'pkl' files. Dataset contains 285 cases in all, so we have 285 'pkl' files storing the data of four modalities image, and 285 'pkl' files storing the data of label data. I randomly set 70% of dataset as training data, and 15% as testing data, and 15% as validation data.

The code of cropping and normalizing

```
def crop_img(img, rtol=1e-8, copy=True, return_slices=False):
    img = check_niimg(img)
    data = img.get_data()
    infinity_norm = max(-data.min(), data.max())
    passes_threshold = np.logical_or(data < -rtol * infinity_norm,
                                     data > rtol * infinity_norm)

    if data.ndim == 4:
        passes_threshold = np.any(passes_threshold, axis=-1)
        coords = np.array(np.where(passes_threshold))
        start = coords.min(axis=1)
        end = coords.max(axis=1) + 1

    # pad with one voxel to avoid resampling problems
    start = np.maximum(start - 1, 0)
    end = np.minimum(end + 1, data.shape[:3])

    slices = [slice(s, e) for s, e in zip(start, end)]

    if return_slices:
        return slices

    return crop_img_to(img, slices, copy=copy)
```

```
def normalize(slice, bottom=99, down=1):

    b = np.percentile(slice, bottom)
    t = np.percentile(slice, down)
    slice = np.clip(slice, t, b)

    image_nonzero = slice[np.nonzero(slice)]
    if np.std(slice) == 0 or np.std(image_nonzero) == 0:
        return slice
    else:
        tmp = (slice - np.mean(image_nonzero)) / np.std(image_nonzero)
        tmp[tmp == tmp.min()] = -9
        return tmp
```

The code of inserting slices

```
def insert_slices(array):
    myblackslice = np.zeros([240,240])
    array = np.insert(array,0,myblackslice,axis = 0)
    array = np.insert(array,0,myblackslice,axis = 0)
    array = np.insert(array,0,myblackslice,axis = 0)
    array = np.insert(array,array.shape[0],myblackslice,axis = 0)
    array = np.insert(array,array.shape[0],myblackslice,axis = 0)

    array_nor = normalize(array)
    crop = crop_ceter(array_nor,160,160)

    patch_block_size = BLOCKSIZE
    numberxy = patch_block_size[1]
    numberz = patch_block_size[0]
    width = np.shape(crop)[1]
    height = np.shape(crop)[2]
    imagez = np.shape(crop)[0]
    block_width = np.array(patch_block_size)[1]
    block_height = np.array(patch_block_size)[2]
    blockz = np.array(patch_block_size)[0]
    stridewidth = (width - block_width) // numberxy
    strideheight = (height - block_height) // numberxy
    stridez = (imagez - blockz) // numberz
    step_width = width - (stridewidth * numberxy + block_width)
    step_width = step_width // 2
    step_height = height - (strideheight * numberxy + block_height)
    step_height = step_height // 2
    step_z = imagez - (stridez * numberz + blockz)
    step_z = step_z // 2

    hr_samples_list = []
    patchnum = []
    for z in range(step_z, numberz * (stridez + 1) + step_z, numberz):
        for x in range(step_width, numberxy * (stridewidth + 1) + step_width, numberxy):
            for y in range(step_height, numberxy * (strideheight + 1) + step_height, numberxy):
                patchnum.append(z)
                hr_samples_list.append(crop[z:z + blockz, x:x + block_width, y:y + block_height])
    samples_arr = np.array(hr_samples_list).reshape((len(hr_samples_list), blockz, block_width, block_height))

    return samples_arr
```

The code of data preparation

```
for file in files[1:]:
    file = file + '/'
    sub_files = os.listdir(root + image_dir + hgg + file)
    hgg_dir_list.append(file)
    flair = []
    seg = []
    t1 = []
    tlce = []
    t2 = []

    for fil in sub_files:
        if 'flair.nii' in fil:
            data = read_image(root + image_dir + hgg + file + fil)
            img = insert_slices(data)
            img = normalize(img)
            flair.append(img)
        elif 'seg.nii' in fil:
            data = read_image(root + image_dir + hgg + file + fil)
            img = insert_slices(data)
            img = normalize(img)
            seg.append(img)
        elif 't1.nii' in fil:
            data = read_image(root + image_dir + hgg + file + fil)
            img = insert_slices(data)
            img = normalize(img)
            t1.append(img)
        elif 'tlce.nii' in fil:
            data = read_image(root + image_dir + hgg + file + fil)
            img = insert_slices(data)
            img = normalize(img)
            tlce.append(img)
        elif 't2.nii' in fil:
            data = read_image(root + image_dir + hgg + file + fil)
            img = insert_slices(data)
            img = normalize(img)
            t2.append(img)
```

```
samples, imagez, height, width = np.shape(flair[0])[0], np.shape(flair[0])[1], np.shape(flair[0])[2], np.shape(flair[0])[3]
fourModels, masks = merge(samples, imagez, height, width, flair[0], t1[0], tlce[0], t2[0], seg[0])

fourModels = np.rollaxis(fourModels, 1, 0)
# fourModels = np.rollaxis(fourModels, 3, 0)
masks = np.rollaxis(masks, 1, 0)
# masks = np.rollaxis(masks, 3, 0)

print(fourModels.shape)
print(masks.shape)

ID = file.replace('/', '')
pickle.dump( fourModels, open( f"{ID}_images.pkl", "wb" ) )
pickle.dump( masks, open( f"{ID}_seg_mask_3ch.pkl", "wb" ) )
```

