An efficient deep learning approach for brain tumor detection using 3D convolutional neural network

by

Syed Muaz Ali
17201014

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

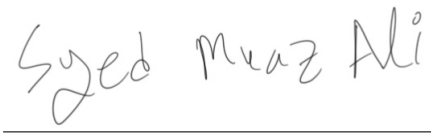
Department of Computer Science and Engineering
Brac University
September 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Syed Muaz Ali

17201014

Approval

The thesis/project titled "An efficient deep learning approach for brain tumor detection using 3D convolutional neural network" submitted by

1. Syed Muaz Ali (17201014)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 19, 2022.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Ashraful Alam

Assistant Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Program Coordinator:
(Member)

Dr. Golam Rabiul Alam

Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Abstract

In medical application, deep learning-based biomedical pixel-wise detection through semantic segmentation has provided excellent results and proven to be efficient than manual segmentation by human interaction in various cases. A well-known and widely used architecture for biomedical segmentation is U-Net. In this work, a convolutional neural architecture based on 3D U-Net but with fewer parameters and lower computational cost is used for pixel-level detection of brain tumor through semantic segmentation. The proposed model is able to maintain a very efficient performance and provides better results in some cases compared to conventional U-Net, while reducing memory usage, training time and inference time. BraTS 2021 dataset is used to evaluate the proposed architecture and it is able to achieve Dice scores of 0.9105 on Whole Tumor(WT), 0.884 on Tumor Core(TC) and 0.8254 on Enhancing-Tumor(ET) on the testing dataset.

Acknowledgement

Firstly, by the grace of the great Allah for whom my thesis have been completed without any issue.
Secondly, I'm very grateful to my supervisor Dr. Md. Ashraful Alam Sir for guiding me during the research.
We would like to thank Chris Rorden(rordenlab) on Github for building the MRIcroWeb which helped us to visualize the 3D volumes.

Table of Contents

Declaration i

Approval ii

Abstract iii

Acknowledgment iv

Table of Contents v

List of Figures vii

List of Tables ix

Nomenclature x

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Research Problem . . . 1
- 1.3 Research Objectives . . . 2

2 Literature Review 3

3 Implementation 5
- 3.1 Dataset Description . . . 5
- 3.2 Data Preprocessing . . . 5
- 3.3 Data Augmentation . . . 6
- 3.4 3D Separable Convolutions . . . 6
- 3.5 Mixed Separable Convolutions . . . 7
- 3.6 Squeeze-and-Excitation block . . . 8
- 3.7 Attention Gates . . . 9
- 3.8 Stem Blocks . . . 10
- 3.9 Modified Convolution Blocks . . . 11
- 3.10 Our Proposed model Architecture . . . 12
- 3.11 U-Net Architecture . . . 13
- 3.12 Deep-Supervision . . . 14
- 3.13 Transfer learning and Fine-tuning . . . 15
- 3.14 Activation functions . . . 15

4 Expirements and Results **17**
4.1 3D Visualization 17
4.2 Measurement 18
4.3 Loss Functions 18
4.4 Training and Results on U-Net 18
4.5 Training and Results on Proposed Model 20
4.6 Comparison of Proposed Model with U-Net 31

5 Conclusion & Future Work **35**
5.1 Conclusion 35
5.2 Future Work 35

Bibliography **38**

List of Figures

3.1 Example data with labels 5
3.2 Data Augmentation 6
3.3 Separable Convolutions 7
3.4 Mixed Convolutions 8
3.5 Squeeze-and-Excitation block 9
3.6 Modified Attention Gate 10
3.7 Stem Blocks 11
3.8 Modified Convolution Block 12
3.9 Our proposed model 13
3.10 U-Net Architecture 14
3.11 U-Net Convolution Block 14
3.12 Swish Activation Function 16
3.13 Smish Activation Function 16

4.1 Example 3D Visualization 17
4.2 U-Net F1 Score during training 19
4.3 U-Net IoU Score during training 19
4.4 Prediction On U-Net (Left Ground Truth, Right Prediction) 20
4.5 Our Proposed Model with ReLU IoU Score during training 21
4.6 Our Proposed Model with ReLU F1 Score during training 21
4.7 Our Proposed Model with Smish IoU Score during training 22
4.8 Our Proposed Model with Smish F1 Score during training 22
4.9 Sample prediction between Our proposed model with ReLU and Smish (Left GT, Middle w/ ReLU, Right w/ Smish) 23
4.10 Our Proposed Model with ReLU (Separable Convolution 2) F1 Score during training 24
4.11 Our Proposed Model with ReLU (Separable Convolution 2) IoU Score during training 24
4.12 Sample prediction between Our proposed model with Separable Convolution 1 and 2 (with ReLU) (Left GT, Middle SC 1, Right SC 2) . 25
4.13 Patch-wise pre-training and transfer learning 26
4.14 IoU Score during Training on patches (LR = 0.001) 27
4.15 F1 Score during Training on patches (LR = 0.001) 27
4.16 F1 Score during Training on patches (LR = 0.0001) 28
4.17 IoU Score during Training on patches (LR = 0.0001) 28
4.18 F1 Score during Training pre-trained model on 128x128x128 images . 29
4.19 IoU Score during Training pre-trained model on 128x128x128 images 29

4.20 Sample prediction between predicting using patches (Middle) and our final model with patch-wise pre-training and transfer-learning model (Right) (Left GT) 30
4.21 Box-plot of F1 (Dice) Scores on testing dataset between Our Proposed Model and U-Net for NET/NCR, ED and ET 33
4.22 Box-plot of F1 (Dice) Scores on testing dataset between Our Proposed Model and U-Net for ET, WT and TC 33
4.23 Sample predictions between U-Net (Middle) and Our Proposed Model (Right) (Left GT) 34

List of Tables

4.1 U-Net Results On Testing Dataset 19
4.2 Our Proposed Model with ReLU results on different classes on testing dataset 22
4.3 Our Proposed Model with Smish results on different classes on testing dataset 23
4.4 Our Proposed Model with ReLU (Separable Convolution 2) results on different classes on testing dataset 25
4.5 Results on testing dataset by predicting using patches 30
4.6 Results on testing dataset through patch-wise pre-training and transfer-learning 30
4.7 Comparison of Accuracy on U-Net and Our Proposed Model 31
4.8 Comparison of IoU on U-Net and Our Proposed Model 31
4.9 Comparison of F1 (Dice Score) on U-Net and Our Proposed Model . 31
4.10 Comparison of Precision on U-Net and Our Proposed Model 32
4.11 Comparison of Recall/Sensitivity on U-Net and Our Proposed Model 32
4.12 Comparison of Specificity on U-Net and Our Proposed Model 32
4.13 Comparison of U-Net and Our Proposed Model 32

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

σ Sigmoid

CNN Convolutional Neural Network

ED Edema

ET Enhancing Tumor

GT Ground Truth

NCR Necrosis

NET Non-Enhancing Tumor

$ReLU$ Rectified Linear Unit

TC Tumor Core

WT Whole Tumor

Chapter 1

Introduction

1.1 Background

Brain tumors are a major health issue affecting a large number of people's lives worldwide. Early detection of these tumors can minimize the health hazard and its fatal consequences. Imaging methods, most commonly Magnetic Resonance Imaging or MRI is used for extracting valuable information to identify the exact location, size, and types for clinicians and surgeons. Hence, we propose to use a 3D U-Net based computationally efficient architecture for detecting brain tumor on pixle-level through segmentation from multiple parametric MRI (MP-MRI) images to identify the essential characteristics of brain tumors and in order to attain a precise diagnosis for achieving the most appropriate treatment protocol.

The recent technologies have achieved considerable progress on medical imaging using deep learning techniques. However, there exists various challenges in the field of computer vision while extrapolating accuracy depicting the exact region from the automated 3D medical image of MRI.

The BraTS[30][9][13] challenge has achieved its tenth anniversary on year 2021 where various segmentation algorithms has been dedicated to facilitate the task of precise brain tumor segmentation. The challenge is solely based on the performance of precisely segmenting brain tumor.

U-Net[10] is a great convolutional neural network architecture for biomedical segmentation and is often adapted for better presentation in BraTS, and one of U-Net's improved versions, nnU-Net[19], became the champion in BraTS 2020 challenge, thus proves the outstanding presentation of U-Net based architectures. Therefore, in order to properly segment biomedical images through deep learning, adapting and improving U-Net architecture is an excellent way to get better results.

1.2 Research Problem

One of the most fatal cancers is Brain tumor. Around 83,570 people would be diagnosed with brain tumor and other type of central nervous system tumor, 18,600 people out of them would die due to the illness [31]. Typically diagnosing brain tumor involves neurological exam, brain scan (CT scan, MRI, PET or an angiogram)

and biopsy. These tests require expert operators to perform and are prone to human error [1]. In order to increase the survival rate of patients, it is important to diagnose brain tumor at a very early phase. Also, the high influx of brain tumor patients increases the cost of diagnosis due to high demand. To solve these issues, an automated diagnosis method is required with high sensitivity and specificity. Currently, MRI scans provides a high sensitivity of (70-100%) and specificity of (95-100%) without effecting the patient with ionizing radiation [6].

CNN architectures based on U-Net have provided great results for detecting brain tumor by each pixel from MRI images through semantic segmentation. However, U-Net based architectures are mostly computationally expensive and require powerful hardware in order to run the models based on millions of parameters and using 3D convolutional neural network increases the parameter count compared to 2D convolutional neural network. Thus, we propose an efficient CNN architecture similar to U-Net but computationally less expensive and requires less than a million parameter to run.

1.3 Research Objectives

This research aims to develop an efficient U-Net like architecture for pixel-level detection of brain tumor through semantic segmentation. The objectives of the research are :

- Build an efficient architecture similar to U-Net for pixel-level brain tumor detection through semantic segmentation
- Reduce computational cost, training time, inference time and memory usage
- Compare the results with conventional U-Net

Chapter 2

Literature Review

Squeeze U-Net[28] architecture is a computationally efficient architecture based on U-Net and inspired by SqueezeNet[12]. The idea of SqueezeNet is to reduce the parameters without decreasing accuracy. The authors of SqueezeNet first introduced a module called 'Fire' module that involves two operation. Both SqueezeNet and Squeeze U-Net were implemented using 2D convoluitonal neural networks. On SqueezeNet, the 'Fire' module performs two operations where one is 'Squeeze' and the other is 'Expand'. On the 'Squeeze' operation, a 1x1 convolution operation is performed. The strategy behind 1x1 convolution is replacing 3x3 convolution to reduce the parameter. Thus, through a dimensionality reduction operation, the number of channels in the layer get reduced thus the term 'Squeeze' is used. The other operation perfomred on the 'Fire' module is called 'Expand'. This operation involves two convolution operations of 1x1 and 3x3 kernels, an increased number of features compared to the 'Squeeze' operation, thus the term 'Expand' is used. The authors found that, compared to AlexNet[5], SqueezeNet has 50 times reduced model size and manages to perform as well or better in terms of accuracy. The Squeeze U-Net was developed being inspired by the performance of SqueezeNet. The model size of Squeeze U-Net is reduced by 12 times, the inference time of the model is also improved by 17% and training time is also improved by 52% compared to conventional U-Net while maintaining similar accuracy. The authors have used 'Fire' module similar to SqueezeNet and also implemented 'Transposed Fire Module' in the network. On the 'Fire' module of the network, a 1x1 convolution is initially performed. Then, on the features, another 1x1 convolution and one 3x3 convolution operation, both with increased number of features are performed. The authors have used 'Transposed Fire Module' on the decoders of the network, which is similar to the original 'Fire' module, consisting of operations 'Squeeze' and 'Expand'. The module initially performs a transposed convolution operation of kernel size 1x1 (Squeeze) and on the output features of the operation, the module performs one 1x1 transposed convolution and one 2x2 transposed convolution and concatenates the outputs (Expand). The authors have used convolution operations with strides set to 2 for performing down-sampling operations as they mentioned that it improves the network's expressiveness. The authors have evaluated the performance by evaluating through CamVid[2][3] dataset which consists of 5 classes, which are Building, Tree, Sky, Car and Road. The authors have measured the performance of their models using True Positive pixels, False Positive pixels and Intersection over Union (IoU). Squeeze U-Net and conventional U-Net sc mean IoU of 0.689 and 0.639

for the class Building, 0.37 and 0.403 for the class Tree, 0.842 and 0.879 for the class Sky, 0.427 and 0.628 for the class Car, 0.751 and 0.8 for the class Road, resulting an average IoU of all the 5 classes of 0.616 for Squeeze U-Net and 0.670 for the U-Net. Again, Squeeze U-Net gained 2% to 3% less accuracy compared to U-Net in terms of the five classes for true positive pixels, where Squeeze U-Net gained an accuracy of 78% and U-Net gained an accuracy of 86.9%. Furthermore, U-Net gained an accuracy approximately 5% more than U-Net for the class Building, achieving 83.3% accuracy, while Squeeze U-Net gained 78.5% accuracy. Moreover, U-Net gained 20% more accuracy for the class Tree where Squeeze U-Net gained an accuracy of 51.1% and U-Net gained an accuracy of 72.6%. Squeeze U-Net gained 71.1% accuracy for the class Car while U-Net gained an accuracy of 84.5%, around 20% less accuracy for Squeeze U-Net. The authors have stated that U-Net usually has less false positive pixels compared to U-Net except for the class Building where U-Net has 17% more false positive pixels compared to Squeeze U-Net and the class Tree where U-Net has 24% more false positive pixels than Squeeze U-Net. For the class Road, Squeeze U-Net has 31.3% false positive pixels where U-Net has 23.4% false positives, around 8% more for Squeeze U-Net and for the class Sky, U-Net has 1.5% less false positive pixels. Finally, the authors stated that Squeeze U-Net requires 12 times less paramaters compared to U-Net to operate and 3 times less MACs. The training time for Squeeze U-Net is around 69% of U-Net and in terms of inference time, Squeeze U-Net performs 17% faster.

