The University of Nottingham

Faculty of Engineering

Department of Electrical and Electronic Engineering

Medical Image Segmentation on Edge Devices

AUTHOR	:	Tham Shung Yan
ID	:	20114707
SUPERVISOR	:	Dr. Hermawan Nugroho
MODERATOR	:	Prof. Nandha Kumar
DATE	:	14/4/2023

Fourth year project thesis is submitted in partial fulfilment of the requirements of the degree of **Master of Engineering**

Abstract

Medical semantic segmentation is crucial in medical image analysis, enabling doctors to identify different anatomical structures or abnormalities. There has been a growing interest in medical semantic segmentation on edge devices in recent years due to their real-time analysis, security, and privacy advantages. In this thesis, I focus on optimizing the performance of the U-Net architecture for medical semantic segmentation and applying it to edge devices. Specifically, I investigate the impact of different types of pooling layers on the performance of U-Net and compare their results. I also explore using both 2D and 3D U-Net for medical segmentation using Brain Tumor Segmentation Dataset (BRATS) and Chest X-ray datasets to evaluate their performance.

Furthermore, I implemented a 2D U-Net on a Google Coral Dev Board Mini, demonstrating the feasibility of performing medical semantic segmentation on edge devices with limited computational resources. My experimental results show that the choice of pooling layer significantly impacts the performance of U-Net, with different pooling layers showing better results on different datasets. Moreover, I prove that 2D and 3D U-Net can achieve competitive performance for medical segmentation tasks compared to other state-of-the-art models like DeepLabv3+. My work provides insights into optimizing U-Net and demonstrates the feasibility of performing medical semantic segmentation on edge devices.

Table of Content

List of Figures

List of Tables

1.0 Introduction

1.1 Background of Project

Image segmentation is a technique for breaking up a digital image into smaller groupings called image segments, which reduces the complexity of the image and makes each segment more easily processed or analysed. In short, segmentation is giving labels to pixels in a picture to distinguish between objects, persons, or other significant aspects.

Object recognition is a widespread use of image segmentation. It is standard protocol to initially apply an image segmentation method to discover an object of interest in the picture before processing the complete image. The object detector may then work with the segmented objects. By stopping the detector from analysing the whole picture, accuracy is increased, and the amount of time for inference is decreased because noises or redundant components are removed.

A crucial component of computer vision technologies and algorithms is image segmentation. It is employed in various real-world contexts, including face recognition, video surveillance, medical image analysis, autonomous cars, and satellite image analysis.

Segmentation is crucial for computer-aided diagnosis (CAD) systems in the medical field to automatically extract the region of interest (ROI) for analysis. It separates an image into sections according to a given description, for example, segmenting lung tissues using lung X-rays or brain tissues using brain MRI.

Numerous image segmentation techniques are introduced, such as thresholding, k-means clustering, region growing, and sparsity-based methods. However, deep learning image segmentation has recently drawn much interest because of how well they handle tasks like segmenting medical images. The encoder-decoder neural networks like U-Net demonstrated good outcomes on medical segmentation compared to other deep learning techniques. After U-net was proposed, the research community suggested several U-Net variations like Residual U-Net, U-Net++, and R2U++. [1]

1.2 Problem statement

1. Overburdened radiologists

Accurate and timely medical image diagnosis results might mean the difference between life and death. Nonetheless, the number of radiologists in practice in the US is insufficient to fulfil the rising demand. The Association of American Medical Colleges (AAMC) predicts that by 2033, the United States will lack 17,000 to 42,000 radiologists, pathologists, and psychiatrists. This shortage is also expected to get worse. An average of up to a five percent increment on medical imaging investigations is performed annually, while only a two percent increment on radiologists. It means that existing radiologist are overburdened.

Increasing radiologist efficiency and reducing radiologist's burden is another approach to this staffing problem. Some research has investigated possible ways radiologists may become more productive, including utilizing AI in practice. AI can be used to assist in measuring lesions and analyse medical images that are more likely to be urgent. Yet, it is unlikely to replace the role of the radiologist completely. According to a study, AI may be used to prioritise radiologists' interpretation of the chest X-rays that were most likely to be abnormal. With this technique, the time it takes to interpret abnormal scans was reduced from 11.2 days to 2.7 days on average. [2]

2. Computational Costly

Most deep-learning techniques require substantial hardware, memory, and computational power—these algorithms often have more than 10 million parameters. For instance, Residual-Net18 has roughly 11 million trainable parameters, Residual-Net34 has 63.5 million, and U-Net++ has 9.04 million trainable parameters. Expensive computer hardware is needed for these models, like graphics processing units (GPUs). [1]

As a result, these algorithms are computationally costly, making it difficult to prototype them in portable devices. Hence, we must execute these models on low-power edge devices to make medical image segmentation models portable and appropriate for prototyping. The medical industry could transform significantly if edge devices segment medical images for diagnosis. Quicker diagnosis, remote access to AI-based analysis, and mobility of diagnosis tools are all enticing features. Additionally, their low energy consumption and small size will aid the gadgets' mobility.

1.3 Aim, Objectives and Deliverables

1.3.1 Aim

This project's main aim is to investigate how to optimize the model's performance without increasing the number of parameters. By achieving this main aim, a small-sized model on an edge device with optimized performance can reduce the burden of radiologist and resolve the high computational cost problem as mentioned in the problem statement. There are two sub-aims to help achieve the main aim.

(1) To compare the semantic segmentation performance of 3D medical images (MRI) and 2D medical images (X-Ray) and determine which approach (e.g., pooling layers) yields better performance.
(2) To reduce the number of parameters in the model so that it can be implemented on edge devices and resolve the high computational cost issue.

1.3.2 Objectives

1.3.2.1 Prework and Literature Review:

(1) To research the types of models, datasets, and edge devices used in medical semantic segmentation. (Aim 1 and 2)

1.3.2.2 Experiment and Development:

(2) To pre-process and augment 3D and 2D medical datasets (BRATs and Chest X-Ray datasets). (Aim 1)
(3) To modify 3D and 2D U-Net to reduce the number of parameters. (Aim 2)
(4) To train and test 3D and 2D U-net with 3D and 2D medical datasets. (Aim 1)
(5) To investigate the performance of different pooling layers. (Aim 1)
(6) To compare the performance of U-Net with other models. (Aim 1)
(7) To implement and test the trained model on an edge device (Aim 2)

1.3.3 Deliverables:

(1) A list of models, datasets, and edge devices for medical segmentation (Objective 1)
(2) A pre-processed and augmented medical 3D and 2D dataset (Objective 2)
(3) A modified 3D and 2D U-Net with reduced parameters (Objective 3)
(4) Trained U-Net model with a decent result (Objective 4)
(5) A comparison results between different pooling layers (Objective 5)
(6) A comparison results between U-Net and other models (Objective 6)
(7) Edge Devices which can perform medical semantic segmentation with decent results and performance that are comparable to other edge device models available in the research community (Objective 7)

1.4 Scope and Limitations

This project's scope is to investigate and compare the effect of different pooling layers on U-Net and choose the best for training. This project does not compare the effect of other parameters like batch size, optimizer, and regularization methods.

The limitation of this project is the limited training time. Two types of U-Net are trained for two datasets: the Brain Tumor Segmentation dataset (BRATS) and the Chest X-Ray dataset. The BRATS dataset is a 3D dataset, and it is trained with a 3D U-Net. The Chest X-Ray dataset is a 2D tumor dataset, and it is trained with 2D U-Net. For 3D U-Net, only two types of pooling layers are tested due to the need for more training time. The 3D U-Net requires around 2 hours for 10 epochs of training. Typically, the 3D U-Net requires 50 to 80 epochs to converge. Besides, the 3D U-Net was trained on Google Colab, which has a 2-hour time restriction per day for using their GPU. For 2D U-Net, four types of pooling layers are trained because it is easier to train and converge, requiring less training time.

Another limitation of this project is the 3D U-Net is not deployed on the edge device. The edge device I used is Google Coral Dev Board Mini. In my 3D U-Net, the Conv3D function is the convolution function for 3D inputs. However, the Conv3D function is not supported on Google Coral Dev Board Mini.

1.5 Project Timeline

At milestone 1, Objective 1 was achieved to understand the models, datasets, and edge devices used for medical semantic segmentation. Milestone 1 ranges from the 5th of September to the 2nd of October.

At milestone 2, the BRATS dataset was pre-processed and augmented dataset and trained with 3D U-net. Milestone 2 ranges from the 3rd of October to the 31st of October.

At milestone 3, the Chest X-Ray dataset was pre-processed, augmented, and trained with 2D U-net. Milestone 3 ranges from the 1st of November to the 20th of December. At milestone 3, Objective 2, 3, and 4 was achieved.

At milestone 4, The trained 2D U-Net model was converted and implemented on Google Coral Dev Board Mini. Thesis writing will be focused from the 15th of February to the 20th of March so that the draft thesis can be submitted before the 20th of March. The methodology was written from the 15th of February to the 28th of February. Results and Discussion was written from the 1st of March to the 10th of March. Comparison between different pooling layers and the performance of different models were compared when writing this section. Lastly, Literature Review, Introduction, and Conclusion are written from the 10th of March to the 20th of March. At milestone 4, Objective 5, 6, and 7 was achieved.

For milestone 5, the Draft thesis will be reviewed and improved before submission on the 17th of April. Then, a presentation for the moderator and the assessors will be prepared from the 14th of April to the 28th of April. The Gantt Chart is included in the appendix.

1.6 Industrial Relevance

Addenbrooke's Hospital in Cambridge, UK, is the first hospital in the world to deploy a deep-learning segmentation tool, InnerEye, that expedites the diagnosis of cancer patients. InnerEye precisely identify the tumor's position based on CT or MRI scans which speeds up the diagnosis time by 90%. [2] Typically, segmentation requires a radiologist to do it manually. However, InnerEye analyses scan 13 times quicker, increasing the precision because experienced radiologists can recheck the image segmented by InnerEye. Radiologists from Addenbrooke's hospital, Dr. Rajesh and Dr. Yvonne said that InnerEye saves time for busy radiologists and lets the patient start therapy quickly, thus increasing the chance of survival for the patient. They also praised the accurate results produced by InnerEye and said that they don't have to modify the results most of the time after checking them.