4.2 Model building

The code of creating a 3D U-Net model.

```
inputs = Input(input_shape)

current_layer = inputs
level_output_layers = list()
level_filters = list()
for level_number in range(depth):
    n_level_filters = (2**level_number) * n_base_filters
    level_filters.append(n_level_filters)

    if current_layer is inputs:
        in_conv = create_convolution_block(current_layer, n_level_filters)
    else:
        in_conv = create_convolution_block(current_layer, n_level_filters, strides=(2, 2, 2))

    context_output_layer = create_context_module(in_conv, n_level_filters, dropout_rate=dropout_rate)
    summation_layer = Add()([in_conv, context_output_layer])
    level_output_layers.append(summation_layer)
    current_layer = summation_layer

segmentation_layers = list()
for level_number in range(depth - 2, -1, -1):
    up_sampling = create_up_sampling_module(current_layer, level_filters[level_number])
    concatenation_layer = concatenate([level_output_layers[level_number], up_sampling], axis=4)
    localization_output = create_localization_module(concatenation_layer, level_filters[level_number])
    current_layer = localization_output
    if level_number < n_segmentation_levels:
        segmentation_layers.insert(0, create_convolution_block(current_layer, n_filters=n_labels, kernel=(1, 1, 1)))

output_layer = None
for level_number in reversed(range(n_segmentation_levels)):
    segmentation_layer = segmentation_layers[level_number]
    if output_layer is None:
        output_layer = segmentation_layer
    else:
        output_layer = Add()([output_layer, segmentation_layer])

    if level_number > 0:
        output_layer = UpSampling3D(size=(2, 2, 2))(output_layer)
```

```
activation_block = Activation(activation = activation_name, name='activation_block')(output_layer)

survival_conv_1 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='survival_conv_1')(summation_layer)
survival_conv_2 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='survival_conv_2')(survival_conv_1)
dropout = SpatialDropout3D(rate=dropout_rate, data_format='channels_first', name='dropout')(survival_conv_2)
survival_conv_3 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='survival_conv_3')(dropout)
survival_GAP = GlobalAveragePooling3D(name='survival_GAP')(survival_conv_3)
survival_block = Dense(1, activation='linear', name='survival_block')(survival_GAP)

tumortype_conv_1 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='tumortype_conv_1')(summation_layer)
tumortype_conv_2 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='tumortype_conv_2')(tumortype_conv_1)
tumortype_dropout = SpatialDropout3D(rate=dropout_rate, data_format='channels_first', name='tumortype_dropout')(tumortype_conv_2)
tumortype_conv_3 = Conv3D(filters=n_level_filters, kernel_size=(3, 3, 3), padding='same', strides=(1, 1, 1), name='tumortype_conv_3')(tumortype_dropout)
tumortype_GAP = GlobalAveragePooling3D(name='tumortype_GAP')(tumortype_conv_3)
tumortype_block = Dense(1, activation='sigmoid', name='tumortype_block')(tumortype_GAP)

model = Model(inputs=inputs, outputs=[activation_block])
```

4.3 Training procedure

The 3D U-Net model is trained with matrices which includes the data of 4 modalities of size $160 \times 32 \times 160$ and batch size 1. The epochs are set as 300 to train this network, and the parameters of this network are updated by Adam algorithm, and I set learning rate equals 10^{-3} (Kingma and Ba, 2015, cited in Po-Yu et al., 2019), and the dropout is 0.3. Loss function is soft-max function to calculate the error from the prediction value to the labeled sample.

Train and save the parameters of network

```
model.compile(optimizer=Adam(lr=1e-2),
              loss={'activation_block': weighted_dice_coefficient_loss},
              loss_weights={'activation_block': 1.},
              metrics={'activation_block': ['accuracy', weighted_dice_coefficient, dice_coefficient]})

params = {'dim': (160, 32, 160),
          'batch_size': 1,
          'n_classes': 3,
          'n_channels': 4,
          'shuffle': True}

# Generators
print(len(train_val_test_dict['train']))
training_generator = SingleDataGenerator(train_val_test_dict['train'], **params)
validation_generator = SingleDataGenerator(train_val_test_dict['val'], **params)

cb_1=keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='au
cb_2=keras.callbacks.ModelCheckpoint(filepath="1Bpred_weights.{epoch:02d}-{val_loss:.2f}.hdf5", mon

results = model.fit_generator(generator=training_generator,
                             validation_data=validation_generator,
                             epochs=100,
                             callbacks=[cb_1, cb_2])
model.save_weights("model_1_weights.h5")
```

4.4 Testing procedure

During the period of testing, we input the matrices of test data into trained 3D U-Net model.

The code of testing the model.

```
history_1_pred = results.history
pickle.dump( history_1_pred, open( "history_1_pred.pkl", "wb" ) )

params = {'dim': (160,192,160),
          'batch_size': 1,
          'n_classes': 3,
          'n_channels': 4,
          'shuffle': False}

validation_generator = SingleDataGenerator(partition['holdout'], **params)

predictions_1_pred = model.predict_generator(generator=validation_generator)

pickle.dump( predictions_1_pred, open( "predictions_1_pred.pkl", "wb" ) )
```

4.5 Implementation Details of System

Firstly, reading data from 'nii' file, and then normalizing and cropping the data. Next, data is stored in the matrices. Subsequently, using pickle package to save these matrices. Also, we need to split dataset into training data, validation data, and test data, and the training data is fed into 3D U-Net for training.