Wei et al.[23] proposed the idea of implementing 3D separable convolutions for brain tumor segmentation. The authors mentioned that using 2D convolution instead of 3D on volumetric data can reduce the ability to acquire information for CNN architectures. However, 3D convolutions require more computational cost compared to 2D. Thus the authors implemented a U-Net based architecture to use 3D separable convolutions to segment brain tumors. The authors have evaluated their results on BraTS 2018 dataset. On the implemented U-Net like architecture, the authors have used convolution blocks called S-3D Convolution Block. On the block, the authors have followed the idea by Xie et al. [21] to use spatio-temporal-separable 3D convolutions to replace the 3D convolutions. On the prposed S3D block, the authors have implemented similar blocks to residual Inception block but with separable convolutions. On the final S3D U-Net, the authors have used S3D blocks with and without residual connections. The architecture takes input size of 128x128x128. There are five levels in the contracting part of the architecture and all the levels except the first level uses S3D blocks. The authors have used F1 Loss function for training the model and for evaluation, they have used sensitivity, specificity, F1 score and the Hausdorff distance. The authors have compared the performance of the proposed model with S3D block with U-Net on the BraTS 2018 dataset through cross validation with five-fold. The proposed model scored a F1 score of 0.73953, 0.88809, 0.84419 and the U-Net without the S3D block scored F1 scores of 0.68428, 0.89912, 0.86772 on Enhancing Tumor (ET), Whole Tumor (WT) and Tumor Core (TC). Again, on the Brats 2018 validation dataset, the proposed model scored F1 scores of 0.74932, 0.89353 and 0.83093 on ET, WT and TC. Finally, on the testing dataset, the model scored F1 scores of 0.68946, 0.83893 and 0.78347 on ET, WT and TC.

Chapter 3

Implementation

3.1 Dataset Description

We used the BraTS 2021 dataset to evaluate the models. The dataset contains 1251 samples where the tumor regions are labeled. The regions include the following categories: 1. Enhancing Tumor (ET), 2. Non-Enhancing Tumor (NET/NCR), 3. Edema (ED). Each of the scans includes four channels T1, Flair, T2, T1CE. The dimension of each of the channel is 240x240x155. Figure 3.1 shows the T1, T2, Flair and T1CE scans and the labels for all the classes except the background.

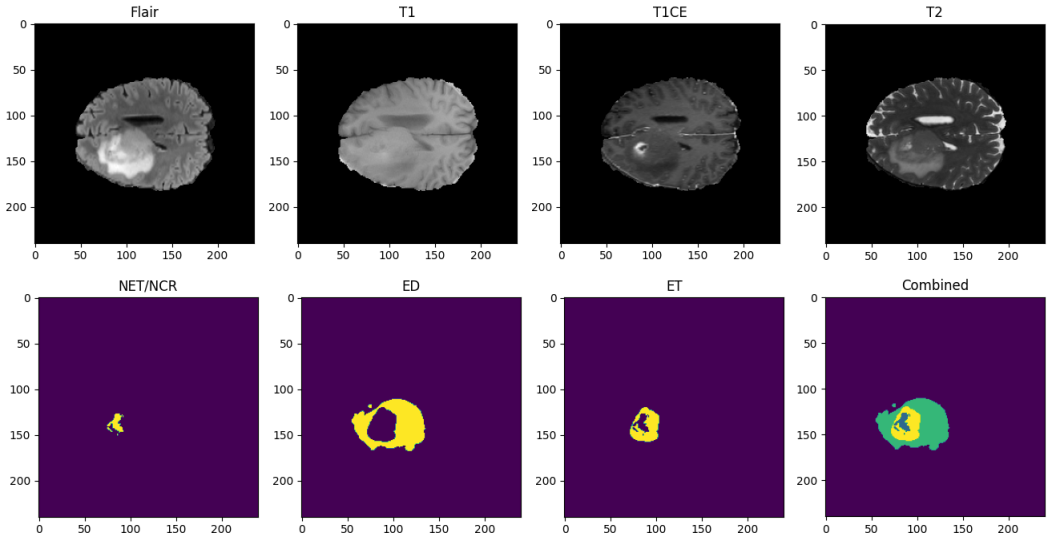


Figure 3.1: Example data with labels

3.2 Data Preprocessing

We divided data preprocessing into two steps. At first, we scaled the Nifti image values' between 0 to 1. For the segmented masks, the labels in the dataset were set as 0 for Background, 1 for Non-Enhancing Tumor/Necrosis, 2 for Edema but 4 for Enhancing-Tumor. We assigned the values for the label 4 Enhancing Tumor to label 3 so the classes can be in an order of 0,1,2 and 3 to train the models. Furthermore, we cropped the the images to 128x128x128 to reduce the number of zero values and to fit into the memory of our GPU. We then used T1CE, T2 and Flair images to

create a four-dimensional Numpy array and saved images where the mask contains at least 1% of it as tumor data. After saving the images, 80% training, 10% validation and 10% testing split were randomly created from the dataset.
On the second step of data preprocessing, we extracted patches of dimenson 64x64x64 with no overlap from each of the images from training set and validation set. Each of the 128x128x128 images resulted in total 8 patches of dimension 64x64x64. We saved the patches where the masks contain at least 1% data of either Enhancing-tumor, Tumor-core or Edema.

3.3 Data Augmentation

Data augmentation was performed on the training dataset of 64x64x64 patches where we randomly rotated and flipped each of the images using Scikit-learn and Numpy. Through data augmentation, there were total 16,536 images for training. Figure 3.2 demonstrates an example of data augmentation.

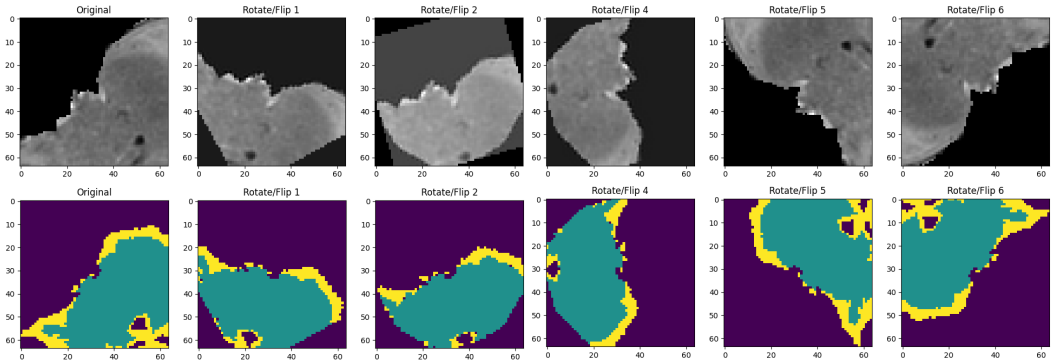


Figure 3.2: Data Augmentation

3.4 3D Separable Convolutions

In separable convolutions, one convolution operation is divided into multiple convolution to reduce the computational cost[23]. For example, dividing 3x3x3 convolution into two convolution operations of 3x3x1 and 1x1x3 or 1x3x3 and 3x1x3 convolutions. We tested 2 different separable convolution operations where we divided convolution operation of NxNxN to NxNx1 and 1x1xN (separable convolution 1) and another convolution operations of 1xNxN and Nx1x1 (separable convolution 2). Figure 3.3 demonstrates the types of separable convolution operations used.

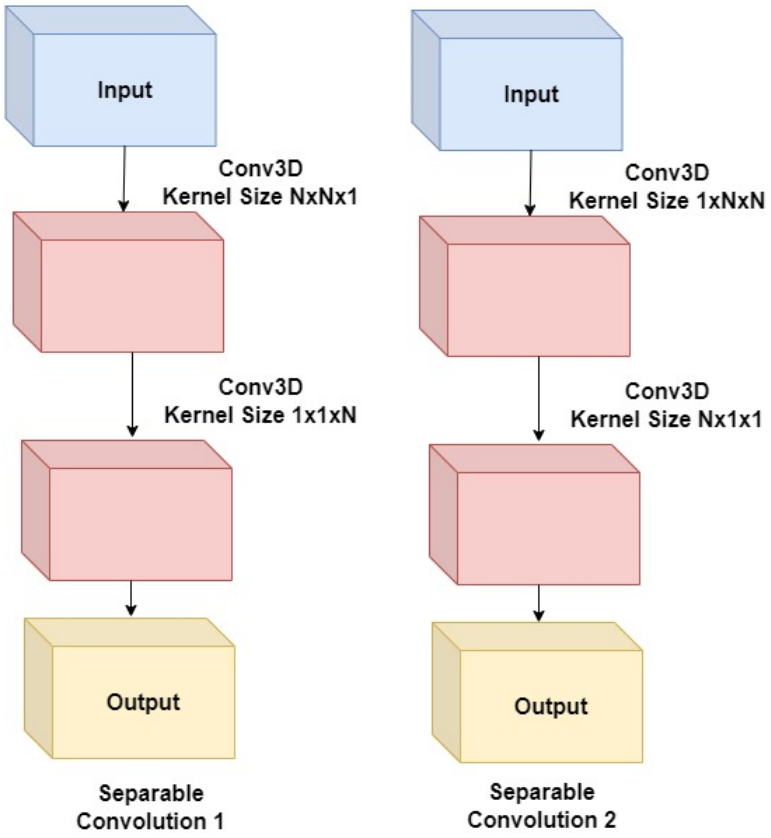


Figure 3.3: Separable Convolutions

3.5 Mixed Separable Convolutions

The authors in the paper[26] proposed the idea of concatenating separate convolutions of different kernel sizes to improve the accuracy. We implemented similar method to concatenate the outputs of separable convolutions. However, unlike MixConv[26] where the features are split into separate groups, we performed convolution operations on the same features but with half the number of filters. Figure 3.4 demonstrates the mixed separable convolutions operation performed.

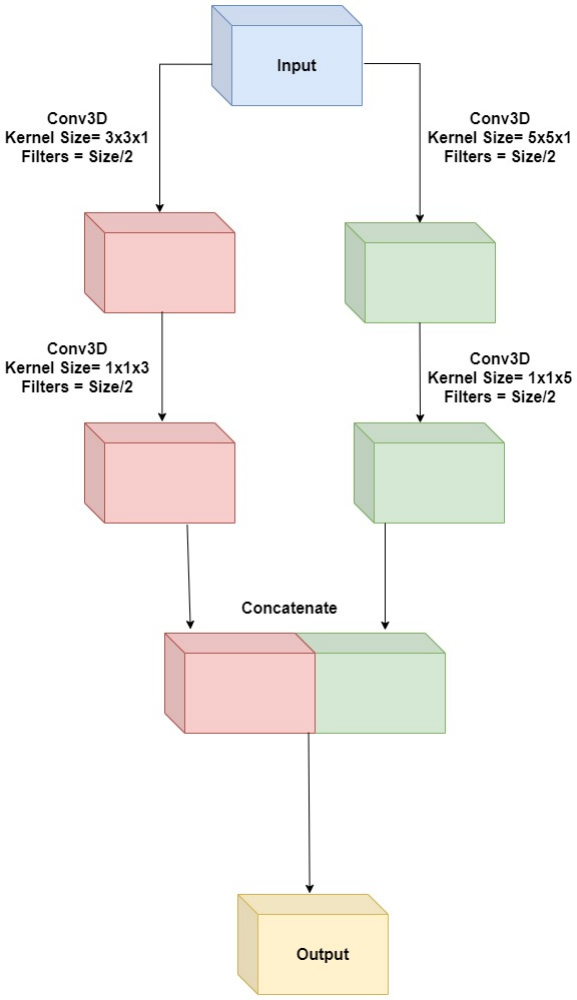


Figure 3.4: Mixed Convolutions

3.6 Squeeze-and-Excitation block

The Squeeze-and-Excitation blocks[18] takes convolution features as input and converts it into an one-dimension through global average pooling, thus the term 'Squeeze' is used. Furthermore, the global average pooling layer output is processed by a dense layer where ReLU activation for non-linearity to reduce channel complexity and a dense layer with a reduced number of units with Sigmoid function further processes the previous dense layer's output which helps the channel to have a smooth gating function. Finally, the convolution features are multiplied with the last fully-connected layers for recalibrating channel-wise feature responses. Jie et al.[18] found that, using Squeeze-and-Excitation blocks can further improve the accuracy of a CNN while not having high computational cost. Figure 3.5 shows the architecture of a Squeeze-and-Excitation block.

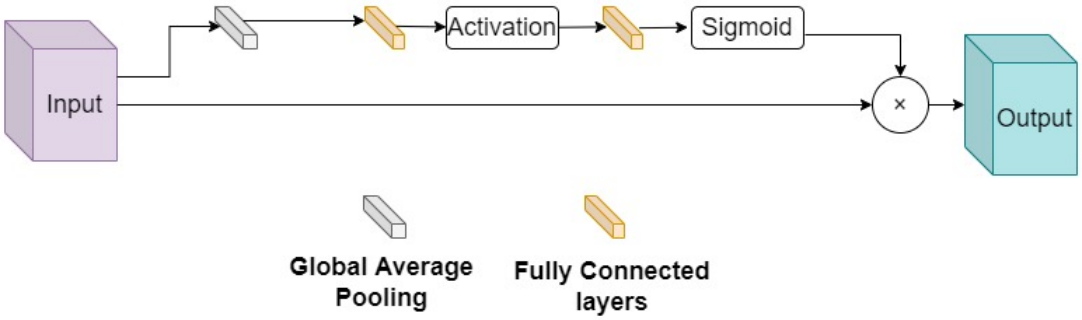


Figure 3.5: Squeeze-and-Excitation block

3.7 Attention Gates

According to Oktay et al.[20], attention gates can help to improve representation of salient features by reducing the irrelevant areas of an image. We slightly modified the attention gate to reduce computational cost by not doing any convolution operation on the features from the encoder block and the gating signal from the decoder. The features from the decoder block are up-sampled and added to the features of the encoder block, processed by an activation function and a convolution of kernel size 1x1x1 and feature size of 1 and finally processed by a Sigmoid activation function. The output is then multiplied with the input features from the encoder block through element-wise multiplication. Figure 3.6 shows the architecture of the modified attention gates.