In Boston, Massachusetts, a startup called PathAI developed segmentation techniques to lower diagnostic inaccuracy and create strategies for personalized medical care. PathAI's model has trained over 29000 segmented images with different cancer types and can recognise nuclei in different tissue types, which is essential to determine the best course of therapy for the patient. PathAI also collaborated with organizations like Bill & Melinda Foundation and pharmaceutical companies like Bristol-Myers Squibb to bring its model to more healthcare sectors. [3]

In May 2020, Intel announced they would collaborate with the University of Pennsylvania and 29 other healthcare organisations to develop a model that recognises brain tumors using federated learning. Federated learning means the models will be passed around 29 healthcare organisations to train with their patient's dataset without sharing with other organisations. Therefore, it allows organisations to work together without sharing patients' data. Around 80000 people were diagnosed with brain tumors based on American Brain Tumour Association in 2020. With this federated learning, a sophisticated model can be trained for early detection, and patient's privacy can be protected at the same time.[4]

1.7 Thesis Outline

Section 1 is the introduction to the whole thesis. First, this section starts with a background review of medical semantic segmentation. This section also discusses the problem statement of medical semantic segmentation, followed by the aim, objectives, and deliverables to resolve the problem statement. This section also discusses the scope and limitations of this project to provide readers with a clear understanding of the parameters of this project. Next, the project timeline is mentioned to let readers know the project's milestones. Lastly, the industrial relevance was discussed to inform readers about the existing medical semantic segmentation application.

In section 2, a literature review of medical semantic segmentation is performed. This section provides a comprehensive study on image segmentation, including the types of image segmentation and the traditional methods for image segmentation. This section also discusses medical imaging and semantic segmentation, including its application, importance, and different types of medical datasets. This section also discusses and compares different deep-learning medical semantic segmentation models and explains why U-Net is the most suitable model for medical semantic segmentation. This section also discusses the importance of edge devices for medical semantic segmentation. Lastly, this section discusses the basics of U-Net, including architecture, components, and metrics.

In section 3, the methodology of this project is discussed. First, the overview of the modifications made for 3D and 2D U-Net was discussed, including several parameters, types of pooling layers, and many more. Then, the pre-processing and augmentation for the 3D Brain Tumor Segmentation (BRATS) and 2D Chest X-Ray datasets are discussed. Next, training and evaluation steps are discussed for 3D and 2D U-Net. Then, the Google Coral Dev Board Mini inference steps for converted 2D U-Net are also discussed. Lastly, the pre-trained DeepLabv3+ model from Matlab was also trained and evaluated with a 2D Chest X-Ray Dataset.

In section 4, the result outcomes are discussed. First, the training scores for 3D U-Net and 2D U-Net are observed and discussed until convergence. Then, the evaluated scores for both 3D and 2D U-Net are compared with similar models—lastly, the ways to improve both the 3D and 2D U-Net based on their weaknesses are discussed.

In section 5, the conclusion is discussed. First, the achievements and deliverables for this project are discussed. Then, the reflection on this project's weaknesses and limitations are discussed. Finally, the future works for this project are discussed.

2.0 Literature Review

2.1 A comprehensive review on Image Segmentation

Image segmentation is a process that generates an output from input images. In the output, each pixel's object class is specified by a mask. Several critical heuristics or high-level image attributes might be helpful for image segmentation. Standard image segmentation algorithms, which include clustering methods like histograms and edges, are built on these attributes. Colour is a typical instance of a heuristic. To ensure that the backdrop of the picture has a consistent colour, graphic designers often employ a green screen. A green screen allows for easier background detection and replacement during post-processing. Contrast is another instance of a helpful heuristic. Image segmentation algorithms can quickly tell a dark figure from a bright backdrop like the sky. Depending on sharp contrast, the algorithm can determine the edges of the pixels.

Even though old image segmentation methods based on these heuristics might be quick and easy, these methods generally need a lot of fine-tuning to accommodate particular scenarios via custom-developed heuristics. These methods usually need to be more precise to segment detailed imagery. Machine Learning and deep learning methods are used in more recent image segmentation methods to improve flexibility and precision. Model training is used in machine learning-based image segmentation techniques to enhance the program's capability to recognise significant characteristics. For image segmentation applications, deep learning methods are incredibly successful.

Different neural network architectures are effective for image segmentation. They often share similar fundamental elements: an encoder, a decoder, and skip connections. An encoder is a set of layers that uses deeper, more focused filters to capture visual characteristics. If the encoder is pre-trained with a similar job like object classification, it can use that to complete segmentation jobs. A decoder is a set of layers that progressively transforms the output of the encoder into a segmentation mask according to the pixel resolution of the input picture. Skip connections comprise multiple long-range neural network connections, enabling the model to recognise characteristics at various sizes to improve the model's accuracy.[5]

2.2.1 Two Main Approaches in Image Segmentation

1. Similarity Approach

   This approach is based on determining if two pixels in the image are similar enough to create a segment based on a threshold. The clustering technique uses this approach to do segmentation.

2. Discontinuity Approach

   This method depends on the image's pixel intensity values being discontinuous. Line, Point, and Edge Detection algorithms use this method to produce interim segmentation findings that may then be processed to produce the final segmented image.

2.2.2 Types of Traditional Image Segmentation Techniques

There are five common image segmentation techniques which include edge-based segmentation, threshold-based segmentation, region-based segmentation, cluster-based segmentation, and watershed segmentation.[5]

Figure 1: Result of Edge Based Segmentation[5]

The first technique, edge-based segmentation, is a technique that recognises the edges of multiple objects in the image. It helps to capture features of the object of interest using its edge. This technique also helps to remove unnecessary data from the image and makes the analysis easier. This technique uses contrast, colour, and saturation to find the edges.

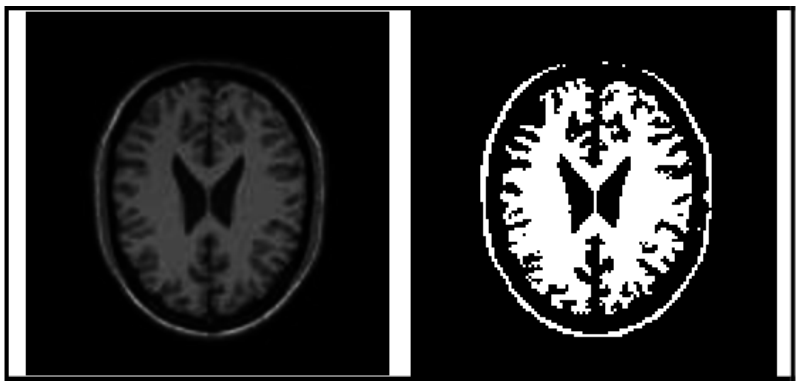


Figure 2: Results of Threshold Based Segmentation[5]

The second technique, threshold-based segmentation, is a technique that divides the pixels into the highest value of the pixel and compares it to the threshold. This technique effectively segments objects with higher pixel values than the background or other objects. The thresholding technique outputs a binary image, 1 for pixels higher than the threshold and 0 for pixels lower than the threshold.

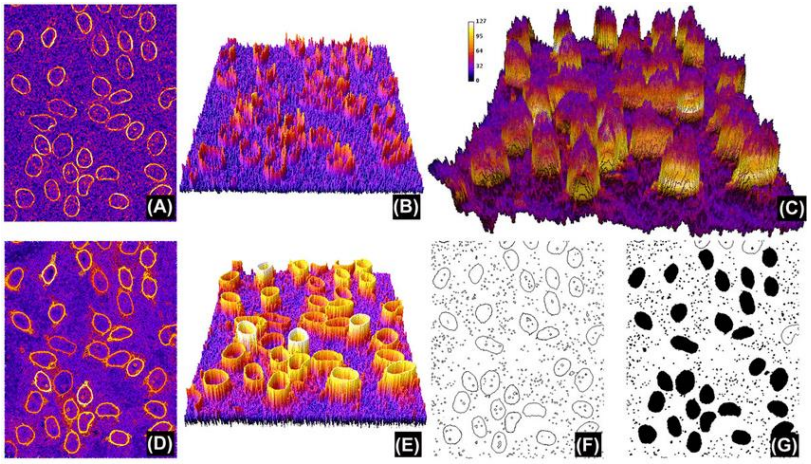


Figure 3: Region Based Segmentation[5]

The third technique, region-based segmentation, is a technique that divides images into a region with similar features. The algorithm of this technique uses seed points to find pixels that belong to them. Each seed point will form its region. The seed point can expand the region by adding more pixels into its region or merging with other seed points.

The fourth technique, cluster-based segmentation, is a technique that helps to identify hidden features in an image. It divides the images with similar properties into clusters and allows us to identify patterns that may not be visible to the human eye.

The fifth technique, watershed segmentation, is a technique that divides an image into regions based on pixel brightness. This technique form lines called ridges and basin to divide the image into regions. It can be used in medical image segmentation, like MRI scans, to differentiate different areas based on brightness. This technique helps to diagnose abnormalities in the MRI scans.

2.3 Types of image segmentation

1. Semantic Segmentation

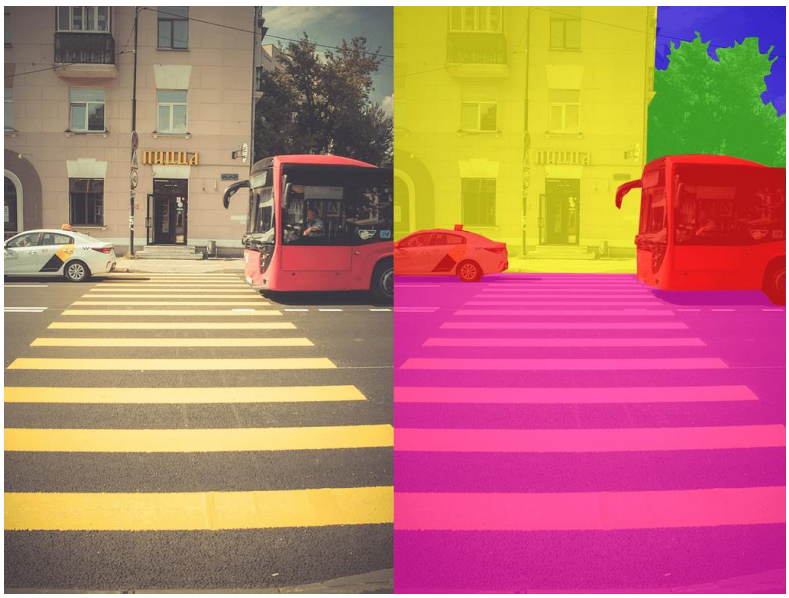


Figure 4: Semantic Segmentation[5]

In semantic segmentation, pixels of the image are arranged according to semantic classes. Each pixel in this model is a single class member, and the semantic segmentation model makes no reference to any other data and context.

For instance, executing semantic segmentation on an image containing several trees and vehicles will provide a mask that groups different kinds of trees into a single class (tree class) and all sorts of vehicles, such as cars, bicycles, or buses, into a single class (vehicles class).