When inputting dataset into 3D U-Net, it will pass the encoding path at first, and this path is used to analyze the entire picture and perform feature extraction and analysis. In this study, the size of original is $160 \times 32 \times 160 \times 4$ where $160 \times 32 \times 160$ is the dimension of image and 4 is the channels, after passing through the first 3D convolutions, the number of channels changes from 4 to 12, and then starting 4 times down-sampling. After being encoded, the output is a feature map with shape $10 \times 2 \times 10 \times 192$, then this feature map will be the input of decoding path, and this path can generate a segmented block image with the same size as original image. During the process of decoding, feature map needs to perform 4 times up-sampling, and the size changes from $10 \times 2 \times 10 \times 192$ to $160 \times 32 \times 160 \times 3$. We notice that the number of channels here is 3, because the number of labels of label image is 3, and labels are 3 sub-regions, whole tumor, tumor core, and enhancing tumor. Note that at the same stage of 3D U-Net, skip-connection is used to concatenate the low-level features and high-level features of image. During the implementation of network, batch normalization is used before ReLU. During training, mean and standard deviation of all pixels are used to

normalize the value of input in each batch. Weighted soft-max loss function is set in this architecture in order to make this model have ability to train sparse annotations (Özgün, 2016). The weight of unlabeled pixels is set as zero, which makes network just learn from labeled pixels, so this network has excellent generalization (Özgün, 2016).

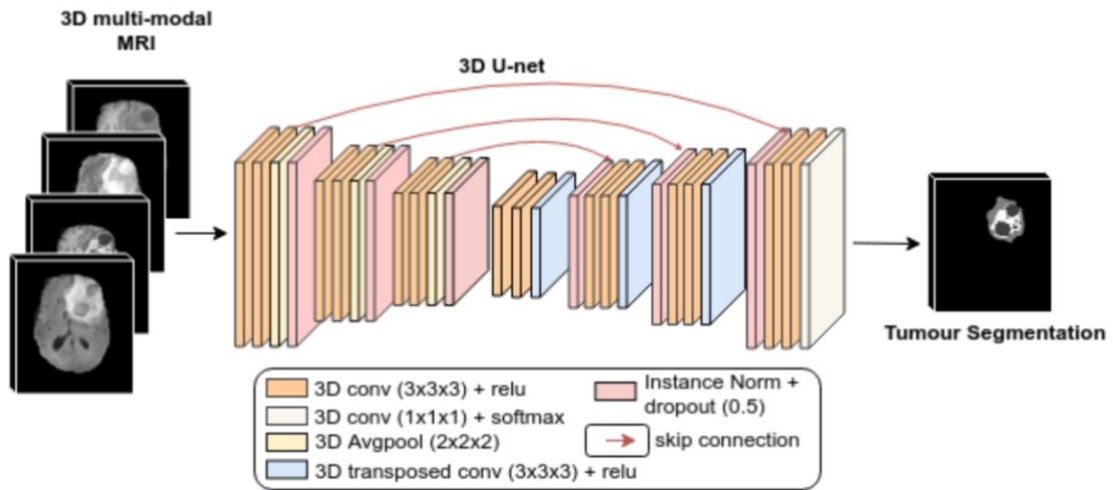


Figure 4.2 the segmentation process of 3D U-Net (Raghav and Tal, 2018)

The summary of this model is shown below.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 160, 32, 160, 4)]	0	
conv3d (Conv3D)	(None, 160, 32, 160, 12)	1308	input_1[0][0]
leaky_re_lu (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d[0][0]
conv3d_1 (Conv3D)	(None, 160, 32, 160, 12)	3900	leaky_re_lu[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d_1[0][0]
spatial_dropout3d (SpatialDropout3D)	(None, 160, 32, 160, 12)	0	leaky_re_lu_1[0][0]
conv3d_2 (Conv3D)	(None, 160, 32, 160, 12)	3900	spatial_dropout3d[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d_2[0][0]

conv3d_3 (Conv3D)	(None, 80, 16, 80, 24)	7800	add[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_3[0][0]
conv3d_4 (Conv3D)	(None, 80, 16, 80, 24)	15576	leaky_re_lu_3[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_4[0][0]
spatial_dropout3d_1 (SpatialDropout3D)	(None, 80, 16, 80, 24)	0	leaky_re_lu_4[0][0]
conv3d_5 (Conv3D)	(None, 80, 16, 80, 24)	15576	spatial_dropout3d_1[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_5[0][0]
add_1 (Add)	(None, 80, 16, 80, 24)	0	leaky_re_lu_3[0][0] leaky_re_lu_5[0][0]
conv3d_6 (Conv3D)	(None, 40, 8, 40, 48)	31152	add_1[0][0]
leaky_re_lu_6 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_6[0][0]
conv3d_7 (Conv3D)	(None, 40, 8, 40, 48)	62256	leaky_re_lu_6[0][0]
leaky_re_lu_7 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_7[0][0]
spatial_dropout3d_2 (SpatialDropout3D)	(None, 40, 8, 40, 48)	0	leaky_re_lu_7[0][0]
conv3d_8 (Conv3D)	(None, 40, 8, 40, 48)	62256	spatial_dropout3d_2[0][0]
leaky_re_lu_8 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_8[0][0]
add_2 (Add)	(None, 40, 8, 40, 48)	0	leaky_re_lu_6[0][0] leaky_re_lu_8[0][0]
conv3d_9 (Conv3D)	(None, 20, 4, 20, 96)	124512	add_2[0][0]
leaky_re_lu_9 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_9[0][0]
conv3d_10 (Conv3D)	(None, 20, 4, 20, 96)	248928	leaky_re_lu_9[0][0]
leaky_re_lu_10 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_10[0][0]
spatial_dropout3d_3 (SpatialDropout3D)	(None, 20, 4, 20, 96)	0	leaky_re_lu_10[0][0]
conv3d_11 (Conv3D)	(None, 20, 4, 20, 96)	248928	spatial_dropout3d_3[0][0]
leaky_re_lu_11 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_11[0][0]
add_3 (Add)	(None, 20, 4, 20, 96)	0	leaky_re_lu_9[0][0] leaky_re_lu_11[0][0]
conv3d_12 (Conv3D)	(None, 10, 2, 10, 192)	497856	add_3[0][0]
leaky_re_lu_12 (LeakyReLU)	(None, 10, 2, 10, 192)	0	conv3d_12[0][0]
conv3d_13 (Conv3D)	(None, 10, 2, 10, 192)	995520	leaky_re_lu_12[0][0]
leaky_re_lu_13 (LeakyReLU)	(None, 10, 2, 10, 192)	0	conv3d_13[0][0]
spatial_dropout3d_4 (SpatialDropout3D)	(None, 10, 2, 10, 192)	0	leaky_re_lu_13[0][0]
conv3d_14 (Conv3D)	(None, 10, 2, 10, 192)	995520	spatial_dropout3d_4[0][0]
leaky_re_lu_14 (LeakyReLU)	(None, 10, 2, 10, 192)	0	conv3d_14[0][0]
add_4 (Add)	(None, 10, 2, 10, 192)	0	leaky_re_lu_12[0][0] leaky_re_lu_14[0][0]