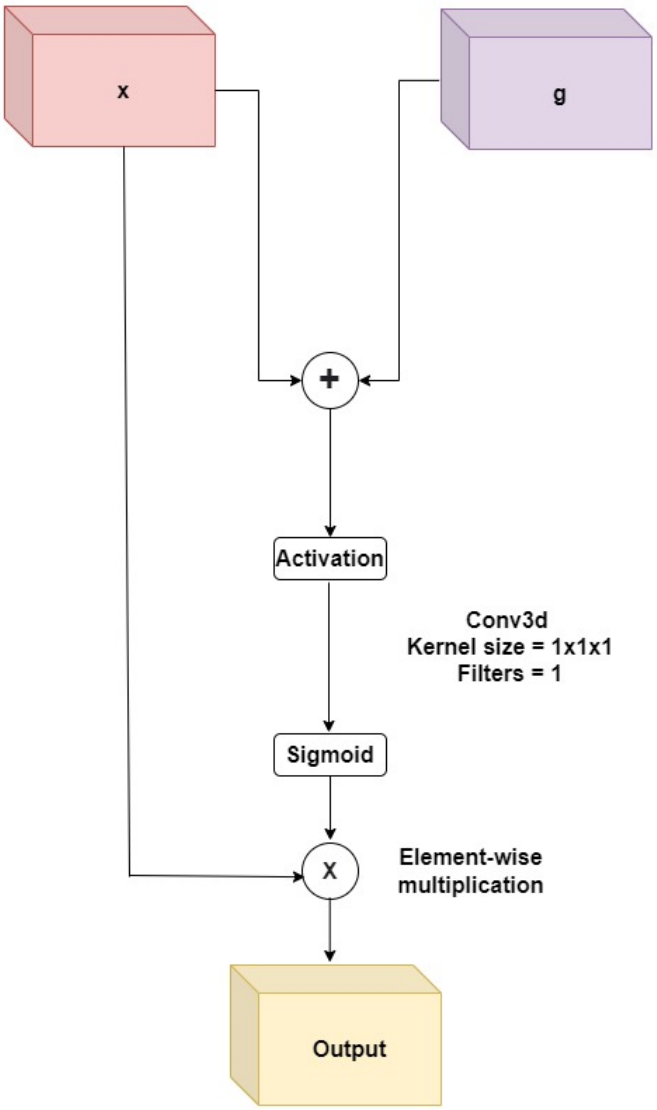


Figure 3.6: Modified Attention Gate

3.8 Stem Blocks

We implemented Stem Blocks similar to the initial block on ResNet[11]. Using convolution operation of kernel size 7x7x7 with stirdes set to 2 gave a decent improvement to training time and memory usage by reducing the computational cost. Furthermore, we modified the stem block based on ResNet-D by Tong et al.[24] by adding an additional average pooling layer followed by a convolution operation of kernel size of 1x1x1. According to the authors of ResNet-D, due to the strides of 2 on ResNet, some of the information on the images are ignored. However, on our proposed models, we used an average pooling operation with padding set to same and strides set to 1 and on the next convolution operation of kernel size 1x1x1, we set the strides to 2 for down-sampling to reduce computational cost. We showed the Stem Blocks in the Figure 3.7.

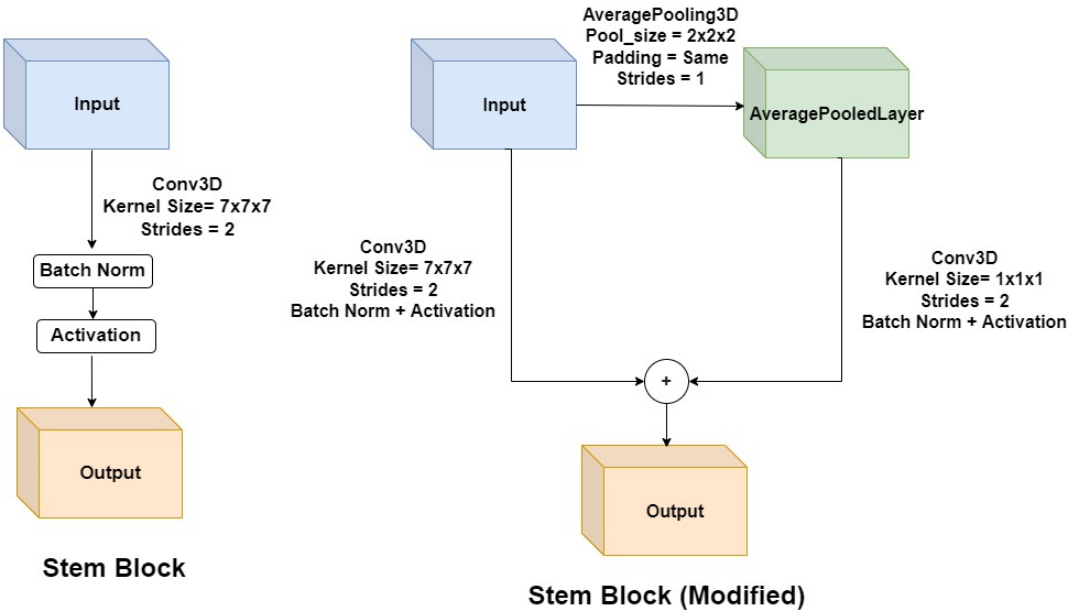


Figure 3.7: Stem Blocks

3.9 Modified Convolution Blocks

Figure 3.8 shows the modified convolution blocks that we have used in encoders, decoders and the bottleneck layers of our proposed model. The block consists of separable convolutions incorporating with mixed convolution. In the figure, we showed the separable convolution operations of kernel sizes of NxNx1 followed by a convolution operation of 1x1xN. A convolution operation of kernel size 1x1x1 is performed on the concatenated output from mixed convolution and added to the original input through element-wise addition, similar to residual-blocks to avoid issue with vanishing gradients and overfitting. A convolution operation is performed on input layer for identity mapping and dimensionality reduction or increasing to match with the output in order for addition. Finally the added output is processed through a squeeze-and-excitation block. Batch normalization, followed by a an Activation function was applied to each of the convolution operation.

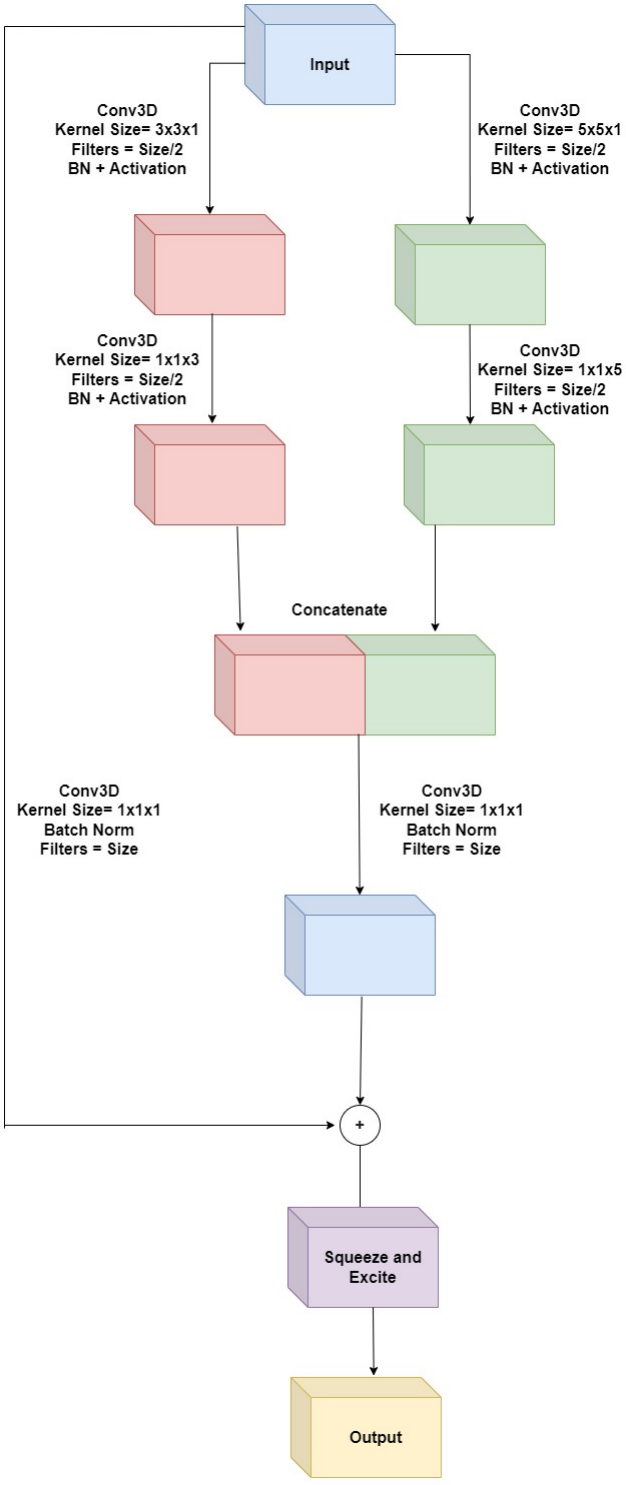


Figure 3.8: Modified Convolution Block

3.10 Our Proposed model Architecture

On our proposed model, we incorporated the modified convolution blocks, stem blocks and attention gates. The input size of our proposed model is 128x128x128x3 or 64x64x64x3 during training with the patches and the output is of 128x128x128x4 or 64x64x64x4. On the up-sampling blocks of our proposed model, the layers are

up-sampled by a size of 2x2x2 followed by a convolution operation of kernel size of 2x2x2, batch normalization and then activation. The encoder and decoder blocks use feature maps starting from 16 to 64 and the bottleneck layer uses filters of 256. However, compared to U-Net, the model is 3 encoder and decoder blocks while U-Net that we used to compare our results uses 4 encoder and decoder blocks. Figure 3.9 shows the architecture of our proposed model.

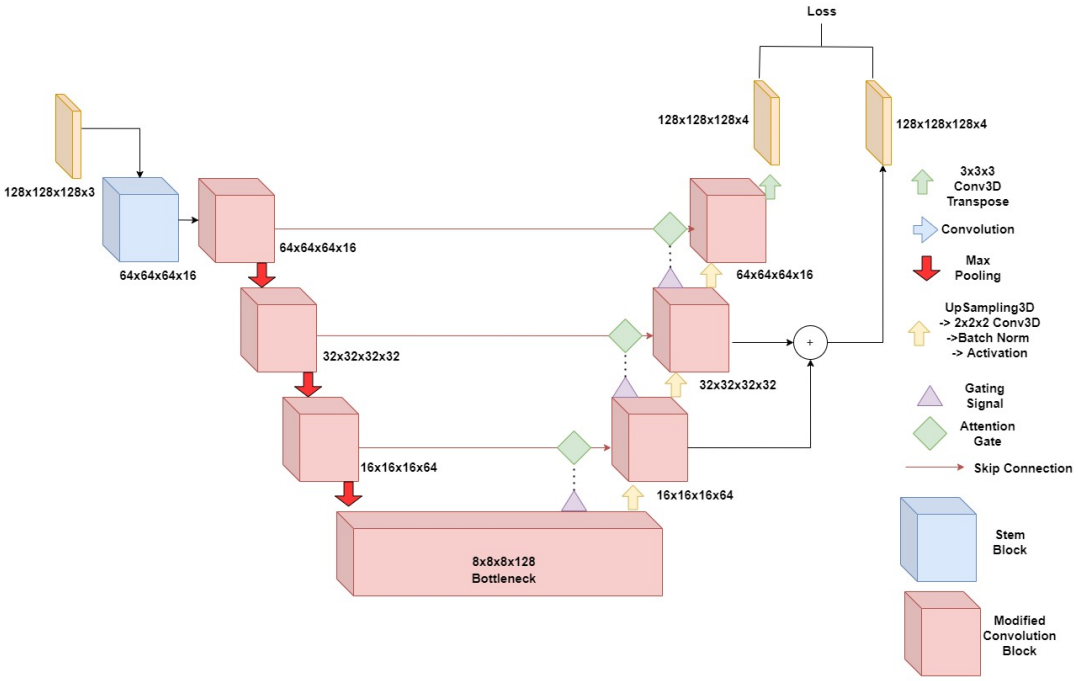


Figure 3.9: Our proposed model

3.11 U-Net Architecture

We used a conventional 3D U-Net where the encoders and decoders have 16,32,64,128 features and bottleneck has 256 features, following a similar pattern of our proposed model. On the convolution blocks, batch normalization was used after each of the convolution operation to avoid overfitting and improve generalization. Figure 3.10 demonstrates the U-Net architecture used to compare with our proposed model and Figer 3.11 demonstrates the convolution blocks.