When using this method is frequently vague, notably when several instances with the same class are clumped together. For instance, the "people" class may be used to categorise a group of people in a picture of a busy street. Such complicated images should be given more in-depth information by semantic segmentation.

2. Instance Segmentation

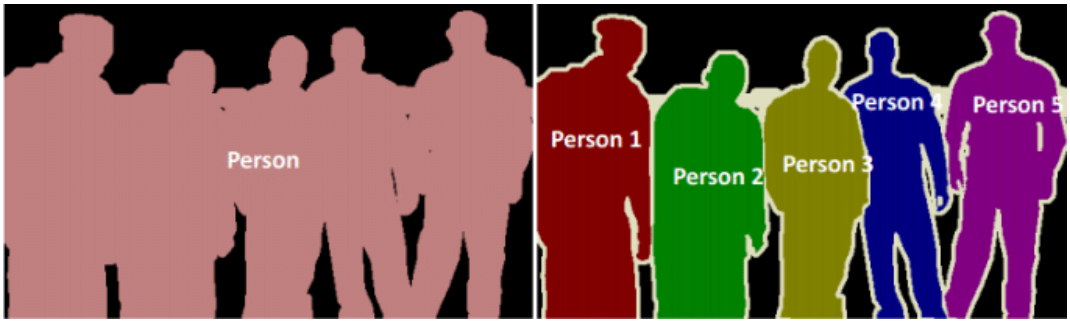


Figure 5: Instance Segmentation[5]

Instance segmentation is known as classifying pixels depending on instances of an item instead of a class of objects. Instance segmentation methods divide identical or overlapped regions based on the borders of objects. The instance segmentation model would analyse a picture of a busy street by counting the occurrences of each object in the image and finding them among the crowd.

3. Panoptic Segmentation

Figure 6: Panoptic Segmentation[5]

A more recent kind of segmentation called panoptic segmentation combines instance and semantic segmentation. It distinguishes between each occurrence of every object in the image and predicts the identity of every object.

Panoptic segmentation is helpful for many devices that need a lot of information to function. For instance, self-driving cars must be able to precisely and swiftly collect and analyse their environment. Self-driving vehicles can do this by applying panoptic segmentation to a live stream of images.

2.4 Medical Imaging and Medical Image Segmentation

Medical imaging is a technique and a procedure that involves taking pictures of the inside of a person's body for diagnostic and treatment purposes. It also shows the condition of organs or tissues visually. Medical imaging aims to identify and cure disease and disclose interior structures covered by the skin and bones. Different medical imaging techniques include X-Ray, Magnetic Resonance Imaging (MRI), Computerized Tomography (CT scan), ultrasound, Endoscopy, and Tactile Imaging. The three most prevalent techniques are X-Ray, MRI, and CT scans; they use different technology to cater to varying types of medical conditions. X-Ray usually visualizes bones and is less effective on body tissues, producing 2D images. CT scans use multiple 2D X-Ray images to form 3D CT scan images to diagnose bones, tumors, organs, and tissues. Both X-Ray and CT scans use electromagnetic radiation to produce images. MRI scans use powerful magnets and radio waves to produce 3D images, and the image they produce is more detailed than CT scans. Due to this advantage, it is used to diagnose more complex organs like the brain and spinal cord. However, MRIs consume the most time and money compared to CT scans and X-Ray. [6]

Medical imaging is essential because it helps doctors better analyse the patient's organs, tissues, bones, and other body parts in a non-intrusive manner. This step assists doctors in finding out the most effective treatment, like which medication option or whether to have surgery. Besides, it also helps to find tumors, blood clots, and bone fractures to help doctors with treatment or removal. [7]

In general, medical imaging has enhanced diagnosis and treatments by significantly lowering the 'guesswork' doctors perform, enabling them to treat patients' illnesses and injuries more precisely.

In recent years, deep learning medical image segmentation models were introduced to improve the accuracy of diagnosis further and automate the whole process. It also helps reduce the burden and save the doctor and patient's time. Medical image segmentation is a process that extracts the Region of Interest (ROI) from medical images to simplify further and analyse the patient's medical condition.

2.4.1 Applications of medical image segmentation

Singh et al. state that examining organ morphometry and accurate diagnosis may benefit substantially from segmenting bone structures. An essential first step in creating a computer-aided diagnosis of chest disorders is to categorize healthy and unhealthy chest X-ray pictures. Yet, merely classifying disorders is insufficient for evaluating chest conditions. For further study and confirmation of illness manifestation, classified unhealthy chest X-rays needed more extensive examination, such as segmentation of organs, suppression of bones, and localization of the disease. The segmentation of bones is a crucial stage in bone suppression and organ morphometry analysis. Bone segmentation is necessary for bone suppression, providing a better vision of the thoracic viscera, including the lungs, airways, heart, mediastinum, and hila. Hence, segmenting the bones in abnormal chest X-rays is crucial for enhanced, unrestricted vision to provide a precise and fast diagnosis.[8]

Agrawal T et al. highlight the significance of lung segmentation in the study of chest radiographs since it permits exact localization of lung sections within the picture and makes it

easier to detect and diagnose different lung diseases and disorders like lung cancer, tuberculosis, and pneumonia. [9]

Hou et al. emphasise the value of precise dental structure segmentation for increasing the precision and effectiveness of dental diagnostic and treatment planning. Segmented images can offer helpful information for the evaluation of missing teeth, the assessment of tooth growth, the finding of buried teeth, and the evaluation of adjacent relationships. In conclusion, the research highlights the value of precise tooth structure segmentation in panoramic X-ray pictures and suggests a brand-new deep learning-based method for accomplishing this goal. [10]

By illustrating how precise segmentation can make it possible to identify the fracture zone inside an X-ray picture, Hržić et al. underlined the significance of segmentation in fracture detection. They proposed a local entropy-based approach to segment the bone region. The authors contend that proper segmentation is particularly crucial since fractures might be subtle and hard to spot in X-ray images. For instance, children frequently suffer fractures of the ulna and radius bones. When children's bones are still developing, a radiologist might readily miss minor ulna and radius fractures. For that reason, an accurate fracture detection system would facilitate the discovery of minor fractures, lowering the possibility of a kid receiving a false-negative diagnosis. [11]

2.5 The advantage of semantic segmentation over instance segmentation in medical applications

Instance segmentation aims to differentiate between several instances of the same class, whereas semantic segmentation categorises each pixel in an image into a predefined class like tumor and lung. For several reasons, semantic segmentation is more appropriate than instance segmentation in medical applications.

The first reason is that medical applications focus on identifying and segmenting different types of organs or tumors. For example, in brain tumor MRI, the goal is to segment different types of tumor (inner tumor core, outer tumor core) and their respective region but not identify and segment each instance of tumor (tumor 1, tumor 2, tumor 3).

The second reason is due to the annotated data. All the annotated medical datasets are annotated for different classes of organs or tumors instead of each instance of the organ or tumor. Obtaining medical datasets with instance-level annotations is time and money-consuming because a professional radiologist must be hired for this job.

2.6 Types of Medical Image Segmentation

There are two types of medical image segmentation, binary and multiclass.

For binary segmentation, it means that one class is in the dataset that needs to be segmented. For example, the only class in the Chest X-Ray dataset is the lung. Once the lung region is segmented and extracted, it can be further analysed to see whether it has diseases. More precise analysis can be done because the unnecessary parts are isolated.

For multiclass segmentation, it means that multiple classes are in the dataset that needs to be segmented with respective class. For example, in the Brain Tumor Segmentation dataset (BRATS), there are multiple class types of tumor. The model will segment different types of tumors according to their respective class to let the doctor further analyse the treatment or surgery plan.

2.7 Medical Image Datasets

Table 1: Medical Image Datasets

Datasets	Medical Imaging Technique	Types of segmentation	Dimension of Image
Chest X-Ray	X-Ray	Binary	2D
Brain Tumor MRI	MRI	Multiclass	3D

Based on Table 1, two datasets that involve two types of segmentation and two types of medical imaging are used in this project. The first dataset is the Chest X-Ray dataset, which comprises Chest X-Ray images to do binary segmentation for the lung. The second dataset is the Brain Tumor Segmentation dataset (BRATS), which comprises Brain MRI Images to do multiclass segmentation for different types of tumors. The importance of using two datasets is mentioned in sections 2.7.2 and 2.7.3.

2.7.1 Types of MRI Scans in BRATS

There are four types of MRI scan for BRATS, which is native (T1) and post-contrast T1-weighted (T1CE), T2-weighted (T2), and Fluid Attenuated Inversion Recovery (FLAIR) volumes. They were obtained from several medical standards and devices.[12]

MRI uses magnets and radio frequency pulses to produce images on a computer. Different images can be produced by altering the order in which Radio Frequency pulses are delivered and captured. The interval between succeeding pulse sequences delivered to the same layer is the repetition time (TR). The period between the Radio Frequency pulse being delivered and the echo signal received is known as the time to echo (TE).

The T1 and T2-weighted scans are the most used MRI sequences. Short TE and TR periods are used to generate T1-weighted photos. T1 features of tissue are primarily responsible for determining the contrast and brightness of the T1 images. On the other hand, longer TE and TR periods are used to generate T2-weighted images. The T2 features of the tissue dominantly control the contrast and brightness in these T2 photos.

The Fluid Attenuated Inversion Recovery sequence (Flair) is a third frequently utilised sequence. The flair sequence's TR and TE timings are significantly longer than those of a T2-weighted picture. As a result, anomalies are kept bright while the normal Cerebrospinal fluid

is dimmed. This sequence greatly facilitates the distinction between Cerebrospinal fluid and an abnormality and is highly sensitive to disease.

2.7.2 Importance of lung segmentation

The classification procedure for identifying if a patient has cancer or other lung disorders can be improved by isolating the lung region from the chest. The classification algorithm can concentrate solely on the lung tissue by segmenting the lung area while ignoring other chest structures that might not be important for the classification goal.