up_sampling3d (UpSampling3D)	(None, 20, 4, 20, 192)	0	add_4[0][0]
conv3d_15 (Conv3D)	(None, 20, 4, 20, 96)	497760	up_sampling3d[0][0]
leaky_re_lu_15 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_15[0][0]
concatenate (Concatenate)	(None, 20, 4, 20, 192)	0	add_3[0][0] leaky_re_lu_15[0][0]
conv3d_16 (Conv3D)	(None, 20, 4, 20, 96)	497760	concatenate[0][0]
leaky_re_lu_16 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_16[0][0]
conv3d_17 (Conv3D)	(None, 20, 4, 20, 96)	9312	leaky_re_lu_16[0][0]
leaky_re_lu_17 (LeakyReLU)	(None, 20, 4, 20, 96)	0	conv3d_17[0][0]
up_sampling3d_1 (UpSampling3D)	(None, 40, 8, 40, 96)	0	leaky_re_lu_17[0][0]
conv3d_18 (Conv3D)	(None, 40, 8, 40, 48)	124464	up_sampling3d_1[0][0]
leaky_re_lu_18 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_18[0][0]
concatenate_1 (Concatenate)	(None, 40, 8, 40, 96)	0	add_2[0][0] leaky_re_lu_18[0][0]
conv3d_19 (Conv3D)	(None, 40, 8, 40, 48)	124464	concatenate_1[0][0]
leaky_re_lu_19 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_19[0][0]
conv3d_20 (Conv3D)	(None, 40, 8, 40, 48)	2352	leaky_re_lu_19[0][0]
leaky_re_lu_20 (LeakyReLU)	(None, 40, 8, 40, 48)	0	conv3d_20[0][0]
up_sampling3d_2 (UpSampling3D)	(None, 80, 16, 80, 48)	0	leaky_re_lu_20[0][0]
conv3d_22 (Conv3D)	(None, 80, 16, 80, 24)	31128	up_sampling3d_2[0][0]
leaky_re_lu_22 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_22[0][0]
concatenate_2 (Concatenate)	(None, 80, 16, 80, 48)	0	add_1[0][0] leaky_re_lu_22[0][0]
conv3d_23 (Conv3D)	(None, 80, 16, 80, 24)	31128	concatenate_2[0][0]
leaky_re_lu_23 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_23[0][0]
conv3d_24 (Conv3D)	(None, 80, 16, 80, 24)	600	leaky_re_lu_23[0][0]
leaky_re_lu_24 (LeakyReLU)	(None, 80, 16, 80, 24)	0	conv3d_24[0][0]
up_sampling3d_3 (UpSampling3D)	(None, 160, 32, 160, 24)	0	leaky_re_lu_24[0][0]
conv3d_26 (Conv3D)	(None, 160, 32, 160, 12)	7788	up_sampling3d_3[0][0]
leaky_re_lu_26 (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d_26[0][0]
concatenate_3 (Concatenate)	(None, 160, 32, 160, 24)	0	add[0][0] leaky_re_lu_26[0][0]
conv3d_27 (Conv3D)	(None, 160, 32, 160, 12)	7788	concatenate_3[0][0]
conv3d_21 (Conv3D)	(None, 40, 8, 40, 3)	147	leaky_re_lu_20[0][0]
leaky_re_lu_27 (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d_27[0][0]
leaky_re_lu_21 (LeakyReLU)	(None, 40, 8, 40, 3)	0	conv3d_21[0][0]
conv3d_25 (Conv3D)	(None, 80, 16, 80, 3)	75	leaky_re_lu_24[0][0]
conv3d_28 (Conv3D)	(None, 160, 32, 160, 12)	156	leaky_re_lu_27[0][0]

up_sampling3d_4 (UpSampling3D)	(None, 80, 16, 80, 3)	0	leaky_re_lu_21[0][0]
leaky_re_lu_25 (LeakyReLU)	(None, 80, 16, 80, 3)	0	conv3d_25[0][0]
leaky_re_lu_28 (LeakyReLU)	(None, 160, 32, 160, 12)	0	conv3d_28[0][0]
add_5 (Add)	(None, 80, 16, 80, 3)	0	up_sampling3d_4[0][0] leaky_re_lu_25[0][0]
conv3d_29 (Conv3D)	(None, 160, 32, 160, 3)	39	leaky_re_lu_28[0][0]
up_sampling3d_5 (UpSampling3D)	(None, 160, 32, 160, 3)	0	add_5[0][0]
leaky_re_lu_29 (LeakyReLU)	(None, 160, 32, 160, 3)	0	conv3d_29[0][0]
add_6 (Add)	(None, 160, 32, 160, 3)	0	up_sampling3d_5[0][0] leaky_re_lu_29[0][0]
activation_block (Activation)	(None, 160, 32, 160, 3)	0	add_6[0][0]
=====			
Total params: 4,649,949			
Trainable params: 4,649,949			
Non-trainable params: 0			

During the process of training, validation data is used to avoid overfitting. After roughly 24 hours training, I get the trained 3D U-Net model, and then using test data to evaluate it.