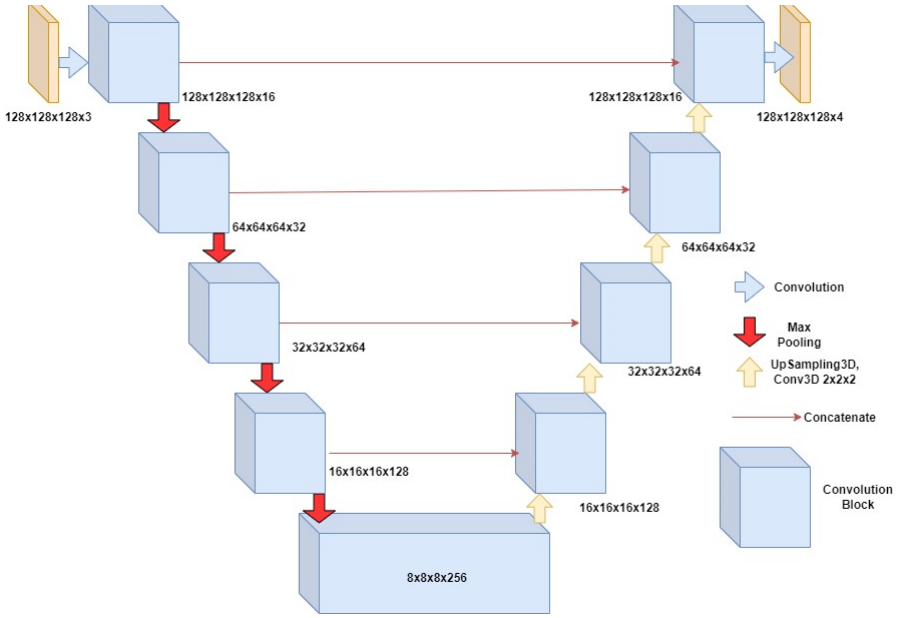


Figure 3.10: U-Net Architecture

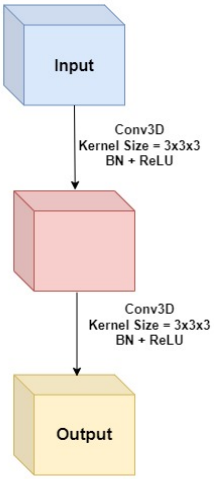


Figure 3.11: U-Net Convolution Block

3.12 Deep-Supervision

Originally proposed by Chen-Yu et al.[8], deep supervision can improve the accuracy of a deep-learning model through enforcing supervision for output and hidden layer. U-Net++[22] and U-Net3+[29] have implemented deep-supervision in their networks. On U-Net3+, according to the authors, from full-scale aggregated feature maps, deep-supervision can be used to learn hierarchical representation. When implementing deep-supervsion for our proposed model, up-sampling all the features from hidden layers of decoder block and performing convolution operation to reduce dimensionality to match output size was expensive in terms of memory usage. Again, doing a dimensionality reduction first on hidden layers to match the channel size of output layer and up-sampling didn't provide good results. In order to make

it computationally more efficient, on our proposed model, we performed convolution transpose operation of kernel size 2x2x2 on the first decoder block and added the output to the next decoder block, finally performing two additional convolution transpose operation on the added output to match the size of the output layer. Each of the convolution transpose operation would reduce the number of filters by half while increasing the size of the features. Finally, we calculated two different loss for the main output layer and the hidden output layer.

3.13 Transfer learning and Fine-tuning

Transfer learning is the process where a pre-trained model is trained on a different dataset by loading the weights as the model is already familiar with a similar task and can provide better results on the different dataset. Fine-tuning is where specific layers are frozen or set to non-trainable while training the other layers for different dataset. Different CNN architectures require different layers to be frozen to provide the best result. Mina et al.[27] have demonstrated the results of freezing layers of different blocks on the U-Net architecture. According to their work, freezing the bottleneck block's layers would provide similar results as training the whole network while reducing the number of parameters by around half. We followed similar methodology and during transfer learning, we froze the layers of the bottleneck layer and the total parameter was 627,155 and the trainable parameters were 417,555, thus reduced computational cost.

3.14 Activation functions

The performance of our proposed models were evaluated using using different activation functions or combined both on different layers of the network. We used Swish[15], Smish[34] and ReLU[17] activation functions to make the comparison to find the best combination for achieving highest accuracy while maintaining efficiency.

$$f(x) = x \cdot \sigma(x) \tag{3.1}$$

Equation 3.1 shows the equation of Swish activation function. The authors of Swish activation function found that using Swish instead of ReLU activation function can improve accuracy and provide strong regularization effects as the function is being bounded below. Figure 3.12 shows the Swish activation function.

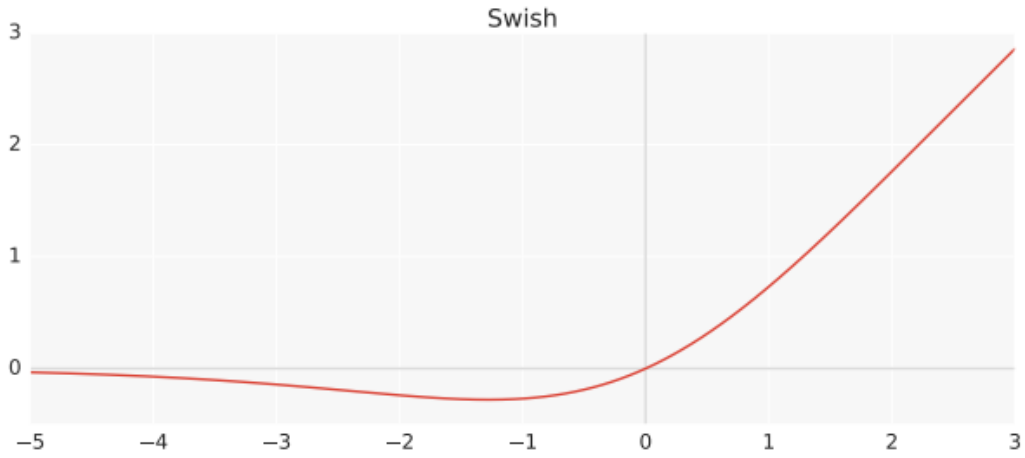


Figure 3.12: Swish Activation Function

$$f(x) = x \cdot \tanh(\ln(1 + \sigma(x))) \tag{3.2}$$

A novel activation function called Smish was proposed by Xueliang et al.[34]. According to the authors, Smish activation function can provide better accuracy compared to ReLU and Swish. The function provides regularization effects for inputs less than zero and has the non-monotonic abilities of the Logish[32] activation function. The function is also similar to Mish[25] activation function, having no upper bound but a lower bound. Equation 3.2 shows the equation and Figure 3.13 shows the Smish activation function.

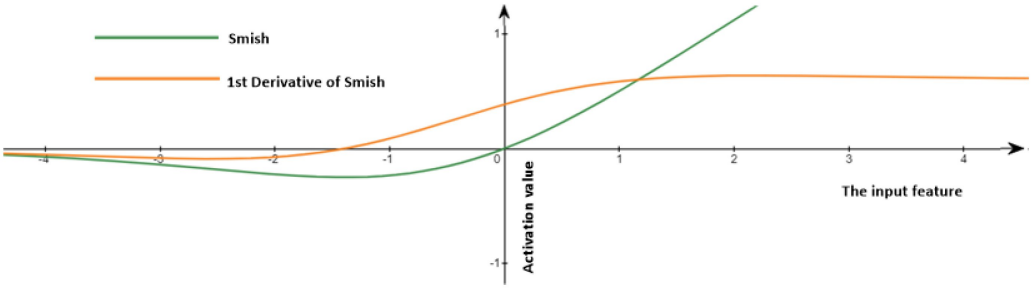


Figure 3.13: Smish Activation Function

$$f(x) = max(0, x) \tag{3.3}$$

ReLU or Rectified Linear Unit is a non-linear function which takes the input and calculates the maximum between the input and zero. If the input is positive, then the function will return the input itself otherwise it will return zero. Equation 3.3 shows the equation of it. ReLU is simple and computationally less expensive compared to bounded by below activation functions but due to producing zero for all negative values can lead to a term called 'dead neurons' and cause vanishing gradient issue, leading to poor performance of the model.

Chapter 4

Expirements and Results

4.1 3D Visualization

In order to visualize a prediction in 3D, we converted the output of the model to a Nifti file. Then we used MRIcroWeb by Rordenlab to render the volume using WebGL. In order to crop or slice an image to view the Enhancing-Tumor or Non-Enhancing Tumor/Necrosis, we manually cropped the images in Numpy then saved to Nifti file. Figure 4.1 shows an example 3D visualization with the regions of the tumor marked as Edema, Non-Enhancing Tumor/Necrosis and Enhancing Tumor.

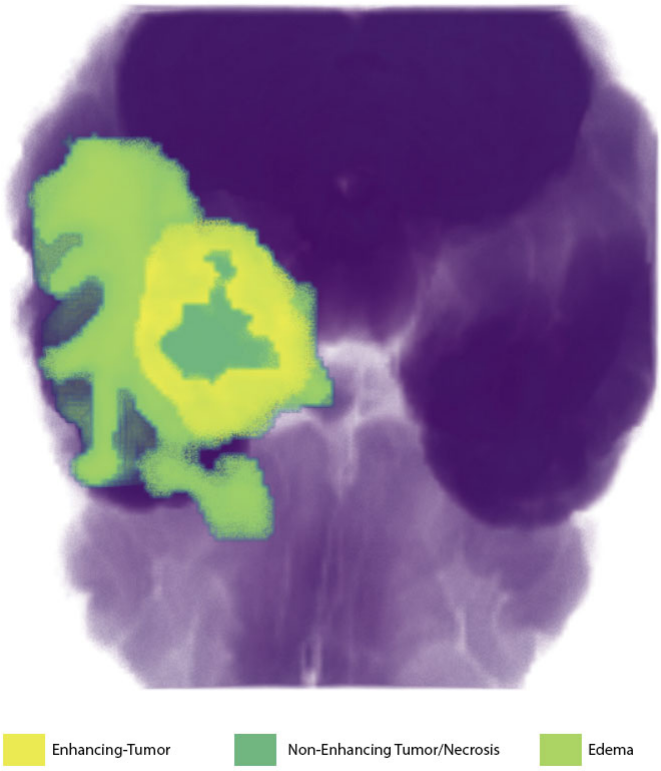


Figure 4.1: Example 3D Visualization

4.2 Measurement

We evaluated the results of all the models using Accuracy, Dice score (F1), IoU, precision, sensitivity and specificity. For each of the predictions, we calculated TP (True Positive), FP (False Positive), TN (True Negative) and FN (False Negative) based on each of the pixel-level detection of brain tumor from testing dataset. Our final results show the mean results for all the predictions on the testing dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Dice = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.2}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{4.6}$$

4.3 Loss Functions

We combined Dice loss[16] and Categorical Focal loss[14] as a loss function to train the models. Equation 4.7 shows the equation for Categorical Focal loss, Equation 4.8 shows the equation for calculating Dice loss and Equation 4.9 shows the combined loss function[33] used during training.

$$\mathcal{L}_{\text{Focal}}(gt, pr) = -gt \cdot \alpha \cdot (1 - pr)^{\gamma} \cdot \log(pr) \tag{4.7}$$

$$\mathcal{L}_{\text{Dice}}(precision, recall) = 1 - (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{4.8}$$

$$\mathcal{L} = \mathcal{L}_{\text{Focal}} + \mathcal{L}_{\text{Dice}} \tag{4.9}$$

4.4 Training and Results on U-Net

We trained the U-Net model with learning rate set to 0.0001 and Adam[7] optimizer with a batch size of 2. It was trained on 128x128x128 images with loss function set to combined Dice loss and Focal loss. The validation scores did not improve after 81th epoch reaching validation IoU score 0.7766140699 and F1 score 0.8506705761 and we stopped the training at 140th epoch. We showed the F1 and IoU scores during training on Figure 4.2 and Figure 4.3. We showed the results on testing dataset for Non-enhancing Tumor/Necrosis, Edema and Enhancing tumor on Table

4.1. We also showed a sample prediction on U-Net on Figure 4.4 where the left image is ground truth and the right image is prediction from U-Net.

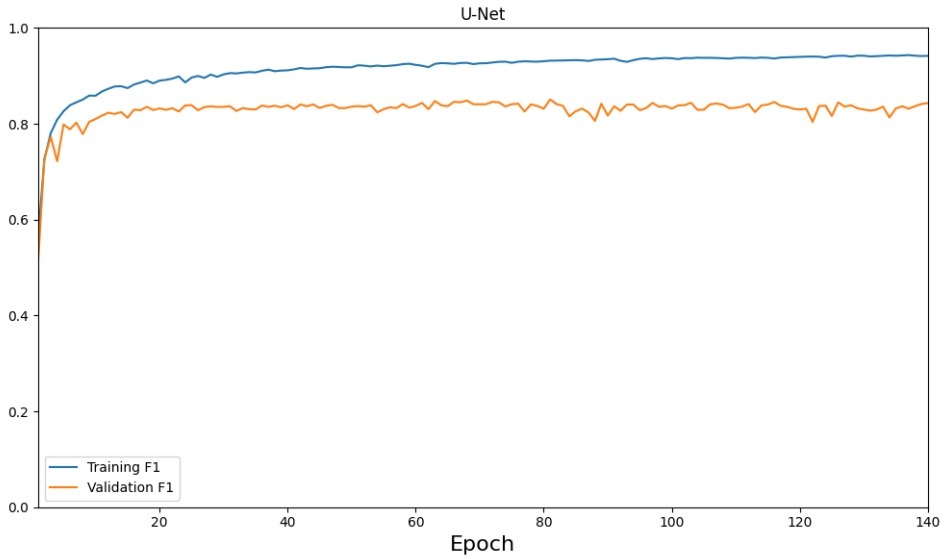


Figure 4.2: U-Net F1 Score during training

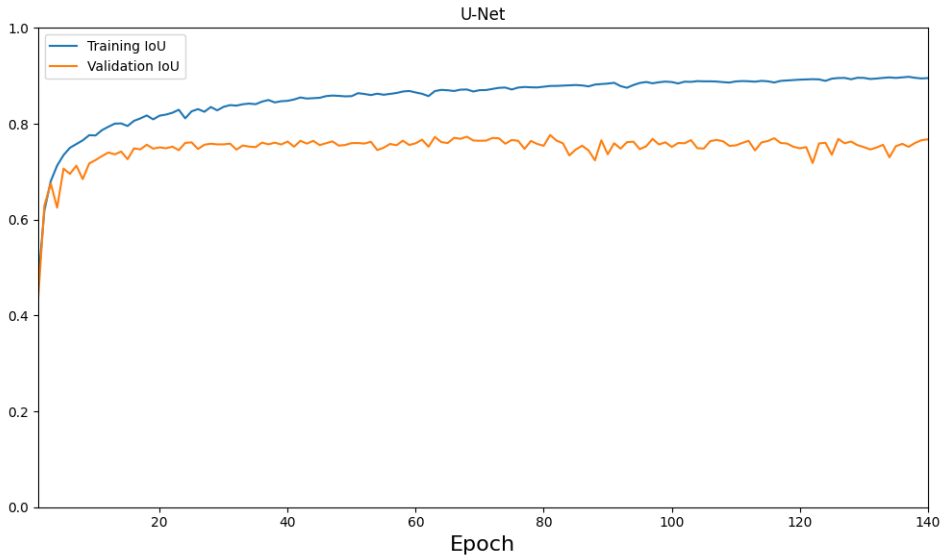


Figure 4.3: U-Net IoU Score during training

Table 4.1: U-Net Results On Testing Dataset

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9976	0.6869	0.7654	0.9986	0.8195	0.7596
ED	0.9916	0.7141	0.8142	0.9958	0.8421	0.8202
ET	0.998	0.7484	0.8273	0.9989	0.8432	0.8333