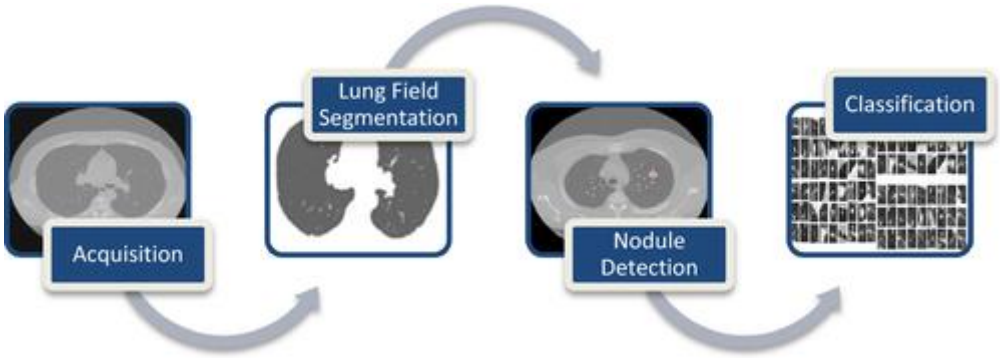


Figure 7: Process of Lung Nodule Detection [13]

Furqan Shaukat et al. begin with pre-processing, eliminating any noise that may be present in the input images, and then perform lung segmentation using thresholding. After that, multiscale dot enhancement filtering is used to improve the picture before nodule identification and feature extraction. Lastly, the Support Vector Machine (SVM) classifier categorizes lung nodules. With lung segmentation, the overall sensitivity has increased, and the number of false positives in each scan has dramatically decreased. With just 2.19 false positives per scan, the sensitivities at the detection and classification phases reach 94.20% and 98.15%, respectively.[13]

Lucas O. Teixeira et al. illustrated the impact of lung segmentation in COVID-19 detection using CXR images (Chest X-Ray) and assessed which image components had the most influence. An F1-Score of 0.88 was obtained for the classification of COVID-19 utilising segmented images. For COVID-19 identification utilising segmented images in the cross-dataset situation, they got an F1-Score of 0.74 and an area under the ROC curve of 0.9. [14]

2.7.3 Importance of Brain Tumor Segmentation

Brain tumor segmentation is crucial for medical image processing. Early diagnosis of brain tumors is essential for treating brain tumor patients. The patient's chances of survival will increase with early brain tumor detection. Brain tumor segmentation also provides helpful information for diagnosis and treatment planning.

2.8 Image Segmentation Models

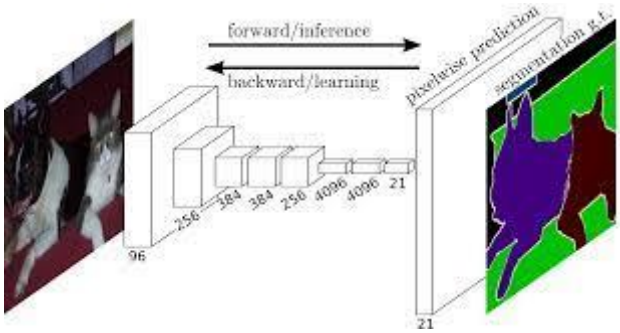


Figure 8: Architechture of Fully Connected Network (FCN)[15]

In 2015, a Fully Convolutional Network (FCN) was proposed. FCN only includes convolutional layers, allowing it to handle images of any size and produce the segmentation map, which is the same size as the input image. The authors modify existing CNN architectures like VGG16 and GoogleNet by replacing all fully connected layers with fully convolutional layers. FCN enables the model to produce a spatial segmentation map instead of a classification score, as each pixel in the input image is labelled with the class it belongs. [15]

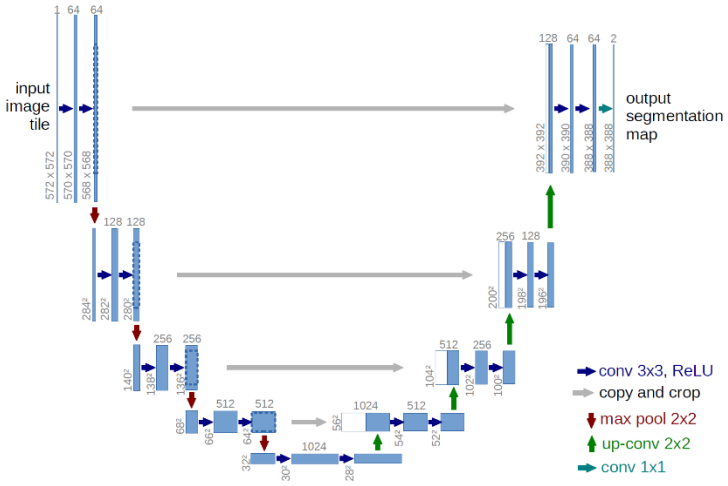


Figure 9: Architechture of U-Net[16]

In 2012, Ciresan and his team won the EM (electron microscopy image) segmentation challenge in ISBI using modified CNN. The image is separated into patches (small regions) and made as an input, and they propose a sliding window technique to predict the class of each patch of the image. Besides, their modified CNN can localize (identify the target's position). However, there are disadvantages to this model. The model is very slow because each patch needs to run separately. Besides, there is a trade-off between context information and localization precision. If the image is divided into small patches, they will get little context information. If the image is divided into large patches, they will get low accuracy on localization because more max pooling layer is needed. Therefore in 2015, U-Net was proposed to have high localization accuracy and high-resolution context information simultaneously without trade-off. The U-net is built with a modified fully convolutional network to get more

precise segmentation with less training data. The pooling operators are replaced with upsampling operators to increase the resolution of the output. High-resolution features from the contracting path (encoder path) are connected to the upsampling path (decoder path) to let the model localize. Many feature channels were added to the upsampling path to allow the model to propagate context information to the output. [16]

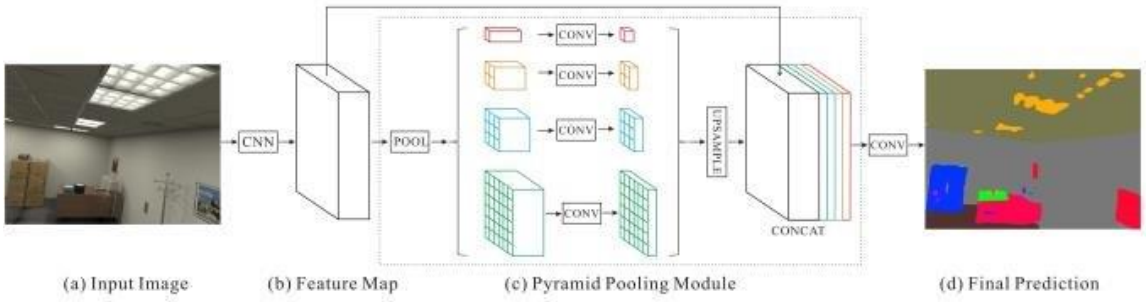


Figure 10: Architecture of PSPNet[17]

In 2016, PSPNet (Pyramid Scene Parsing Network) was proposed. It uses ResNet to extract features from the input image, and then the feature is used as input to the pyramid pooling structure to differentiate their feature. The image is pooled in 4 different sized convolution layers and then upsampled to make them smaller in size. The upsampled result is then concatenated with the feature extracted by ResNet to collect context data. Lastly, the pixel-by-pixel result is outputted by the convolution layer. [17]

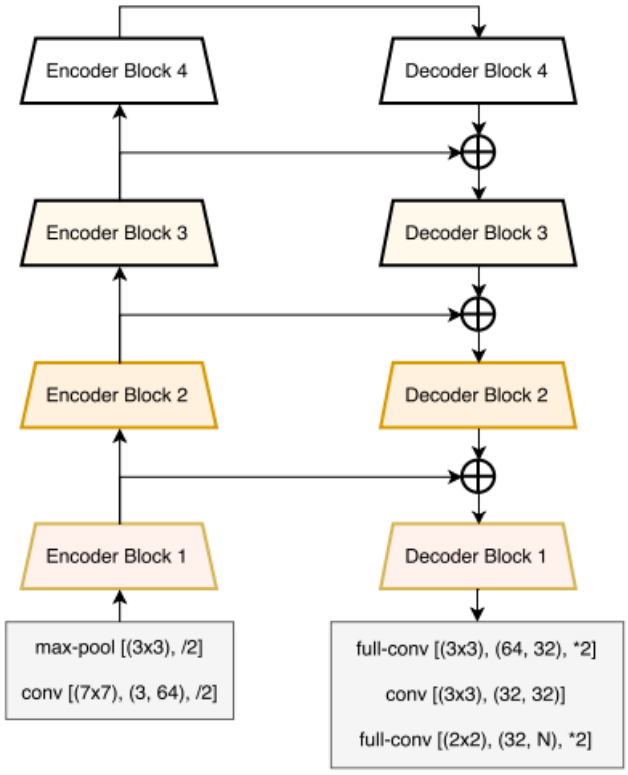


Figure 11: Architechture of LinkNet[18]

In 2017, LinkNet was proposed. Linknet consists of an encoder path, a decoder path, and skip connections. Linknet combines the feature map from the encoder with the corresponding decoder using element-wise addition. LinkNet can help the model to propagate important features from the encoder to the decoder, which is essential for accurate segmentation.[18]

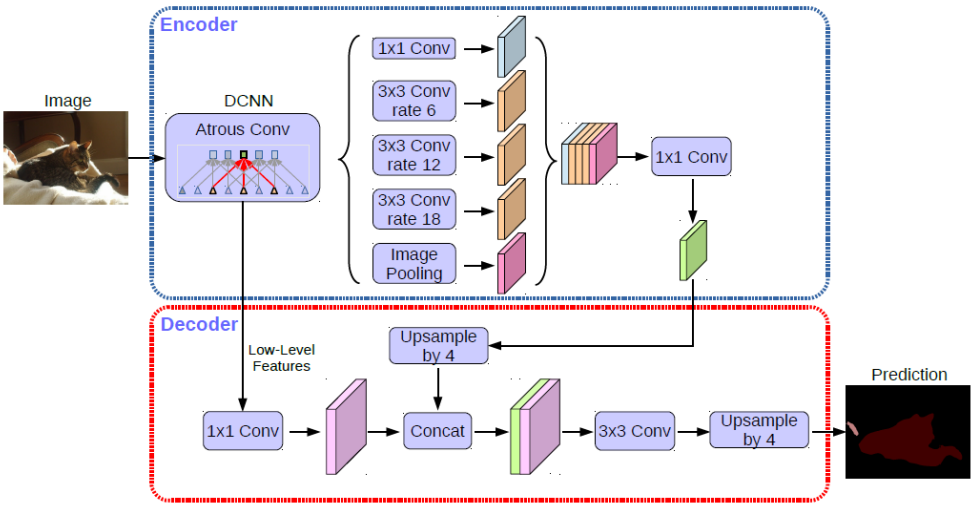


Figure 12: Architecture of DeepLabV3+[19]

In 2018, DeepLabV3+ was proposed. It uses dilated convolution to resolve the low-resolution problem in the model caused by max pooling. It also uses Atrous Spatial Pyramid Pooling to examine the feature map at different sample rates and reliably partition the image in various sizes. [19]

2.8.1 Comparison of DeepLab and U-Net

DeepLab and U-Net are the two most popular architectures for medical image segmentation, so they are selected for comparison. Both have their strengths and weaknesses.