4.6 Summary

Firstly, this chapter gives the code of data preparing, model building, training and test procedures. Subsequently, the details of system implementation are present. At last, showing the summary of 3D U-Net model.

Chapter 5

System Evaluation

5.1 Evaluation Metrics

5.1.1 Dice Coefficient (DSC)

Dice coefficient is used as evaluation metric. It is a function that can calculate the similarity of predicted lesions and ground-truth lesions (Takafumi et al., 2020), and the value range of it is from 0 to 1.

It is defined as:

$$s = \frac{2|X \cap Y|}{|X| + |Y|}$$

(Takafumi et al., 2020)

where $|X \cap Y|$ is the intersection between X and Y, $|X|$ and $|Y|$ represent the number of voxels in the ground-truth and prediction respectively (Po-Yu et al., 2019). Among them, the coefficient of the numerator is 2, because the denominator has repeated calculations of the common elements between X and Y.

The code of this function is shown here:

```
def dice_coefficient(y_true, y_pred, smooth=1.):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def dice_coefficient_loss(y_true, y_pred):
    return -dice_coefficient(y_true, y_pred)

def weighted_dice_coefficient(y_true, y_pred, axis=(-3, -2, -1), smooth=0.00001):
    return K.mean(2. * (K.sum(y_true * y_pred,
                                axis=axis) + smooth/2)/(K.sum(y_true,
                                                                axis=axis) + K.sum(y_pred,
                                                                axis=axis) + smooth))

def weighted_dice_coefficient_loss(y_true, y_pred):
    return -weighted_dice_coefficient(y_true, y_pred)

def label_wise_dice_coefficient(y_true, y_pred, label_index):
    return dice_coefficient(y_true[:, label_index], y_pred[:, label_index])

def get_label_dice_coefficient_function(label_index):
    f = partial(label_wise_dice_coefficient, label_index=label_index)
    f.__setattr__('__name__', 'label_{0}_dice_coef'.format(label_index))
    return f
```

5.1.2 Hausdorff Distance (HD95)

Hausdorff distance is a measure to describe the similarity between two sets of points, and in this paper, Hausdorff Distance is used to compute the distance between the predicted value and the ground-truth value (Normand and Mikael, 1998). Suppose there are two sets of sets $A=\{a_1, \dots, a_i\}$, $B=\{b_1, \dots, b_i\}$ where A is the set of predicted pixels and B is the set true pixels, then the function of computing Hausdorff distance is defined as:

$$H(A,B)=\max(h(A,B),h(B,A))$$

Where,

$$h(A,B)=\max (a \in A) \min (b \in B) \| a-b \|$$

$$h(B,A)=\max (b \in B) \min (a \in A) \| b-a \|$$

$\| \cdot \|$ represents the Euclidean Distance

(Normand and Mikael, 1998)

5.1.3 Intersection over Union (IoU)

IoU is a parameter in the evaluation system of target detection (Adrian, 2016). It is the ratio of the intersection of the real image and the predicted image generated by the model, and the formula of computing IoU is the intersection of the true value and the predicted value divided by their union, which is the detection accuracy IoU (Adrian, 2016):


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 4.5.2.1 function of IoU (Adrian, 2016)