Table 4.1 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for the model U-Net

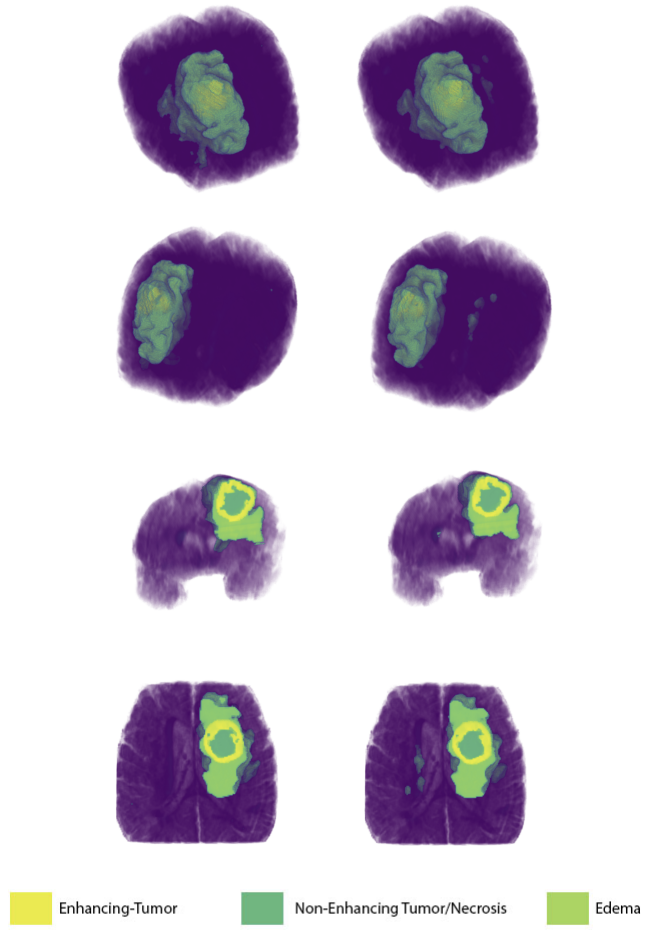


Figure 4.4: Prediction On U-Net (Left Ground Truth, Right Prediction)

4.5 Training and Results on Proposed Model

Firstly, we trained our proposed model with the unmodified Stem block from Figure 3.7, Separable Convolution 1 from Figure 3.3 and without any deep-supervision. We tested how ReLU and Smish activation functions perform on the model. In order to achieve this, we used ReLU activation function on the layers that require non-linearity without replacing the existing Sigmoid or Softmax functions. Secondly, we replaced all the ReLU activation functions with Smish activation function and trained the models. The models were trained on 128x128x128 images with learning rate set to 0.0001, Adam optimizer, batch size of 2 and we used L2 Regularization[4] with a weight decay of 0.0005. Our proposed model with ReLU activation validation scores didn't improve after 166th epoch reaching validation IoU score 0.7535068393 and F1 score 0.8332134485 and the model with Smish didn't improve after 188th epoch reaching validation IoU score 0.7605672479 and F1 score 0.8377211094. Both of the models were trained using Tensorflow with mixed precision enabled to use both 16-bit and 32-bit floating point values, however we found that using Smish function tends to consume more memory. We manually allocated 2048MB video memory from our GPU to train the models but Smish required more and ReLU worked fine without causing the program to crash. For the model with Smish, we had to set video memory to 2300MB. We also found that, our proposed model

with Smish performed better on Edema and Enhancing-Tumor classes in terms of IoU and F1 scores compared to our proposed model with ReLU. Figure 4.5 shows the IoU score, Figure 4.6 shows the F1 Score of our proposed model with ReLU during training and Figure 4.7 shows the IoU score and Figure 4.8 shows the F1 score of our proposed model with Smish activation during training. Table 4.2 and 4.3 demonstrates the results of the models on the testing set for Non-enhancing tumor/Necrosis, Edema and Enhancing Tumor on testing dataset. On Figure 4.9, we showed a sample prediction in 3D where the left image is ground truth, middle is our proposed model with ReLU and right image is our proposed model with Smish.

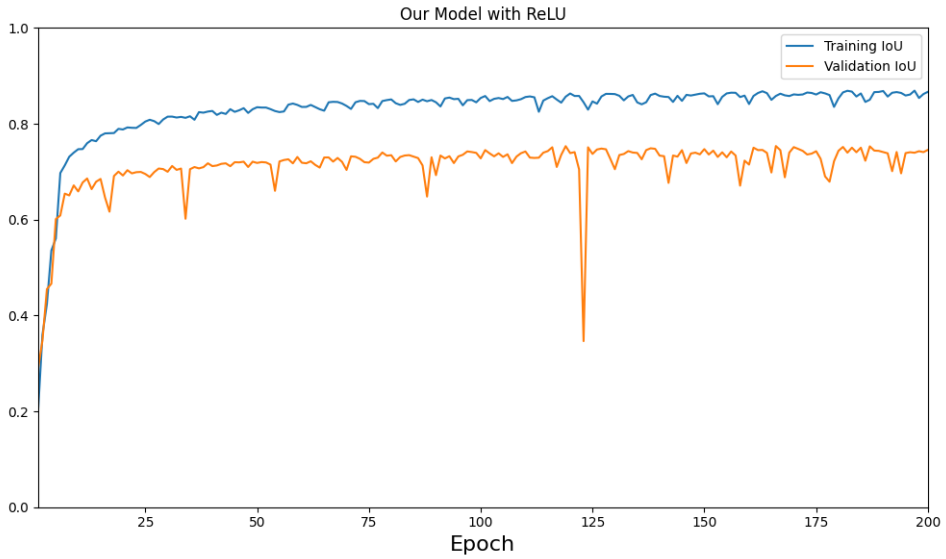


Figure 4.5: Our Proposed Model with ReLU IoU Score during training

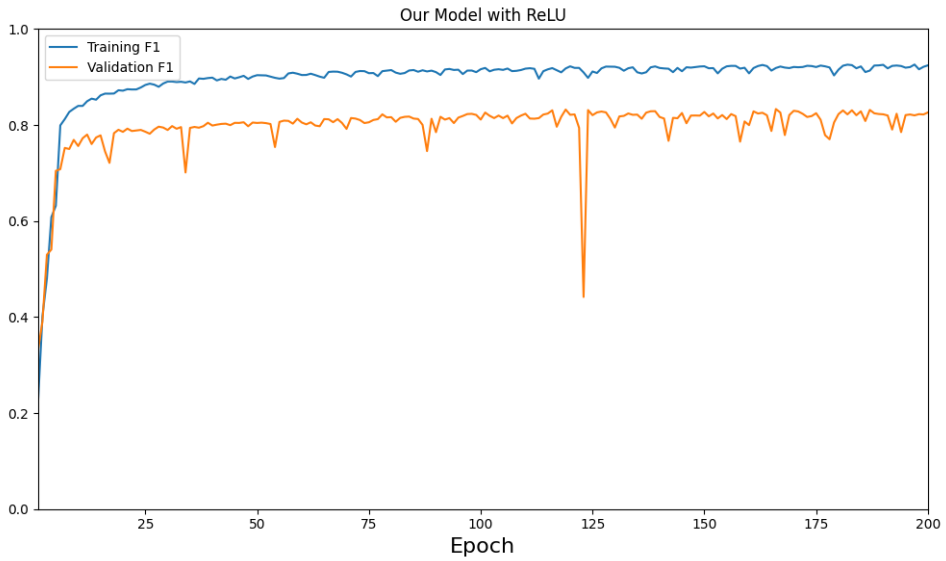


Figure 4.6: Our Proposed Model with ReLU F1 Score during training

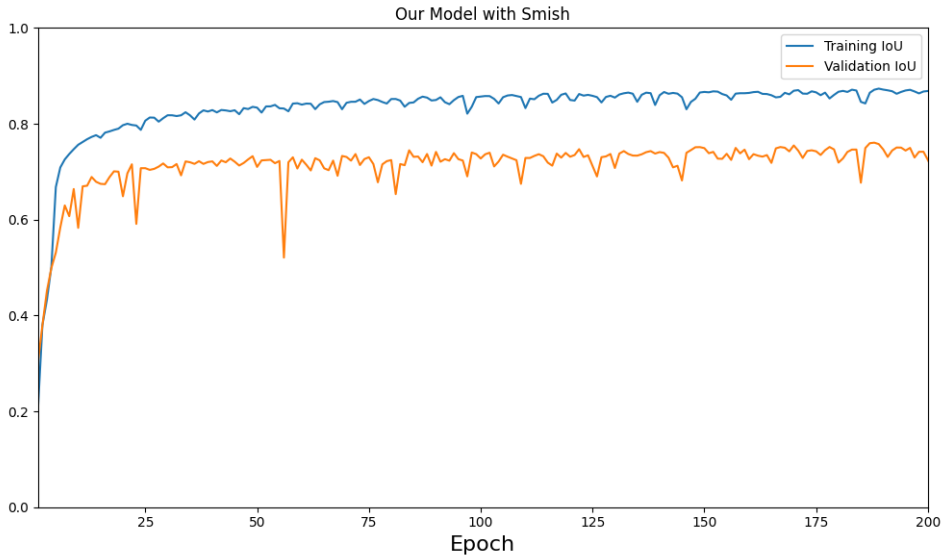


Figure 4.7: Our Proposed Model with Smish IoU Score during training

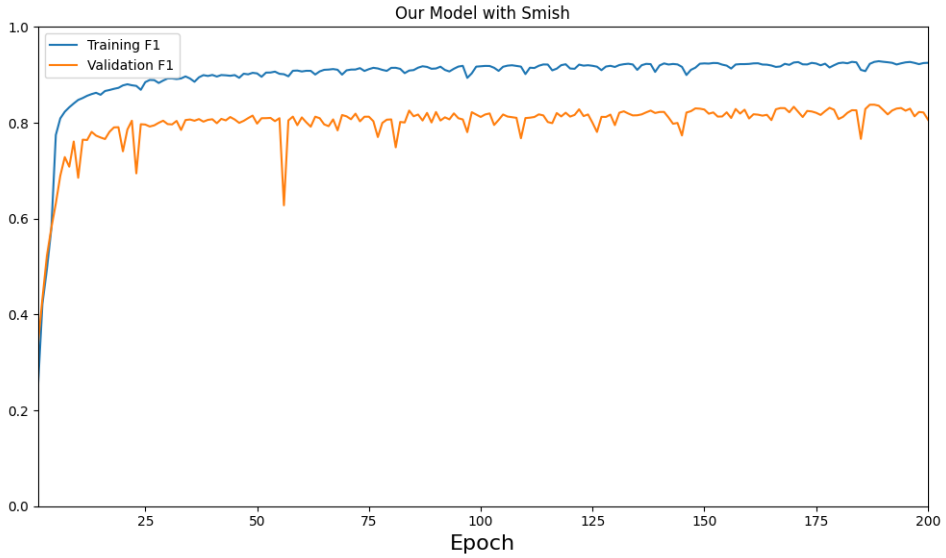


Figure 4.8: Our Proposed Model with Smish F1 Score during training

Table 4.2: Our Proposed Model with ReLU results on different classes on testing dataset

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9973	0.663	0.7462	0.99810	0.8135	0.7406
ED	0.9885	0.6833	0.7906	0.997	0.7658	0.866
ET	0.9976	0.693	0.7818	0.9983	0.8361	0.7596

Table 4.2 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for our proposed model where ReLU activation was used. The results are shown for the classes

Non-Enhancing Tumor or Necrosis (NET/NCR), Edema (ED) and Enhancing-Tumor (ET).

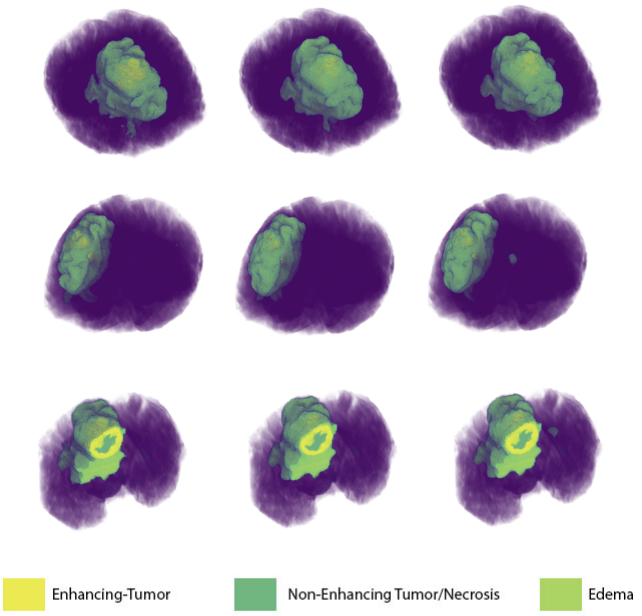


Figure 4.9: Sample prediction between Our proposed model with ReLU and Smish (Left GT, Middle w/ ReLU, Right w/ Smish)

Table 4.3: Our Proposed Model with Smish results on different classes on testing dataset

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9975	0.658	0.7427	0.9983	0.8005	0.7373
ED	0.9911	0.7007	0.8042	0.9963	0.8142	0.8283
ET	0.9976	0.7235	0.8104	0.9989	0.8151	0.8364

Table 4.3 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for our proposed model where Smish activation was used. The model performed better in terms of IoU and F1 score on the classes ET or Enhancing Tumor and ED or Edema classes compared to when ReLU activation was used. The results are shown for the classes Non-Enhancing Tumor or Necrosis (NET/NCR), Edema (ED) and Enhancing-Tumor (ET).