	DeepLab	U-Net
Architecture	More complex architecture using atrous (dilated) convolutions and atrous spatial pooling pyramid pooling	It is easy to build and train due to its simple yet effective encoder decoder architecture.
Parameters	Has more parameters, increasing its capability for complicated datasets.	Has fewer parameters which suitable for limited datasets and computational resources.
Data Handling	Utilizes atrous spatial pyramid pooling (ASPP) to accommodate multiple scales features	Utilizes network's encoder and decoder path to effectively capture and localize important features

In conclusion, U-Net is a viable option for smaller data sets and limited computing resources. In contrast, DeepLab is often better suited for applications demanding fine details and multi-scale processing. Ultimately, the decision between the two architectures will be based on the specific details of the work and the available computational resources.

In this project, U-Net was chosen because a low-parameter model is needed to be implemented on an edge device for fast and efficient inference. Besides, U-Net is simple to modify, so it is more suitable for this project because the objective is to modify and investigate the pooling layers of the model.

2.8.2 Advantage of U-Net for medical image segmentation

Due to its distinctive design and capabilities, U-Net's popular deep-learning architecture is a good choice for medical image segmentation. These are some of the factors that make U-Net suitable for this medical image segmentation.

First, U-Net can precisely segment the region of interest with a small amount of training data. Obtaining segmented medical images is time and money-consuming because it requires a professional radiologist. Therefore, the ability to train with a small amount of data is a plus for U-Net in the medical image segmentation field. [16]

Second, U-Net is good at handling imbalanced data, which often happens in medical images. Foreground or background imbalance is very common in medical images. For example, in the brain tumor dataset, some of the medical images has very small tumor while the rest of the images are background which causes most of the model to overlook the data. U-Net utilizes loss functions like dice loss to penalise the model when it misclassifies the tumor as background. [16]

Third, U-Net is very flexible for medical semantic segmentation tasks. The U-Net can adapt to different datasets and output good results by tweaking its training parameters or structure. Piao et al. improve the accuracy of U-Net by modifying the convolutional layer.[20] Besides, U-Net is easy to customize due to its simple architecture. For example, Owais Ali et al. modified the number of filters and number of layers in U-Net to reduce the size of U-Net so that it can increase the inference speed when running on edge devices.[1]

Forth, U-Net is highly scalable for medical semantic segmentation tasks. Deep learning is a fast-growing field, and more techniques are being proposed. Many U-Net variants are introduced, like U-Net++, ResNet, and Attention U-Net, based on the newly developed technique. These variants show that U-Net is highly scalable, and the basic architecture of U-Net will be used and improved for a wide range of segmentation tasks. For example, an attention mechanism is applied in U-Net to let U-Net focus on the critical area to improve accuracy (Attention U-Net).[20]

In summary, U-Net is still a widely adopted model in the medical image community after so many years because of its capability to train with small datasets, manage imbalanced data, and have high flexibility and scalability. U-Net is set to play a significant role in developing cutting-edge medical image analysis systems thanks to its ability to work effectively on challenging and complex medical images.

2.9 Types of Edge Device

An edge device is a computing device that controls data flow at the boundary or "edge" between two networks. Edge devices can process data, store, monitor, and transmit between networks. Edge devices are located close to the data source to reduce latency for real-time processing. Edge devices have become more critical because of the increased demand for cloud computing and the Internet of Things (IoT).[21]

There are many types of edge devices, including traditional edge devices, intelligent edge devices, and AI (Artificial Intelligence) or Machine Learning (ML) edge devices. Traditional edge devices, like edge routers, transmit data across a secure network with minimal computational power. Intelligent edge devices like sensors can carry out edge computing operations close to the data source for industrial automation. AI or ML edge devices like smartphones can conduct complex analysis or AI and ML inference like real-time processing, image recognition, natural language processing, and voice recognition. [22], [23] In this project, AI or ML edge device are used to carry out medical semantic segmentation.

2.9.1 Importance of Edge Device for medical field

1. Real time processing

Real-time processing is frequently needed for medical image analysis, especially when making judgements quickly in an emergency. Edge devices could carry out on-site image segmentation and other analytic operations without sending data to a distant server. Edge devices enable faster analysis and decision-making times. 30% of all data worldwide comes from the healthcare industry based on Deloitte and MIT Technology Research. With the volume of data generated by the healthcare industry, gathering and drawing insights in real time is essential to support quicker clinical choices. [24]

2. Enhanced privacy and security

Edge devices can protect the privacy and security of patients because the medical information of patients is confidential and sensitive. Patients' images may be segmented and analysed locally on edge devices, making them less susceptible to security vulnerabilities and data breaches.[24] Naif et al. outlines the typical methods researchers use to include privacy in their healthcare solutions and highlights the shortcomings of these solutions in terms of their longevity, technical complexity, and efficacy. So, the article suggested using edge devices to protect the security of the data. [25]

3. Economic Impact

By eliminating the demand for costly hardware and software infrastructure, edge computing can help reduce the price of medical image diagnosis. Edge devices can be used to segment data and do analysis locally with low-cost edge devices instead of depending on a central server. The need to transfer high-bandwidth data, such as large medical images or video streams, across the network or off-premises is reduced by AI processing at the edge.[24] Md. Golam et al. suggested a cost-effective patient care system with edge computing to reduce healthcare data processing costs. [26] Nowadays, each hospital bed in American hospitals has between 10 and 15 edge devices linked to it that constantly monitor the patient's condition. It is predicted

that 75% of medical data will be produced at the edge by 2025. Also, the global market for connected medical devices is anticipated to increase from $41 billion in 2017 to $158 billion in 2022. [24]

4. Scalability

Other than that, edge devices are very scalable. Edge devices may be easily installed in several locations, including hospitals, rural clinics, and even patients' homes. Edge devices enable medical image analysis to be scaled to a much broader population, boosting access to treatment, and assisting in the eradication of healthcare inequities.[27]

5. Robust infrastructure and Reliability

With built-in failover capabilities, edge devices can be made to be very reliable. Healthcare facilities can continue operating normally even during network failures by processing data locally on edge devices. [24] For medical applications, a delay in the analysis might have significant repercussions. So, reliability is a big concern in medical imaging. Todd J. Traver stresses the critical role of edge computing in supporting the fast-growing IoT ecosystem in his conclusion and the need to adopt a holistic strategy to ensure reliable deployments of edge devices. [28]

2.10 U-Net Basic Architecture

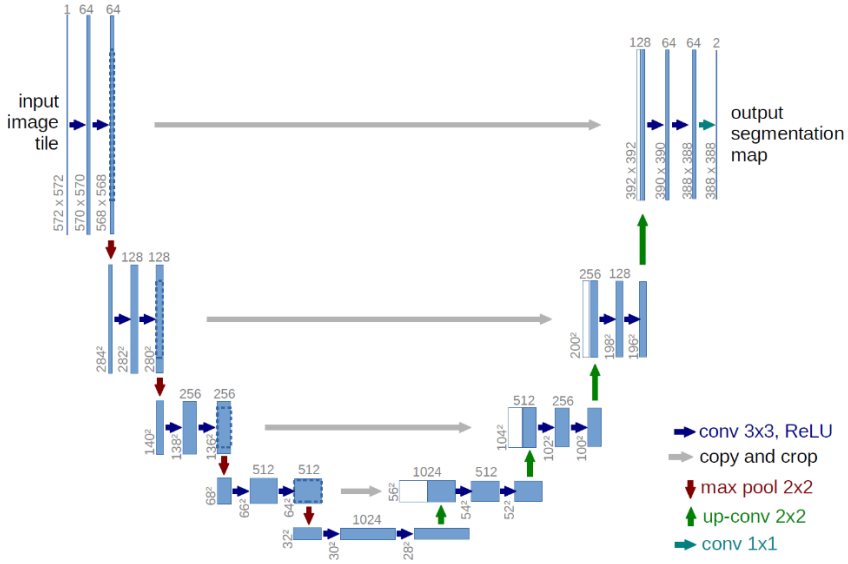


Figure 13: Architecture of U-Net [16]

The U-Net collects "what" and "where" information to accomplish the segmentation. "where" information is crucial for segmentation, but a standard CNN does not retain it in image classification. Consequently, a corresponding up-sampling network is added to the U-net to recover the lost context. U-net comprises a downsampling (on the left) and upsampling (on the right) path, also known as the encoder and decoder, as shown in Figure 13. Encoder is utilised to determine the context of input images. The encoder recognizes "what" is in the input image. Decoder is the symmetric expanding path for the encoder network. The position of the region of interest in the input image is located by the decoder.

In the downsampling path, the number of filters in each layer gradually increases, and the size of the feature maps gradually decreases. The contracting path comprises a series of convolutional layers (to produce the feature maps) followed by pooling layers (to reduce the feature map size). When the number of filters increases, the number of feature maps produced increases. The model can capture a wide range of complex features. These features are also called fine-grained features, which means that the features are on a small scale, and they are localized.

In the upsampling path, the number of filters in each layer gradually decreases, and the size of the feature maps gradually increases. The upsampling path comprises a series of transposed convolutional layers followed by the concatenation of the feature map from the encoder layer.

The concatenation of the feature map from the encoder layer to the corresponding decoder layer is called "skip connection". The skip connection is shown in Figure 13 with a grey arrow. The decoder layer utilised the skip connections to get the fine-grained details provided by the encoder layer to localize the position of the feature.

The purpose of the upsampling convolutional layers is to recover the lost details in the downsampling path by increasing the size and resolution of the feature map back to a full-size image.

2.11 Basic Structures inside U-Net

2.11.1 Convolutional Layers

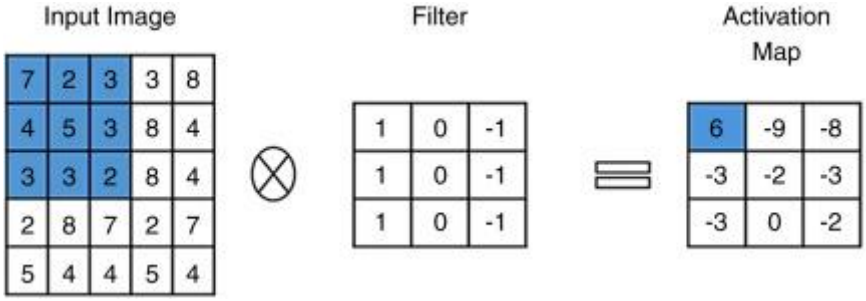


Figure 14: Convolutional Layers [29]

Convolutional layers are essential for CNN and U-Net. It has several filters (kernels), and the parameters of the filters will be learned and improved throughout the training process. The original image is often bigger than the filters' size. Each filter produces an activation map (feature map) after it convolves with the image. The filter is slid over the image's height and width in the convolution process. At every spatial location (pixel), the sum of the dot product between each filtered pixel and the input is evaluated. [29]

Based on Figure 14, the first element of the activation map (highlighted blue) is determined by the sum of the dot product between the highlighted pixels in the image and the filter. This process is repeated for every input image pixel to produce every element of the activation map. The activation maps of each filter are stacked along the depth dimension to create the convolutional layer's output volume.