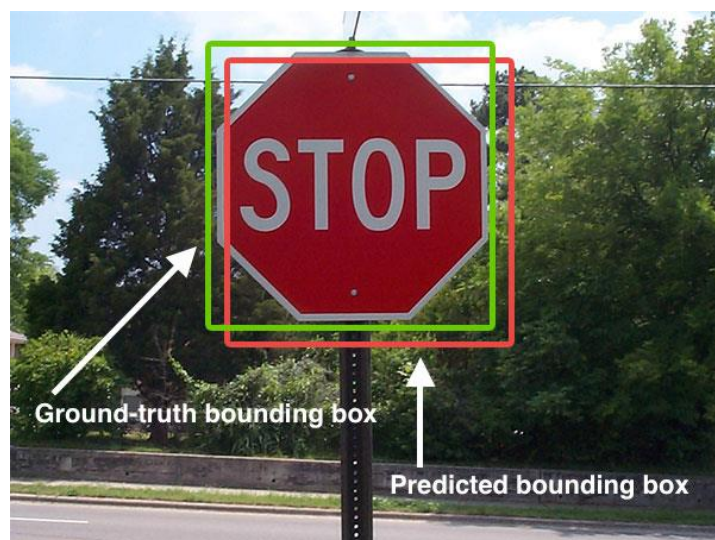
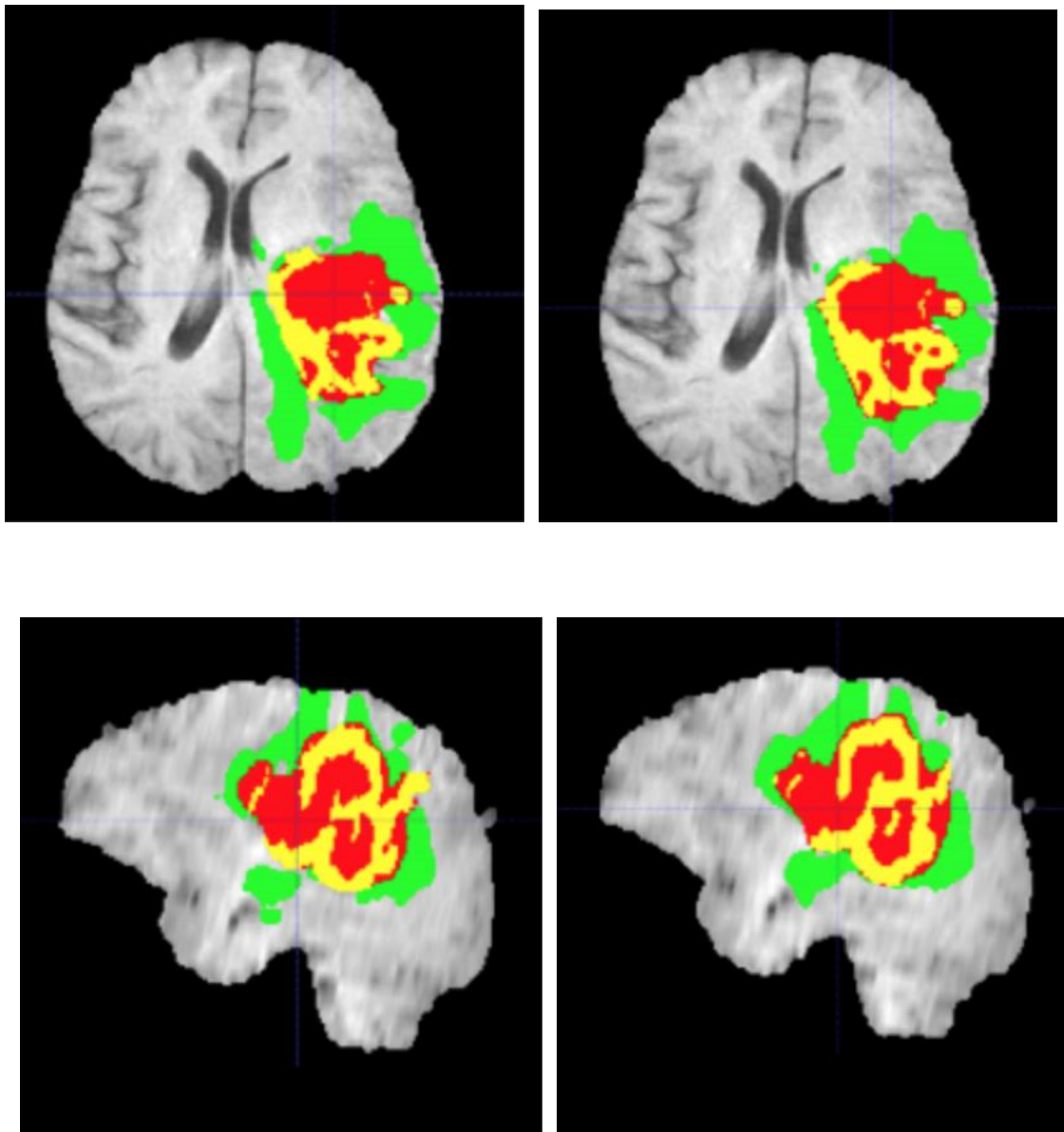


Figure 4.5.2.2 example of IoU (Adrian, 2016)

5.2 Evaluation of Generated Image vs. Truth Image



Truth Image

Predicted Image

5.3 Evaluation from DSC and HD95

3D U-Net	DSC	HD95
ET	75.3	5.73
WT	90.1	4.91
TC	76.9	10.55

Table 5.1 evaluation of 3D U-Net

ET, WT, TC are three sub-regions of glioma, and DSC scores are 75.3, 90.1, and 76.9 respectively, and HD95 scores are 5.73, 4.91, and 10.55 respectively.

5.4 3D U-Net Performance vs. state-of-the-art Method

	DSC_ET	DSC_WT	DSC_TC	HD95_ET	HD95_WT	HD95_TC
this model	75.3	90.1	76.9	5.73	4.91	10.55
Rupal and Mehul	92	90	79	4.07	4.23	3.75
Andriy	76.6	88.4	81.5	3.77	5.9	4.8
Po-Yu et al	78.3	90.7	81.3	3.72	4.35	7.77

Table 5.2 3D U-Net vs. state-of-the-art Methods

5.5 Summary

This chapter shows the evaluation metrics of this project, and presents the evaluation results depending on these metrics, which represents the performance of model designed in this study.

Chapter 6

Conclusion and Future Work

This chapter aims to give the summary of this paper, and describe the outcome of this project, and then display shortcomings which are required to enhance in the future.

6.1 Conclusion

Firstly, to understand the problem which is segmentation of multi-modal Magnetic Resonance Images from brain tumor based on the investigation of previous researches with the same topic.

Secondly, to explain the concept of glioma, MRI and deep learning.

Thirdly, to check the aim of this project which is to design, build and evaluate a neural network model to segment brain tumor in pre-operative MRI scans. In order to complete the objectives and deliverables of this paper, multiple research surveys about segmentation of brain tumor from MRI are read and analyzed, and depending on these surveys, appropriate techniques to build this model are chosen. Except building model, another important step in this study is to preprocess the datasets for training network, which plays a vital role in improving the performance of this model.

Then, to describe the functions and advantages of these techniques, and give the reasons why choose these techniques here.

Next, to start design stage, describe the architecture of 3D U-Net, and emphasize the key features of this model, such as U-shaped structure and skip-connection. Also, to describe the approaches of preprocessing data, including cropping and normalizing.

Moreover, to present the code of this system, and give the details of implementation process of this system

At last, to give the evaluation metrics of this model, and use these metrics to evaluation the performance of the model designed in this thesis.

In conclusion, this paper presents a high-quality model that can automatically segment brain tumor in MRI.

6.2 Future Work

This paper just reproduces the performance of 3D U-Net. However, actually, I have a novel idea to modify this model. Because of the limited time and facilities, it is hard to make this idea come with true. When running the code, the memory of GPU is always broken down, and every time this error happening will consume lots of time running again. Therefore, I have to put it into practice in the future.