On the otherhand, we trained our proposed model with ReLU activation with Separable Convolutions 2 from Figure 3.3. This time the model showed better results in terms of IoU and F1 score compared to using Separable Convolutions 1 on the testing dataset. The model was trained with corresponding hyper parameters as our proposed model with ReLU and the validation loss did not improve after 189th epoch reaching validation IoU score 0.7545309663 and F1 score 0.8326327205. Figure 4.10 and Figure 4.11 shows the F1 and IoU scores of the model during training. Figure 4.12 shows a sample prediction between our proposed model with ReLU where we

compared separable convolution 1 and 2, the left image is the ground truth, the middle image is our proposed model with separable convolution 1 and the right image is our proposed model with separable convolution 2.

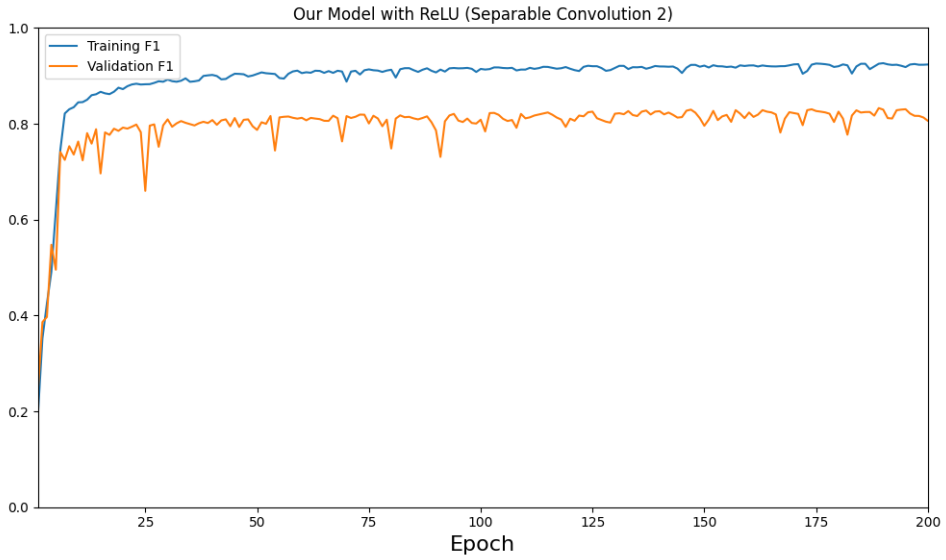


Figure 4.10: Our Proposed Model with ReLU (Separable Convolution 2) F1 Score during training

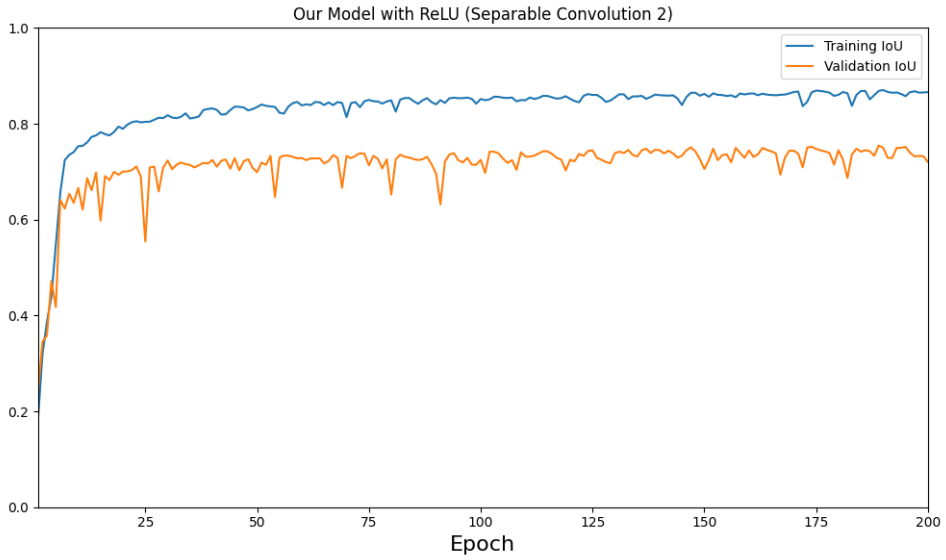


Figure 4.11: Our Proposed Model with ReLU (Separable Convolution 2) IoU Score during training

Table 4.4 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for our proposed model where separable convolution 2 was used. In this case, the model performed better on IoU and F1 score compared to when separable convolution 1 was used in Table 4.2. The results are shown for the classes Non-Enhancing Tumor or Necrosis (NET/NCR), Edema (ED) and Enhancing-Tumor (ET).

Table 4.4: Our Proposed Model with ReLU (Separable Convolution 2) results on different classes on testing dataset

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9975	0.6653	0.7563	0.9984	0.8117	0.7527
ED	0.9906	0.6929	0.7958	0.9954	0.8278	0.8039
ET	0.9947	0.7181	0.8049	0.9986	0.8296	0.8058

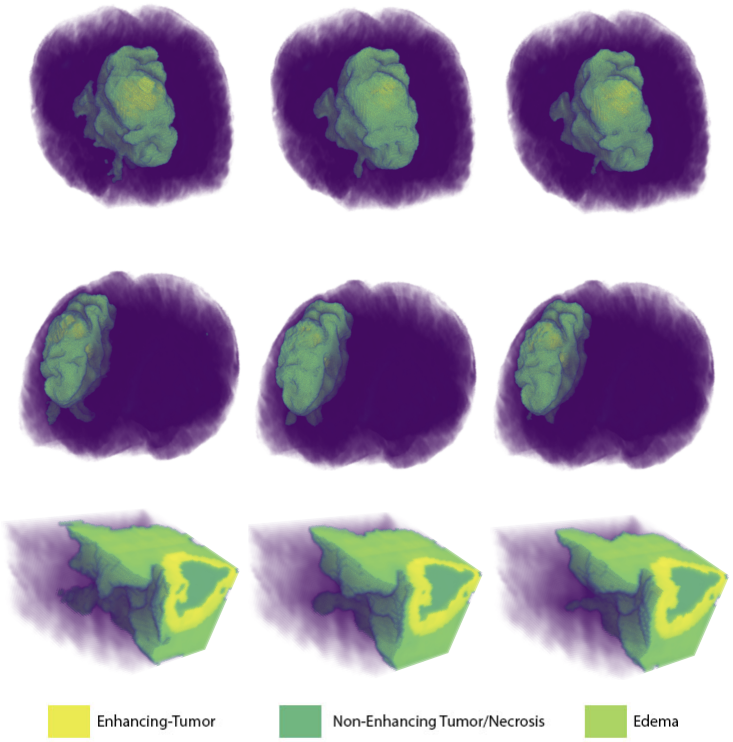


Figure 4.12: Sample prediction between Our proposed model with Separable Convolution 1 and 2 (with ReLU) (Left GT, Middle SC 1, Right SC 2)

Finally, we modified our proposed model to use the separable convolution 2 and the modified stem block based on ResNet-D. We also implemented deep-supervision to the model and trained the model on the dataset of 64x64x64 patches. We used Smish activation function in the last encoder, first decoder and bottleneck block layers and Swish activation in other blocks' layers as we found that the Smish function is more memory consuming than Swish function. After training the model with the patches, we trained the model on the dataset of 128x128x128 images using transfer learning to load the weights and for fine-tuning, we set the layers in the bottleneck layer to not-trainable. The models were trained using same hyper parameters as our previous models, however, during training the patches, the learning rate was initially set to 0.001 and batch size of 16. The model's validation scores didn't improve after 31th epoch reaching validation IoU score 0.7575016022 and validation F1 score 0.848244727 for main output layer so we stopped the training and saved the best model and again trained the model at a learning rate at 0.0001. This time the validation scores didn't improve after 32th epoch reaching validation IoU score 0.77687698602 and F1 score 0.860830307 for main output layer. Finally, we

used the best model and trained the model on the 128x128x128 images by loading the weights from training on patches, freezing the layers on bottleneck block. The learning rate was set to 0.0001 and the validation scores didn't improve after 11th epoch reaching validation IoU score 0.7801004648 and F1 score 0.851087749, higher than U-Net. Figure 4.14 and Figure 4.15 shows the IoU and F1 scores of the model during training on patches with learning rate set to 0.001, Figure 4.17 and Figure 4.16 shows the IoU and F1 scores when learning rate set to 0.0001. Finally, Figure 4.18 and Figure 4.19 shows the F1 and IoU scores of the model when the pre-trained model on patches is used to train on the 128x128x128 images. On the figures, we showed only the curves of main output as when deep-supervision is used, the model would output two results for two different outputs. Figure 4.13 shows the process of training the model using patches, transfer learning and fine-tuning.

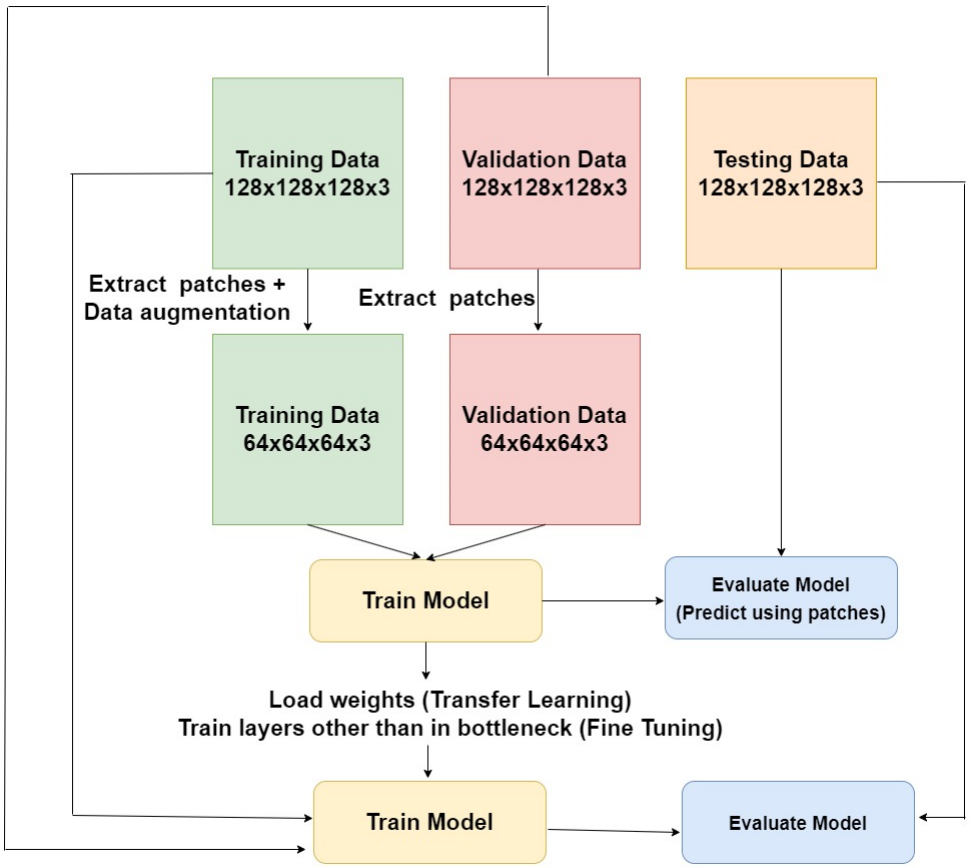


Figure 4.13: Patch-wise pre-training and transfer learning

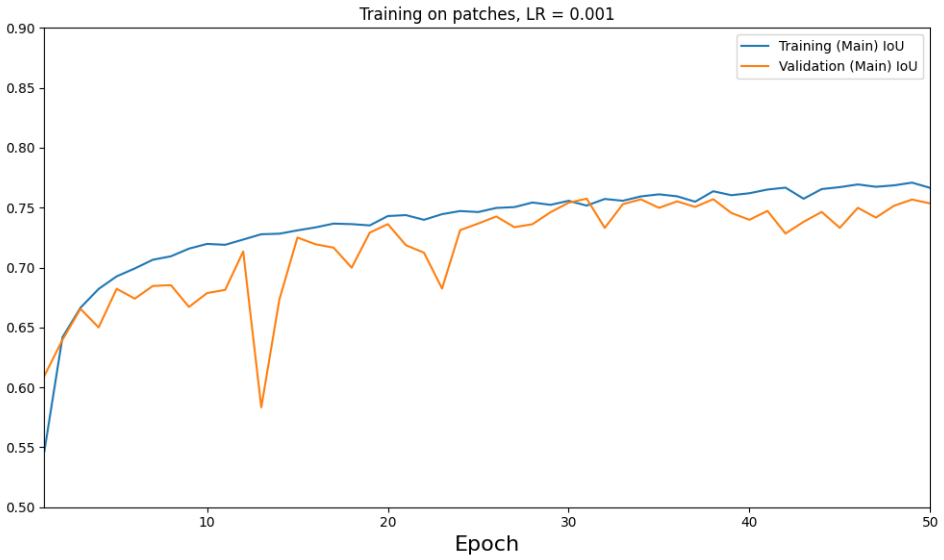


Figure 4.14: IoU Score during Training on patches (LR = 0.001)