Every element of the activation map may be a neuron's output. As a result, each neuron is connected to a discrete local area in the input image, and the area's size is equal to the filter's size. Each neuron in an activation map has parameters in common. The convolutional layer's local connection forces the network to learn filters that have a maximum response to a particular local area of the input. The early convolutional layers extract the low-level features like lines of images, whereas the later layers extract the high-level features like shapes and specific objects.

The Receptive field is a frequently utilised term, the area of the input volume that the selected feature extractor (filter) is examining. Based on Figure 14, the receptive field is the blue highlighted part in the input image that the filter covers.

2.11.2 Pooling Layers

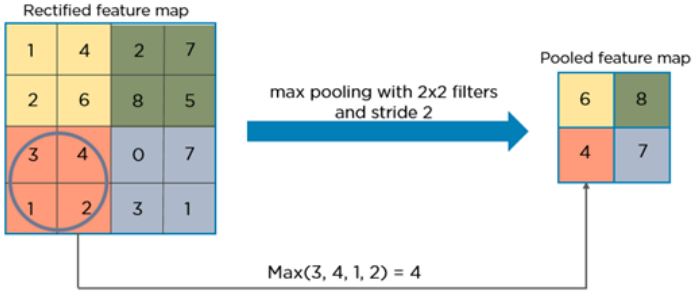


Figure 15: Max pooling Layer [30]

Pooling layers is crucial in U-Net and CNN to downsample the size of feature maps so that fewer parameters are trained while retaining the essential features. Based on Figure 15, the max pooling operation with stride has successfully reduced the feature map size from 4x4 to 2x2 by only choosing the maximum pixels in each region.

2.11.3 Stride

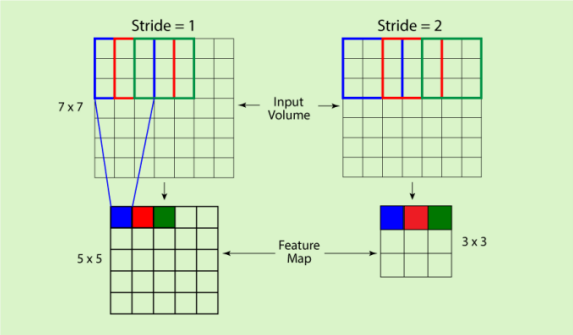


Figure 16: Stride [31]

Stride is an important parameter for convolution layers because it determines the size of the output feature map. A computer reads an image from the top left to the bottom right corner. Stride means how much the filter moves on the image or vice versa.

Based on Figure 16, if a 7x7 input image is convolved by a 3x3 filter with a stride of 1 and a 5x5 feature map is produced. If a 7x7 input image is convolved by a 3x3 filter with a stride of 2, a 3x3 feature map is created.

2.11.4 Padding

Based on the discussion above, the convolutional layers will reduce the size of the output feature map. So, padding layers can expand the output size while preserving the information provided in the corners. Padding increases the outside dimension of the image's rows and columns. As a result, the size of the input data and the output data will stay identical. [32]

The same padding is used in my U-Net convolution layers by adding zero values to the other frame of the feature map so that it has the same size as the input.

2.11.5 Activation Function

The activation function in a neural network converts the node's summed weighted input into the node's activation. In U-Net, rectified linear activation function (ReLU) was used as an activation function for the hidden layers.

Several limitations exist to traditional activation functions like the sigmoid and Tanh functions. The first limitation is that these functions suffer from a vanishing gradient problem. This problem occurs when the deep layers in the network fail to receive the gradient because the gradient becomes very small during back-propagation, making the network difficult to learn. The second limitation is that they saturate, which means the output value is very close to 0 or 1, making the network hard to learn. The third limitation is that the functions are only sensitive to variations near the mid-point of their input. Therefore, ReLU is introduced to tackle the vanishing gradient problem and prevent saturation. [33]

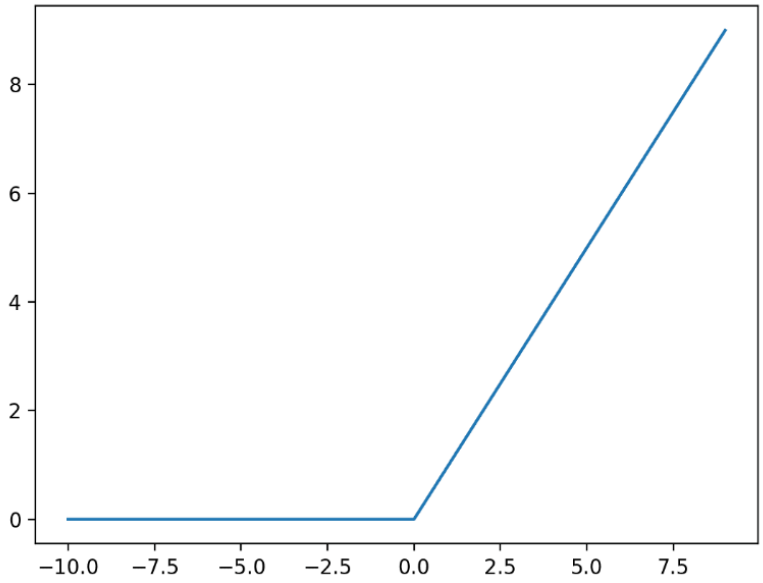


Figure 17: ReLu activation function [33]

Based on Figure 17, ReLU will output the input directly if the input is positive and zero if the input is negative. Most models utilise it because it is simple to train and performs well.

2.11.6 What happened after convolution and pooling?

The height and width of the image gradually decrease in a typical convolutional network because of downsampling due to pooling and convolution. The reduced size of the image allows the filters in the deeper layers to concentrate on a broader receptive field (context). The number of filters constantly increases to extract more complicated characteristics from the image. The pooling process leads logically to the following conclusion. The model gains a better understanding of "What" is in the image by downsampling, but it loses information about "Where" it is present. [30]

Semantic segmentation produces more than just a class label or a set of bounding box parameters. The result of semantic segmentation is a comprehensive, high-resolution image with pixel classification. "What" and "Where" information are required for semantic segmentation. It is necessary to upsample the image (change a low-resolution image to a high-resolution image) to retrieve the "Where" information. There are several methods to upsampling an image. However, transposed convolution is the preferred option for upsampling an image in most cutting-edge networks.

2.11.7 Transposed Convolution

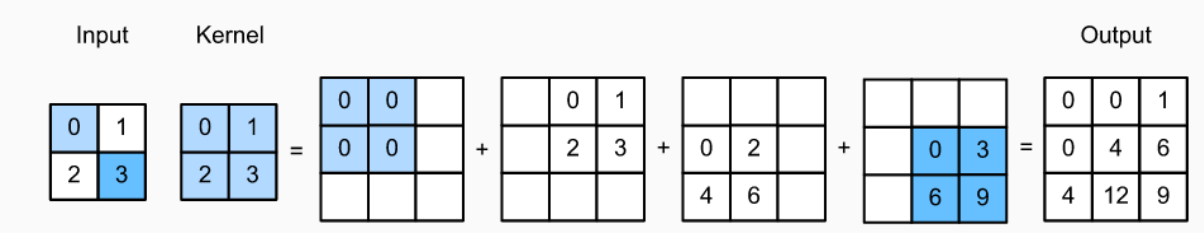


Figure 18: Transposed Convolution [34]

A transposed convolutional layer is an upsampling layer that produces an output feature map bigger than the input image. A transposed convolutional layer operates similarly to a regular convolutional layer but performs convolution in the opposite way. Based on Figure 18, each element in the kernel is multiplied by the corresponding element of the output feature map. The results of these element-wise multiplications are then summed up to produce a bigger output feature map. The stride and padding parameters of the layer can be used to modify the output's size. Based on Figure 18, the first pixel of the input is multiplied with every element in the filter (kernel), and the output is a 2x2 value in a 3x3 feature map. With the stride of 1, the same process is repeated for every input element, and the sum of all the elements in the feature map will be the output.[35]

2.12 Basic Metrics

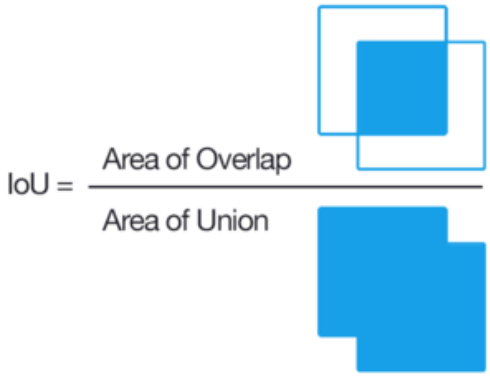
A metric is a function used to assess how well your model works. Metric functions are comparable to loss functions, except that the outcomes of a metric's evaluation are not considered during model training.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Figure 19: Confusion Matrix[36]

There are four categories for the assessment unit for semantic segmentation on a single pixel:

1. True positive: the pixel was classified correctly as a class of interest.
2. True negative: the pixel was classified correctly as the background class.
3. False positive: the pixel was classified incorrectly as a class of interest.
4. False negative: the pixel was classified incorrectly as the background or a different class.

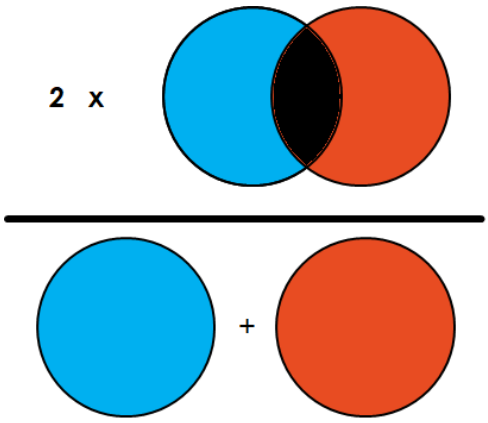


$$IoU = \frac{TP}{(TP + FP + FN)}$$

Figure 20: Intersection-Over-Union (IOU) [37]

Figure 21: IoU formula[38]

Intersection-Over-Union (IOU), often known as the Jaccard Index, is one of the most popular metrics in semantic segmentation and with good reason. The IOU is a highly effective metric that is relatively simple to use. Based on Figure 20, the IOU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. This metric ranges from 0–1. Zero denoting no overlap at all, and one denoting complete overlapping segmentation. [37] The formula for IOU is True Positive divide by the sum of True Positive, False Positive and False Negative.