This idea is that pre-trained VGGNet or ResNet or other models is set as encoding path, and the structure of decoding path is set to symmetry with encoding path. During the process of training, we just need to update the parameters in decoding path and keep no change of parameters in encoding path, because I think the function of encoding path is to extract the features of the images, and the pre-trained model has been trained by massive datasets, it will be certain to complete this function perfectly without more training. This way is able to not only extract features excellently, but also save much time and source to train encoding path. The implementation of decoding path is just like U-Net to update parameters, but due to the more complicated structure, the process will be more time-consuming.

However, the only drawback of this idea is that the requirement of training this model is very high, although we just need to update the parameters of decoding path, the structure of

decoder is very complicated and the size of input data is also very big, which means running it need to account for much memory. Obviously, the GPU in Google Colab is not sufficient for it, when an advanced hardware equipment is available to me, I will try the feasibility of this method.

6.3 Challenge

I encountered several challenges during the period of doing this project. Firstly, I need to spend much time reading and understanding a large number of surveys about segmenting images depending on neural network. Also, when writing code of this project, I met with various ridiculous errors. Sometimes just one trivial error cost me one or more hours to find a way to deal with it. For example, when I was building the structure of 3D U-Net, I found I always got the wrong output size, and then I spent roughly 3 days trying to understand why this error happened, at last, I found just because one function in tensorflow had changed with the change of version of tensorflow. However, I think the main challenge during this period is limited facility. The most frequent error I encountered during the period of writing the code of this project is that my RAM broke down. I run the project in the Google Colab, and the memory of GPU is 25 GB, but when I input the image data into the model for training, the memory is always not enough to run this code, finally, I had to slice the image data, but this operation would have a bad effect on the performance of this model.

6.4 Personal Reflection

I almost put all my effort into writing my dissertation during the recent period, although due to limited facilities, the final result does not come up to my expectation, I still succeed in solving many challenges and completing this project in the end. I think it is a really precious experience for me, during this period, I learnt a lot not only about the knowledge of deep learning, but also about the attitudes to my life. I met with various challenges when designing and coding this project, many times I felt very exhausted, but I still persisted and completed this project in the end. The process is just like our life, no matter how many challenges we

face, we should never give up, and overcome them, and in the end we will be successful. Certainly, the biggest gain is enhancement of knowledge. During the process of finishing this project, I had to read, analyze and understand a large number of researches, so I have a deeper understanding of the theory of many deep learning algorithms, especially Convolutional Neural Network. Also, I master several new algorithms and methods of deep learning during this period, such as Fully Convolutional Network and skip-connection. What is more, much time spent on writing code, which is beneficial to improving my programming logic, and helps me use Python more proficiently.

List of References

Zeina, A.S, Mahbubul, A, Lasitha, V, Linmin, P, Mohamed I.E, and Khan, M.I. 2019. Feature-Guided Deep Radiomics for Glioblastoma Patient Survival Prediction. Front Neurosci.

[Online]. [Accessed 20 September 2019]. Available from:

<https://doi.org/10.3389/fnins.2019.00966>

Alberto, A, Antonio. A, and Francisco. 2018. A, Extending 2D Deep Learning Architectures to 3D Image Segmentation Problems. In: 4th International Workshop, 16 September 2018, Granada, Spain.

Huang, Y, and Feng Q. 2018. Segmentation of brain tumor on magnetic resonance images using 3D full-convolutional densely connected convolutional networks. J South Med Univ. 38(6), pp.661-668.

Tsang, S.H. 2018. Review: FCN — Fully Convolutional Network (Semantic Segmentation).

[Online]. [Accessed 5 October 2018]. Available from: [https://towardsdatascience.com/review-](https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1)

[fcn-semantic-segmentation-eb8c9b50d2d1](https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1)

Masri, A. 2019. How Does Back Propagation in Artificial Neural Networks Work. [Online].

[Accessed 29 January 2019]. Available from: [https://towardsdatascience.com/how-does-](https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7)

[back-propagation-in-artificial-neural-networks-work-c7cad873ea7](https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7)

Brownlee, J. 2019. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks.

[Online]. [Accessed 6 August 2019]. Available from:

<https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>

Brownlee, J. 2019. A Gentle Introduction to Batch Normalization for Deep Neural Networks. [Online]. [Accessed 16 January 2019]. Available from:

<https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/>

Tsang, S.H. 2019. Review: 3D U-Net — Volumetric Segmentation (Medical Image Segmentation). [Online]. [Accessed 2 April 2019]. Available from:

<https://towardsdatascience.com/review-3d-u-net-volumetric-segmentation-medical-image-segmentation-8b592560fac1>

Sugi, Y.K. 2018. What exactly can you do with Python? Here are Python's 3 main applications. [Online]. [Accessed 15 June 2018]. Available from:

<https://towardsdatascience.com/what-can-you-do-with-python-the-3-main-applications-518db9a68a78>

Yegulalp, S. 2019. What is TensorFlow? The machine learning library explained. [Online]. [Accessed 15 June 2018]. Available from: <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>

Baid, U, Talbar, S, Rane, S, Gupta, S, Meenakshi, H, Thakur, Moiyadi, A, Sable, N, Akolkar, M, and Mahajan, A. 2020. 4 modalities of MRI. [Online]. [Accessed 18 February 2020]. Available from: https://www.frontiersin.org/files/Articles/495068/fncom-14-00010-HTML-r1/image_m/fncom-14-00010-g001.jpg

Brownlee, J. 2017. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. [Online]. [Accessed 15 June 2018]. Available from: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

Brownlee, J. 2019. What is Deep Learning? [Online]. [Accessed 20 December 2019]. Available from: <https://machinelearningmastery.com/what-is-deep-learning/>

Sharma, S. 2017. Activation Functions in Neural Networks. [Online]. [Accessed 6 September 2017]. Available from: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

Gupta, D. 2020. Fundamentals of Deep Learning – Activation Functions and When to Use Them? [Online]. [Accessed 30 January 2020]. Available from: <https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/>

Kulkarni, V. 2019. Cross-Entropy for Dummies. [Online]. [Accessed 28 December 2019]. Available from: <https://towardsdatascience.com/cross-entropy-for-dummies-5189303c7735>

Nemoto, T, Futakami, N, Yagi, M, Kumabe, A, Takeda, A, Kunieda, E and Shigematsu, N. 2020. Efficacy evaluation of 2D, 3D U-Net semantic segmentation and atlas-based segmentation of normal lungs excluding the trachea and main bronchi. Journal of Radiation Research. 61(2), pp.257–264.