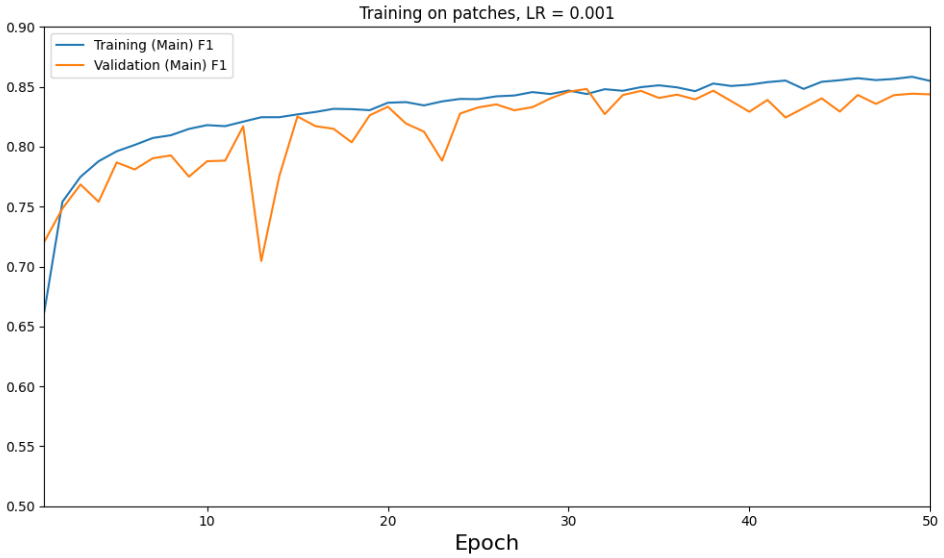


Figure 4.15: F1 Score during Training on patches (LR = 0.001)

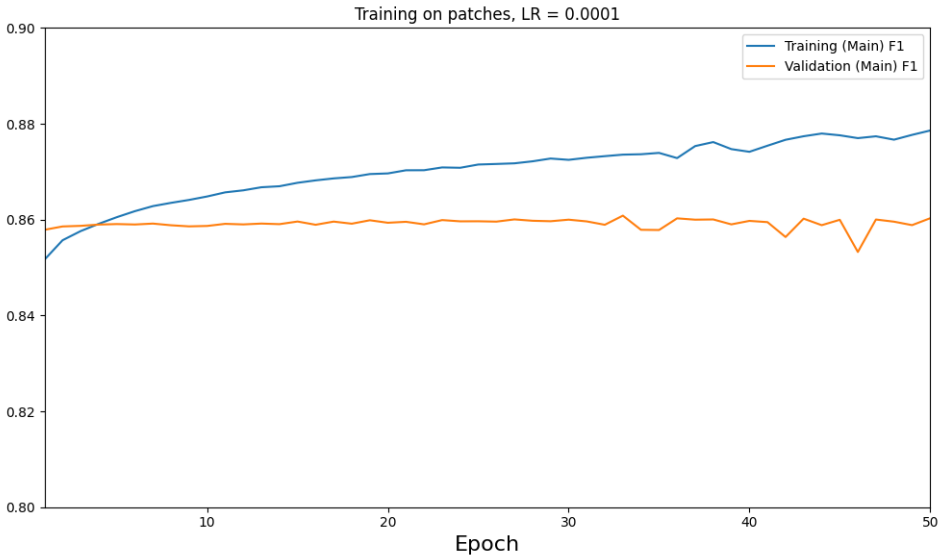


Figure 4.16: F1 Score during Training on patches (LR = 0.0001)

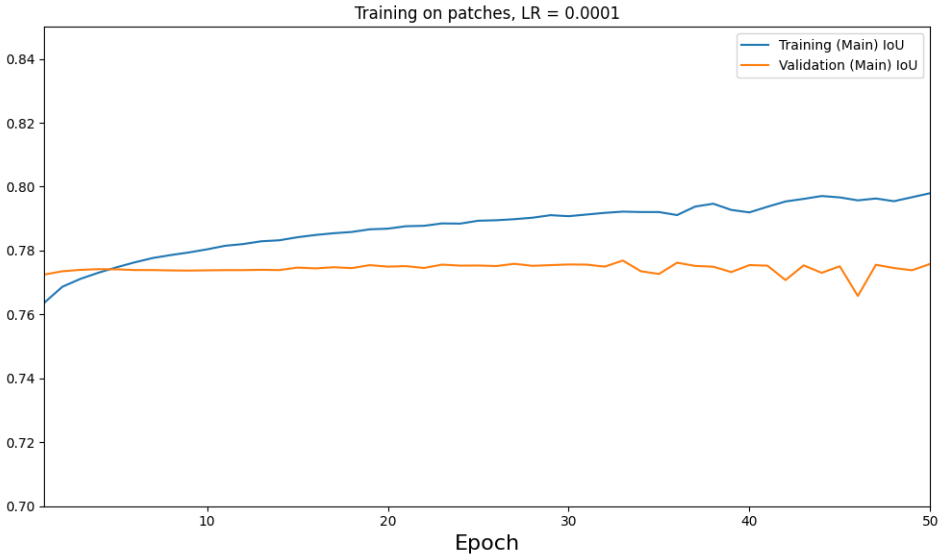


Figure 4.17: IoU Score during Training on patches (LR = 0.0001)

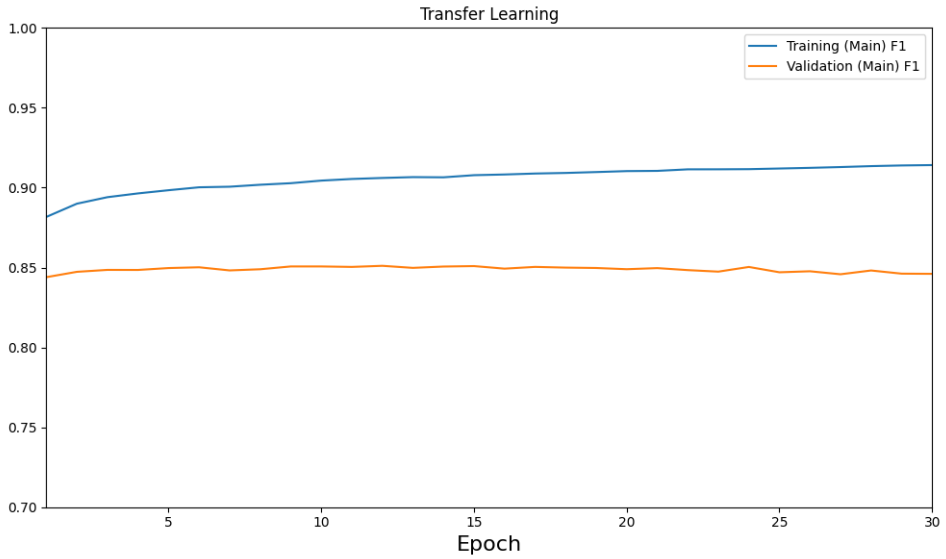


Figure 4.18: F1 Score during Training pre-trained model on 128x128x128 images

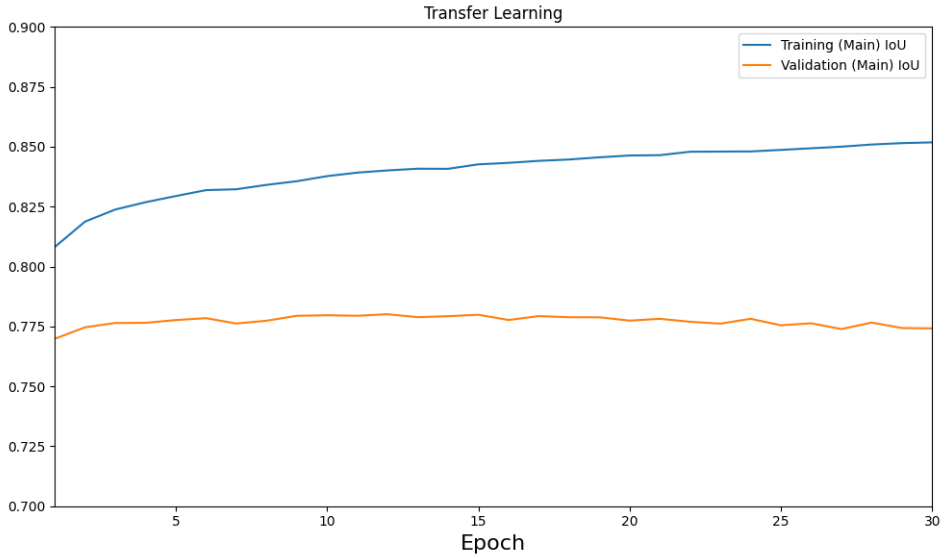


Figure 4.19: IoU Score during Training pre-trained model on 128x128x128 images

We found that, pre-training the model on the patches and again training the model on the same images that were used to extract the patches can improve the accuracy. The final model through transfer learning showed better results compared to our previous models. Table 4.5 shows the results on the testing dataset when we used only the model trained using patches, for predicting 128x128x128 images using model trained on 64x64x64 images, we had to extract patches for the images and predict on the patches individually and again combined the predicted patches back together to original shape and calculated the results. Table 4.6 shows the results of the model on testing dataset where we used patch-wise pre-training and transfer-learning. Figure 4.20 shows sample prediction where left image is ground truth, middle image prediction is where we used the model trained using patches to predict by creating patches from the image and the right image prediction is where we used transfer-learning from the patch-based trained model.

Table 4.5: Results on testing dataset by predicting using patches

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9972	0.6468	0.73445	0.9983	0.788	0.7413
ED	0.9897	0.674	0.7810	0.9958	0.783	0.8197
ET	0.9954	0.6574	0.7522	0.9986	0.759	0.7889

Table 4.5 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for our proposed model where it was trained only using patches of 64x64x64 dimensions and tested on 128x128x128 images through extracting patches of 64x64x64 dimensions from the testing image and un-patching them back to 128x128x128 dimension to calculate the results. The results are shown for the classes Non-Enhancing Tumor or Necrosis (NET/NCR), Edema (ED) and Enhancing-Tumor (ET).

Table 4.6: Results on testing dataset through patch-wise pre-training and transfer-learning

Class	Accuracy	IoU	F1	Specificity	Sensitivity	Precision
NET/NCR	0.9976	0.6915	0.772	0.9985	0.8429	0.7507
ED	0.9924	0.725	0.819	0.9959	0.8426	0.8256
ET	0.9981	0.7444	0.8254	0.9989	0.8362	0.8281

Table 4.6 shows the mean values of Accuracy, Intersection over Union (IoU), Dice Score (F1), Specificity, Sensitivity and Precision on the testing dataset for our proposed model where the weights from the model trained on 64x64x64 were loaded via transfer learning and again trained on the 128x128x128 images. In this case, it showed better results in all cases than the model that was only trained on patches on Table 4.5.

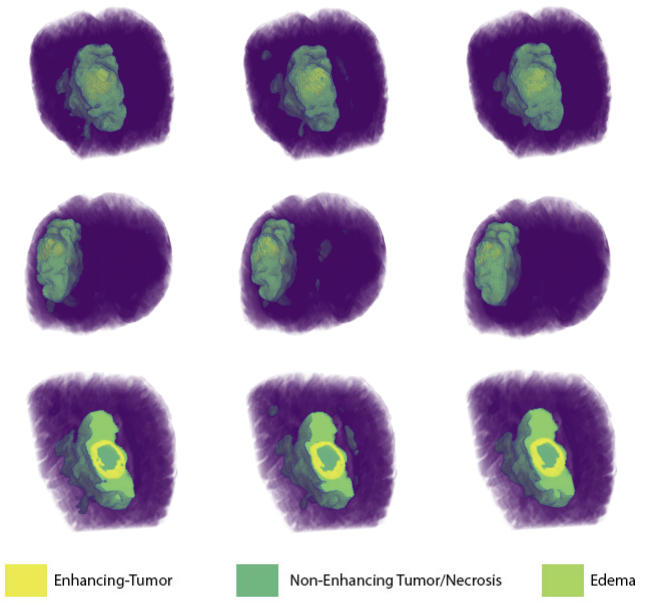


Figure 4.20: Sample prediction between predicting using patches (Middle) and our final model with patch-wise pre-training and transfer-learning model (Right) (Left GT)

4.6 Comparison of Proposed Model with U-Net

We compared our final model based on patch-wise pre-training and transfer learning with U-Net as it performed the best compared to our previous experiments on the testing dataset. Furthermore, we also compared the results for Whole Tumor (WT) and Tumor Core (TC). The Whole Tumor is the combination of Edema or ED, Enhancing Tumor or ET, Non-Enhancing Tumor/Necrosis or NET/NCR and Tumor Core or TC is the combination of Non-Enhancing Tumor/Necrosis or NET/NCR and Enhancing-Tumor or ET.

$$WT = ET + ED + NET|NCR \tag{4.10}$$

$$TC = ET + NET|NCR \tag{4.11}$$

In order to get the results for WT or Whole Tumor and TC or Tumor Core, first we added the predicted classes based on the Equation 4.10 and 4.11 and set the value of prediction between 0 or 1 keeping a threshold of 0.5. If the predicted value is less than or equal to 0.5 then we set the value to 0 otherwise we set it to 1. We compared the accuracy on Table 4.7, IoU scores on Table 4.8, F1 Scores on 4.9, Precision on Table 4.10, Sensitivity or Recall on Table 4.11, and Specificity on Table 4.12 on testing dataset. Again, we showed the box-plots of F1 (Dice) Score on the Figure 4.21 for Edema (ED), Enhancing Tumor (ET) and Non-Enhancing Tumor/Necrosis (NET) and box-plots of F1 (Dice Score) on Figure 4.22 for Whole Tumor(WT), Enhancing Tumor(ET) and Tumor Core(TC). We showed sample predictions between Our Proposed Model and U-Net on Figure 4.23.

Table 4.7: Comparison of Accuracy on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.998	0.9929	0.997	0.998	0.9916
Proposed Model	0.9981	0.9935	0.997	0.9976	0.992

Table 4.7 compares the mean accuarcy on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED.