$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

Figure 22: Dice Coefficient [37]

Figure 23: Dice Coefficient Formula[39]

Dice Coefficient is two times the Area of Overlap divided by the total number of pixels in both images. Similar to IoU, this metric ranges from 0–1, and 1 denotes the highest similarity

between the ground truth and prediction. [37] The formula for Dice Score is two times True Positive divide by the sum of two times True Positive, False Positive and False Negative.

Both metrics used are not affected by the class imbalance issue. The class imbalance issue is caused when one class or a few classes dominates the image.

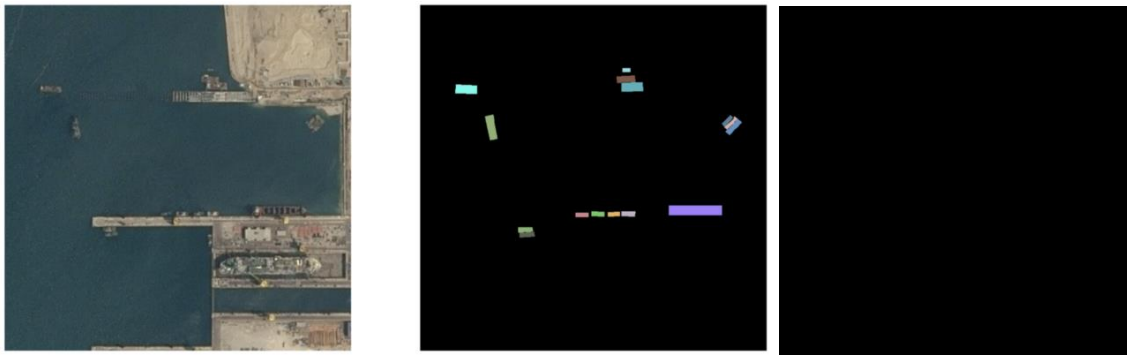


Figure 24: Input image, Ground Truth Mask and Predicted Mask[37]

Based on Figure 24, the left image is an image, the middle image is the segmented mask, and the right image is the predicted mask. If accuracy metrics are used, the score will be 95% because the background class dominates the image. If dice coefficient and IOU metrics are used, the score obtained will be 47.5% for both metrics. The dice coefficient and IOU are a better indication of how our segmentation model performs.[37]

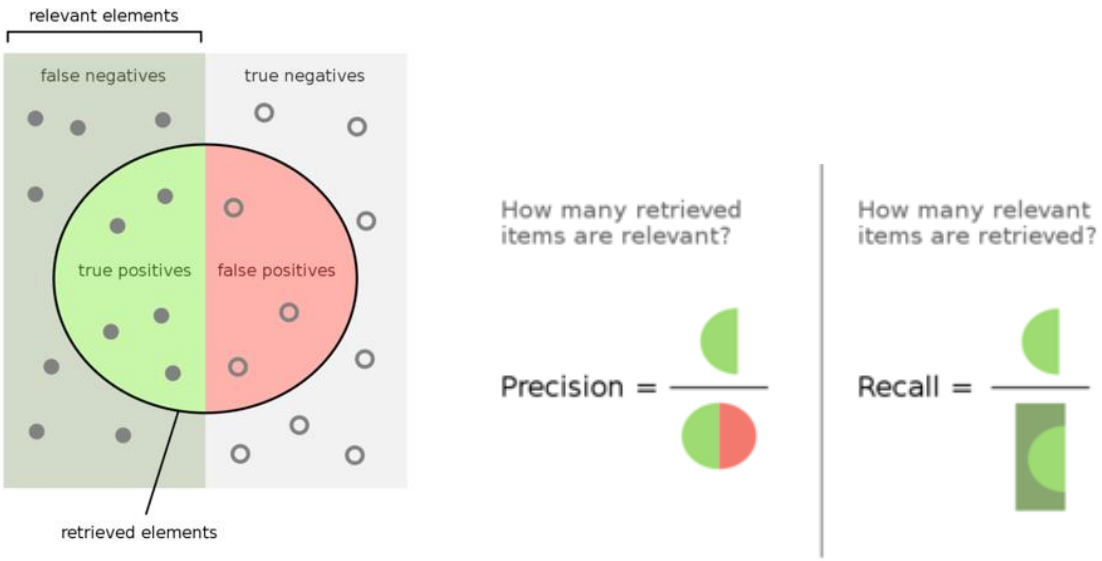


Figure 25: Precision and Recall[40]

Recall and Precision are also often used to evaluate segmentation results. Based on Figure 25, Precision is True Positive divided by the sum of True Positive and False Positive, while Recall is True Positive divided by the sum of True Positive and False Negative.

3.0 Methodology

3.1 Modification made for U-Net

There are two types of U-Net which are 3D U-Net and 2D U-Net. 3D U-Net trains with a 3D dataset, and 2D U-Net trains with a 2D dataset. Both datasets are trained because different datasets will have different results for pooling layers. The 3D dataset chosen is Brain Tumor Segmentation Dataset (BRATS), while the 2D dataset chosen is Chest X-Ray Dataset.

The first sub-aim of this project is to investigate the effect of different pooling layers on U-Net. The reason to investigate pooling layers instead of other components of U-Net is that different pooling layers can provide different ways to downsample the feature maps and extract important features from the image, significantly impacting the results. Besides, by modifying the pooling layers, the number of parameters of the U-Net will not increase.

A lot of paper has investigated and shown results on how pooling layers can affect the performance of a model. However, researchers/students have yet to investigate the effect of different pooling layers on U-Net and how they perform with different datasets. Therefore, it is a great chance to show that modifying pooling layers can improve the performance of U-Net without any trade-off, like increasing the number of parameters.

The architecture of 3D U-Net and 2D U-Net are slightly different in terms of convolutional kernel shape because they are built to train different dimension datasets. Besides, the training parameters like loss function and regularization are slightly different because they are slightly adjusted for each of the U-Net respectively. Other than that, the number of parameters of U-Net is reduced while retaining comparable results.

The architecture of the 2D and 3D U-Net remains the same when comparing the performance of different pooling layers; only the pooling layers are modified. 2D and 3D U-Net are trained and compared separately. For 3D U-Net, two types of pooling layers tested are max pooling and average pooling. For 2D U-Net, four types of pooling layers tested are max pooling, average pooling, hybrid pooling and Atrous Spatial Pyramid Pooling. The U-Nets are trained until the result converges before comparing their performance.

The second sub-aim of this project is to reduce the number of parameters of U-Net to implement it on edge device. The way to reduce the number of parameters is to reduce the number of filters in U-Net because number of filters contributes the most to the number of parameters.

3.1.1 Modification made for pooling layer of U-Net

In the original U-Net paper, only max-pooling was used. For this project, three more types of pooling layers are used to investigate the performance of U-Net.

3.1.1.1 Max Pooling layer

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, Max pooling layer is tested.

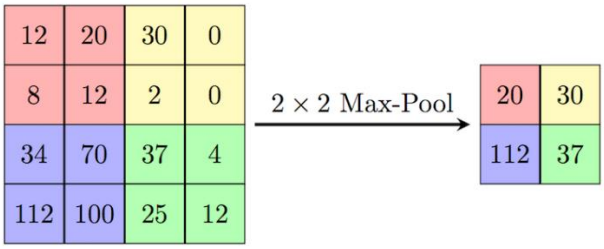


Figure 26: Max Pooling Process [41]

Max Pooling determines the maximum value among the patches of a feature map and uses it to create a downsampled feature map. Max Pooling retains the most prominent features of the feature map, so the resulting image is sharper than the original.

3.1.1.2 Average pooling Layer

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, Average pooling layer is tested.

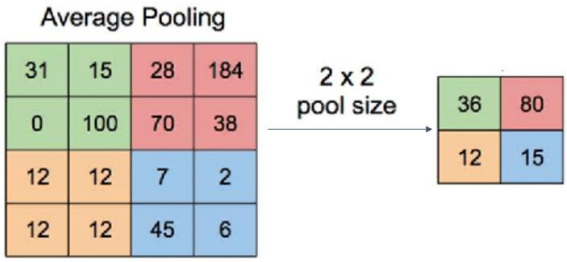


Figure 27: Average Pooling Process [42]

Average pooling determines the average value among the patches of a feature map and uses it to create a downsampled feature map. The average values of the features on the feature map are preserved through average pooling. While maintaining the essential qualities of the feature, it blurs and smooths the image.

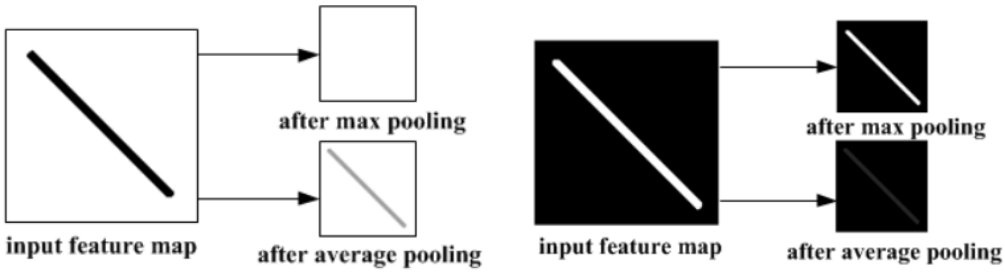


Figure 28: Result of Average and Max pooling with white and black background [43]

Figure 28 shows the effect of max pooling and average pooling when applying an image with a white and black background. Max pooling works well on black background, while average pooling works well on white background.