Cicek, O, Abdulkadir, A, Soeren, S, Lienkamp, Brox, T, and Ronneberger, O. 2016. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. [Online]. MICCAI 2016. Available from: <https://arxiv.org/abs/1606.06650v1>

Cicek, O, Abdulkadir, A, Soeren, S, Lienkamp, Brox, T, and Ronneberger, O. 2016. Volumetric Segmentation with the 3D U-Net. [Accessed 21 June 2016]. Available from: <https://arxiv.org/abs/1606.06650v1>

Bit, S. 2019. The framework of cancer metastases detection. [Online]. [Accessed 16 December 2019]. Available from: <https://www.zhihu.com/question/269914775>

Wang, X. M. 2017. Artificial Intelligence | Oncology Medical Imaging AI Recognition Technology Practice. [Online]. [Accessed 24 August 2017]. Available from: <https://segmentfault.com/a/1190000010821712>

Myronenko, A. 2018. 3D MRI brain tumor segmentation using autoencoder regularization. [Accessed 27 October 2018]. Available from: <https://arxiv.org/abs/1810.11654>

Goldbaum M, Moezzi, S, Taylor, A, Chatterjee, S. 1996. Automated diagnosis and image understanding with object extraction, object classification, and inferencing in retinal images. In: The 3rd IEEE International Conference on Image Processing, 19-19 September 1996. Switzerland: IEEE, pp.695-698.

Lam, P. 2018. What to know about MRI scans? [Online]. [Accessed 24 July 2018]. Available from: <https://www.medicalnewstoday.com/articles/146309>

Karagiannakos, S. 2019. the process of implementing FCN. [Online]. [Accessed 25 January 2019]. Available from: https://theaisummer.com/Semantic_Segmentation/

Ciresan, D.C, Giusti, A, Gambardella, and L.M, Schmidhuber, J. 2012. Deep neural networks segment neuronal membranes in electron microscopy images. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, 25-25 January 2012. Lake Tahoe, Nevada: Curran Associates Inc., pp.2843–2851.

Saha, S. 2018. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [Online]. [Accessed 15 December 2018]. Available from: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Gao, X, Wang, X.L. 2019. Advances in the application of deep learning in medical imaging. Journal of Medical Sciences. 46(3), pp.408-413.

Shukla, L. 2019. the structure of simple neural network. [Online]. [Accessed 19 November 2019]. Available from: <https://www.kdnuggets.com/2019/11/designing-neural-networks.html>

Hardesty, L. 2017. Explained: Neural networks. MIT News Office. [Online]. [Accessed 14 April 2017]. Available from: <http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>

Mehta, R and Arbel, T. 2018. the segmentation process of 3D U-Net. [Accessed 26 January 2018]. Available from: https://link.springer.com/chapter/10.1007/978-3-030-11726-9_23

Resuly. 2018. 3D Convolution Layer. [Online]. [Accessed 18 May 2018]. Available from: <http://resuly.me/2018/05/18/3d-convolutional-networks-for-traffic-forecasting/>

Balachandran, S. 2020. Max Pooling Layer. [Online]. [Accessed 30 March 2020]. Available from: <https://dev.to/sandeepbalachandran/machine-learning-max-pooling-with-color-images-2i21>

Oldpan. 2018. Up-Convolution Layer. [Online]. [Accessed 7 April 2018]. Available from: <https://oldpan.me/archives/upsample-convolve-efficient-sub-pixel-convolutional-layers>

Rosebrock, A. 2016. example of IoU and function of IoU. [Online]. [Accessed 7 November 2016]. Available from: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

Hou, J. L. 2020. valid convolution, same convolution and full convolution. [Online]. [Accessed 2 February 2016]. Available from: https://blog.csdn.net/weixin_40519315/article/details/104469365

Zeiler, M. D, Fergus, R. 2013. Visualizing and Understanding Convolutional Networks. [Online]. [Accessed 12 November 2013]. Available from: <https://arxiv.org/abs/1311.2901>

Agravat, R. R, Raval, M. S. 2020. Brain Tumor Segmentation and Survival Prediction. [Online]. Gujarat: Springer Cham. [Accessed 19 May 2020]. Available from: https://doi.org/10.1007/978-3-030-46640-4_32

Birkfellner, W, Figl, M, Furtado, H, Renner, A, Hatamikia, S and Hummel, J. 2108. Multi-Modality Imaging: A Software Fusion and Image-Guided Therapy Perspective. [Online]. [Accessed 17 July 2018]. Available from: <https://doi.org/10.3389/fphy.2018.00066>

Rorden, C. 2002. original image vs. normalized image. [Online]. [Accessed 13 October 2002]. Available from: <https://people.cas.sc.edu/rorden/anatomy/home.html>

Grégoire, N and Bouillot, M. 1998. Hausdorff Distance between convex polygons. [Online]. [Accessed Fall 1998]. Available from: <http://www-cgri.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html>

Rosebrock, A. 2016. example of IoU and function of IoU. [Online]. [Accessed 7 November 2016]. Available from: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>