Table 4.8: Comparison of IoU on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.7484	0.8384	0.8282	0.6869	0.7141
Proposed Model	0.7444	0.8511	0.8297	0.6915	0.725

Table 4.8 compares the mean IoU Score on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED

Table 4.9: Comparison of F1 (Dice Score) on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.8273	0.9044	0.885	0.7654	0.8142
Proposed Model	0.8254	0.9105	0.884	0.772	0.819

Table 4.9 compares the mean F1 Scores on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED

Table 4.10: Comparison of Precision on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.8333	0.897	0.88	0.76	0.8202
Proposed Model	0.8281	0.9037	0.8752	0.7507	0.8255

Table 4.9 compares the mean F1 Scores on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED

Table 4.11: Comparison of Recall/Sensitivity on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.8432	0.929	0.9202	0.8195	0.8421
Proposed Model	0.8362	0.9338	0.9148	0.8429	0.8427

Table 4.11 compares the mean Sensitivity or Recall on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED

Table 4.12: Comparison of Specificity on U-Net and Our Proposed Model

Model	ET	WT	TC	NET/NCR	ED
U-Net	0.9989	0.996	0.9982	0.9986	0.9958
Proposed Model	0.9989	0.9963	0.9982	0.9984	0.9959

Table 4.12 compares the mean Specificity on the testing set between U-Net and Our final model on the classes ET, WT, TC, NET/NCR and ED

Table 4.13: Comparison of U-Net and Our Proposed Model

Model	Parameters	Size	Traning Time	Prediction Time
U-Net	5,651,716	65.0 MB	15+ min	0.355ms
Proposed Model	604,495	2.87 MB	5+ min	0.194ms

Table 4.13 compares the number of parameters required, Size, Training time per epoch on 128x128x128 images and prediction time per image on the testing dataset between Our final model and U-Net
Based on the final results, our proposed model has achieved a very comparable performance compared to U-Net and performed better in some cases while maintaining around 11% of total parameters, reduced computational cost and less inference time. During prediction, we removed the hidden output layer that was used for deep-supervision as the main output layer produces the best result. Removing the layer reduced the model size from 6.62 MB to 2.87 MB, reduced parameters from 627,155 to 604,495 and decreased inference time. The model took around 12 minutes per epoch when training on the patches of the images, the Table 4.13 shows the parameters, size on disk, training time per epoch on training dataset of 128x128x128 images and prediction time per image on testing dataset.

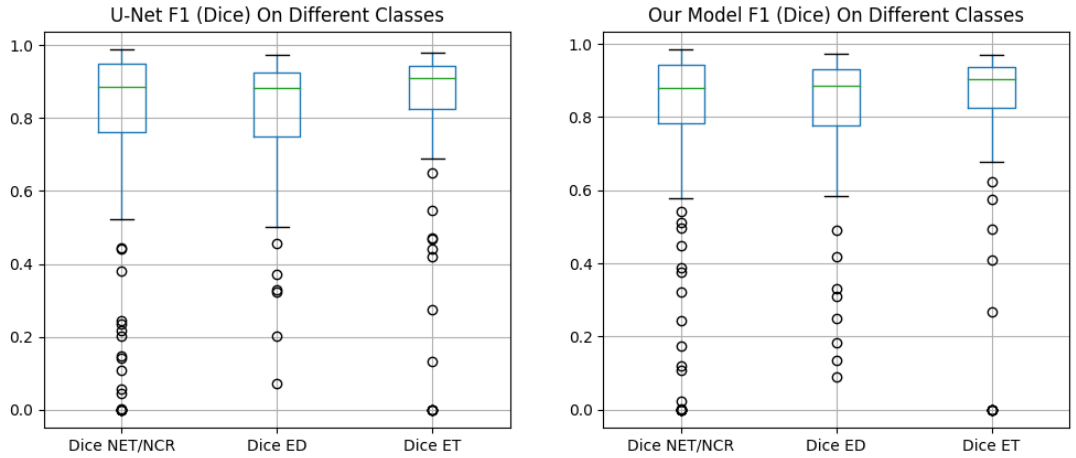


Figure 4.21: Box-plot of F1 (Dice) Scores on testing dataset between Our Proposed Model and U-Net for NET/NCR, ED and ET

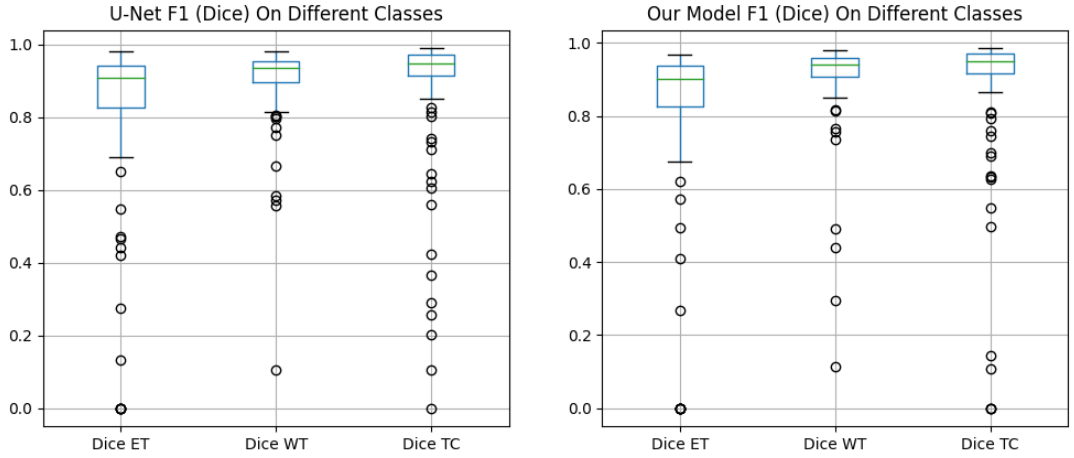


Figure 4.22: Box-plot of F1 (Dice) Scores on testing dataset between Our Proposed Model and U-Net for ET, WT and TC

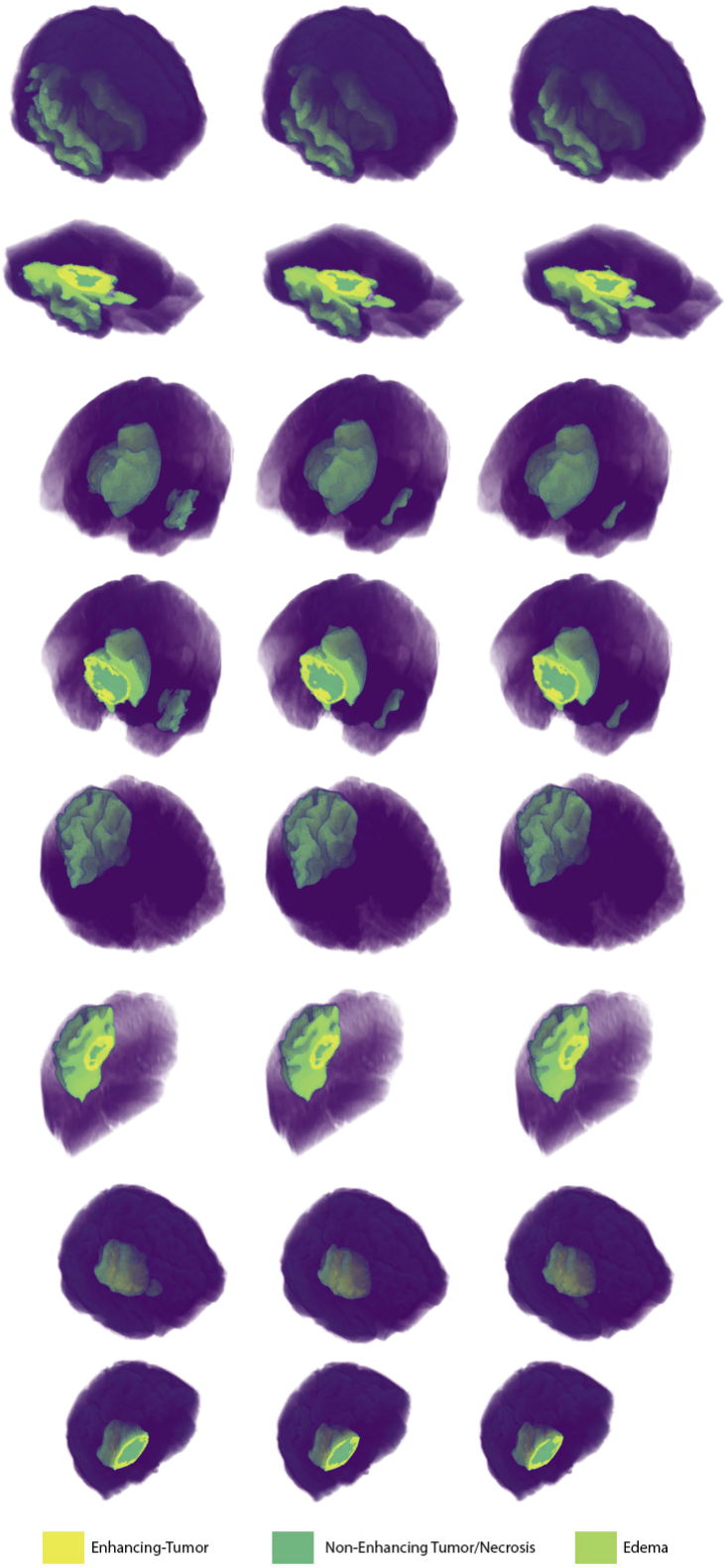


Figure 4.23: Sample predictions between U-Net (Middle) and Our Proposed Model (Right) (Left GT)

Chapter 5

Conclusion & Future Work

5.1 Conclusion

Our final model with around 11% parameters, around one-third of training time and around 45% faster inference time compared to U-Net managed to provide better results in various cases and provided very comparable performance where it didn't manage to achieve higher scores. The model is able to greatly reduce computational cost thus making it more efficient.

5.2 Future Work

In the future, we hope to implement 3D Depth-wise convolution to our model after Tensorflow releases it. We also want to evaluate the model's performation through BraTS online validation.

Bibliography

[1] S. T. Chao, J. H. Suh, S. Raja, S.-Y. Lee, and G. Barnett, *The sensitivity and specificity of fdg pet in distinguishing recurrent brain tumor from radionecrosis in patients treated with stereotactic radiosurgery*, en, 2001. DOI: 10.1002/ijc.1016. [Online]. Available: http://dx.doi.org/10.1002/ijc.1016.

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. xx, no. x, pp. xx–xx, 2008.

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, 2008, pp. 44–57.

[4] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 regularization for learning kernels," *ArXiv*, vol. abs/1205.2653, 2009.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[6] W. Wang, Y. Hu, P. Lu, *et al.*, *Evaluation of the diagnostic performance of magnetic resonance spectroscopy in brain tumors: A systematic review and meta-analysis*, en, D. Monleon, Ed., Nov. 2014. DOI: 10.1371/journal.pone.0112577. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0112577.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[8] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *ArXiv*, vol. abs/1409.5185, 2015.

[9] B. H. Menze, A. Jakab, S. Bauer, *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," en, *IEEE Trans. Med. Imaging*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015.

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

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[12] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size," *ArXiv*, vol. abs/1602.07360, 2016.

[13] S. Bakas, H. Akbari, A. Sotiras, *et al.*, "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," en, *Sci. Data*, vol. 4, p. 170 117, Sep. 2017.

[14] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[15] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," *arXiv: Neural and Evolutionary Computing*, 2017.

[16] C. H. Sudre, W. Li, T. K. M. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," *Deep learning in medical image analysis and multimodal learning for clinical decision support : Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, held in conjunction with MICCAI 2017 Quebec City, QC,...*, vol. 2017, pp. 240–248, 2017.

[17] A. F. Agarap, "Deep learning using rectified linear units (relu)," *ArXiv*, vol. abs/1803.08375, 2018.

[18] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[19] F. Isensee, J. Petersen, A. Klein, *et al.*, "Nnu-net: Self-adapting framework for u-net-based medical image segmentation," *ArXiv*, vol. abs/1809.10486, 2018.

[20] O. Oktay, J. Schlemper, L. L. Folgoc, *et al.*, "Attention u-net: Learning where to look for the pancreas," *ArXiv*, vol. abs/1804.03999, 2018.

[21] S. Xie, C. Sun, J. Huang, Z. Tu, and K. P. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *ECCV*, 2018.

[22] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S...*, vol. 11045, pp. 3–11, 2018.

[23] W. Chen, B. Liu, S. Peng, J. Sun, and X. Qiao, "S3d-unet: Separable 3d u-net for brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, Eds., Cham: Springer International Publishing, 2019, pp. 358–368, ISBN: 978-3-030-11726-9.

[24] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 558–567, 2019.

[25] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *ArXiv*, vol. abs/1908.08681, 2019.

[26] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," *ArXiv*, vol. abs/1907.09595, 2019.

[27] M. Amiri, R. Brooks, and H. Rivaz, "Fine-tuning u-net for ultrasound image segmentation: Different layers, different outcomes," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 67, no. 12, pp. 2510–2518, 2020. DOI: 10.1109/TUFFC.2020.3015081.

[28] N. Beheshti and L. Johnsson, "Squeeze u-net: A memory and energy efficient image segmentation network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1495–1504. DOI: 10.1109/CVPRW50498.2020.00190.

[29] H. Huang, L. Lin, R. Tong, *et al.*, "Unet 3+: A full-scale connected unet for medical image segmentation," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1055–1059, 2020.

[30] U. Baid, S. Ghodasara, M. Bilello, *et al.*, "The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification," *ArXiv*, vol. abs/2107.02314, 2021.

[31] K. D. Miller, Q. T. Ostrom, C. Kruchko, *et al.*, "Brain and other central nervous system tumor statistics, 2021," *CA: A Cancer Journal for Clinicians*, vol. 71, no. 5, pp. 381–406, 2021. DOI: https://doi.org/10.3322/caac.21693. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21693. [Online]. Available: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21693.

[32] H. Zhu, H. Zeng, J. Liu, and X. Zhang, "Logish: A new nonlinear nonmonotonic activation function for convolutional neural network," *Neurocomputing*, vol. 458, pp. 490–499, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.06.067. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221009917.

[33] R. Solovyev, A. A. Kalinin, and T. Gabruseva, "3d convolutional neural networks for stalled brain capillary detection," *Computers in Biology and Medicine*, vol. 141, p. 105 089, 2022. DOI: 10.1016/j.compbiomed.2021.105089.

[34] X. Wang, H. Ren, and A. Wang, "Smish: A novel activation function for deep learning methods," *Electronics*, vol. 11, no. 4, 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11040540. [Online]. Available: https://www.mdpi.com/2079-9292/11/4/540.