3.1.1.3 Hybrid Pooling Layer

In Chest X-Ray 2D U-Net, Hybrid pooling is one of the pooling methods tested. Tong, Zhiqiang et al. suggest hybrid pooling to enhance the generalisation ability of Convolutional Neural Network (CNN). Pooling techniques like the max convolution layer might cause context loss and make the result worse when training on a complicated dataset. The max pooling and average pooling are combined in the hybrid pooling layer to overcome this weakness. Hybrid pooling methods are tested on CIFAR-10, CIFAR-100, and ImageNet datasets and compared with other popular pooling layers, showing that it can significantly improve performance. [44]

3.1.1.4 Atrous Spatial Pyramid Pooling Layer

In Chest X-Ray 2D U-Net, Atrous Spatial Pyramid Pooling is one of the pooling methods tested. He, Kaiming et al. suggest Atrous Spatial Pyramid Pooling (ASPP) to improve the performance of CNN. Typically, CNN uses pooling layers to reduce the size of the feature map but cause a loss in context, especially for images with different size. The strength of ASPP is that it can effectively extract context information from the image with different sizes because it splits the input feature map into a fixed number of regions before applying max pooling to each of them. It is put between convolutional layers and fully connected layers.[45]

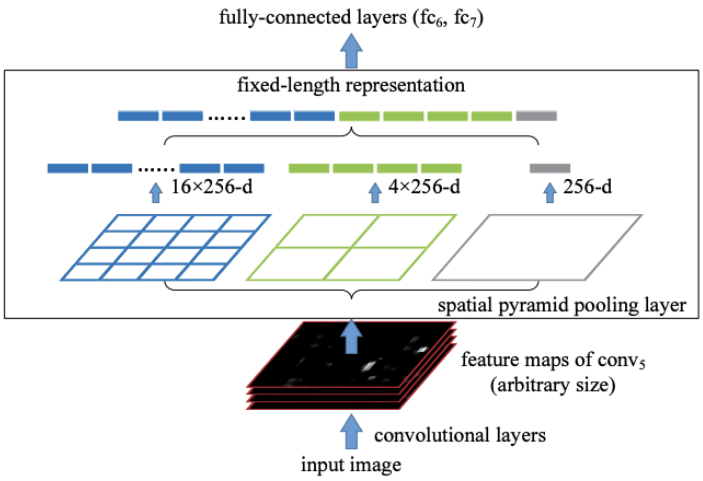


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

Figure 29: Architecture of Spatial Pooling Layer [45]

Based on Figure 29, the feature map generated by the convolution layer is split into three sizes before applying max pooling and connecting it to fully connected layers. He et al. shows that adding ASPP can significantly improve performance on PASCAL VOC 2007 and 2012 datasets.

3.1.2 Modification made for number of filters in U-Net

The number of filters in U-Net affects the number of parameters in U-Net. The filters are learnable parameters to generate feature maps that capture important features from the input image. When the number of filters increases, the number of parameters increases.

In the original paper, the original U-Net uses 64 filters at the starting layer, with 34.5 million parameters. **For my BRATS 3D U-Net,** I use 16 filters at starting layer and reduce the number of parameters to 10 million parameters. **For my Chest X-Ray 2D U-Net,** I use 8 filters at the starting layer and reduce the number of parameters to 0.7 million parameters. There are several reasons for this modification.

The first reason is that U-Net can train faster because the model becomes less complex, and it can save time and computational resources.

The second reason is that the risk of overfitting for U-Net is reduced. More complex models and models with more parameters are more prone to overfitting. U-Net with less parameters has better generalization ability as it can still capture important features from new data that have never been seen before.

The third reason is that low parameter U-Net has low memory and processing requirements. A low parameter U-Net requires less memory to store and process, which is suitable for resource-limited devices like edge devices. Besides, less parameter U-Net need less processing power to make predictions, especially in edge devices.

3.1.3 Modification made for U-Net from 2D to 3D.

In the original U-Net paper, 2D U-Net was proposed. However, there are a lot of 3D medical datasets from MRI and CT scans that need to be trained. So, the 2D U-Net was modified to a 3D U-Net to train and segment 3D datasets. First, the input layer is modified to accept 3D data with height, width and depth dimensions instead of 2D data with only height and width dimensions. The convolutional kernel size is 3x3 in the convolutional layers for the original 2D U-Net, and it is modified to 3x3x3 to produce 3D feature maps. The pooling layers and transposed convolutional layers are also modified to downsample and upsample 3D feature maps by adding another depth dimension to all the layers.

3.1.4 Modification made for output activation function.

For BRATS 3D U-Net, activation function for the output layer is softmax activation. For multiclass segmentation, the goal is to assign every input image pixel to one of the classes. Softmax activation allows the model to predict multinomial probability distribution. Simply put, it will predict the probability of every class for every image pixel. [46]

For Chest X-Ray 2D U-Net, activation function for the output layer is sigmoid activation. For binary segmentation, the goal is to assign every input image pixel to either the object of interest or the background. Sigmoid activation acts as a squash function to squash the model's output from 0 to 1. The model can predict the probability of the object of interest and background for every pixel. [47]

3.1.5 Modification made for regularization method.

3.1.5.1 Dropout

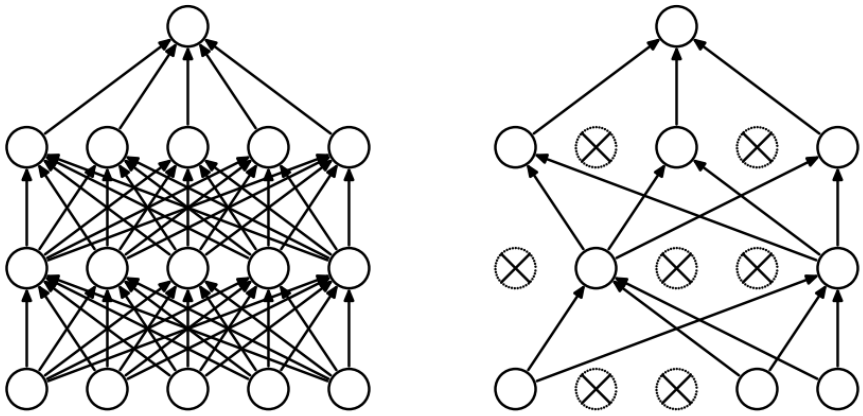


Figure 30: Network before and after applying dropout. [48]

Dropout is a computationally cheap yet remarkably effective regularization technique to prevent large and complex neural networks from overfitting and improving generalization errors. During training, some neurons will be ignored or "dropped out". Dropout means that the weights of the neurons ignored will not be updated. In every iteration of training, different neurons will be ignored so that the neural network does not rely on only specific neurons, but it lets all the neurons be equally trained. Based on Figure 30, the connections in the network applied with dropout are much less than in the standard neural network. The dropout rate is the ratio of the number of neurons ignored over the total number of neurons. In most networks, the dropout rate is set between 0.5 to 0.8, meaning that 50% to 80% of the neurons are ignored in each layer. [49]

In the original U-Net paper, dropout is applied to the hidden layers, but the dropout rate is not mentioned. **In my BRATS 3D U-Net**, the dropout rate is set initially at 0.1 for the first two encoder layers and gradually increases until 0.3 at the end of the encoder layer. As the encoder layers go down, the image will become smaller, but the number of feature maps will increase due to the increasing number of filters. When the number of feature maps (context) increases, the model is more likely to overfit. Therefore, the dropout rate gradually increases along the encoder layer to prevent overfitting. Likewise, in the decoder layer, the dropout rate decreased from 0.3 to 0.1 because the number of feature maps was also reduced. In other U-Net papers, they might use a higher dropout rate, but 0.1 to 0.3 was chosen because if the dropout rate is too high, it might cause underfitting.

3.1.5.2 Early Stopping

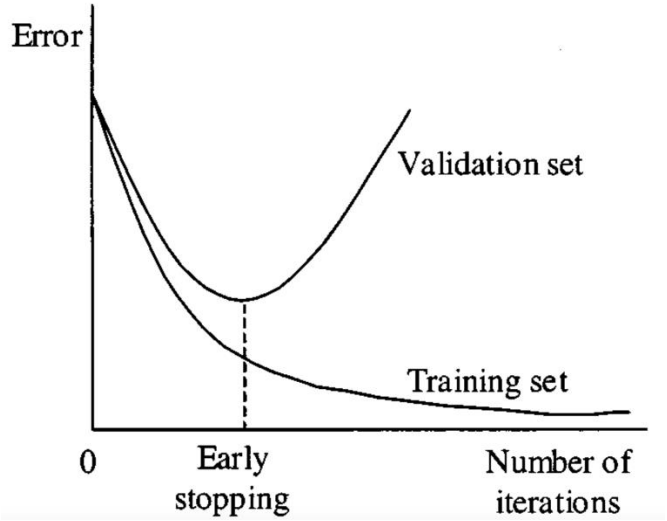


Figure 31: Plot of Error vs Number of Iterations for Training and Validation Set [50]

Early stopping is another type of regularisation technique used to stop the training of the neural network earlier before it has overfitted over the training data. Early stopping can reduce overfitting and improve generalization at the same time. How long it takes to train neural networks is a significant problem. The model will underfit the training and test sets if insufficient training exists. A model that trains too much will overfit the training dataset and perform poorly on the test set. A compromise has to be made by training on the training dataset with a stop at the point where the error on the validation dataset begins to increase. [51]

Based on Figure 31, the error of the training and validation set decreases until the error of the validation set goes up. That is the early stopping point to stop the model from continuing training which can reduce overfitting and improve the model's generalization.

In the original U-Net paper, the early stopping is not applied to the hidden layers. **For Chest X-Ray 2D U-Net,** early stopping is used for regularization. The early stopping monitors the validation loss and stops the training process when it overfits. Another parameter called patience was set to delay the stopping of the training. Neural network training is noisy and unpredictable. The validation loss will go up and down a few times when plotted on the graph. This scenario means that the early stopping function will stop the training due to noise if the patience value is not set. This delay will let the early stopping function observe if the validation error continues to go down or it is just a local minimum.
[51]

3.1.6 Modification made for Training Parameters for U-Net

3.1.6.1 Loss function

The loss function is crucial because it determines how well the model is learning and performing and most importantly, the one that improves the model.

In the original U-Net paper, the loss function used is cross-entropy loss. **For BRATS 3D U-Net,** the loss function is focal loss plus dice loss. **For Chest X-Ray 2D U-Net,** the loss function used is dice loss.

Focal loss is an enhanced version of cross-entropy loss that addresses the issue of class imbalance by giving harder or more susceptible to misclassification examples (such as backgrounds with noisy textures, partial objects, or the object of my interest) more weight while de-weighting simple examples like backgrounds. As a result, focal loss decreases the loss contribution from simple instances and emphasises the need to classify hard examples correctly.[52] The focal loss was initially proposed to improve the one-stage object detector, but due to its ability to tackle imbalanced classes, it can perform well in multiclass semantic segmentation.[53]

Dice loss is derived from the dice coefficient, which determines the similarity between two images. The advantage of the focal loss and dice loss function is that it works well with imbalanced class datasets. It is important because background class imbalance is prevalent in medical images. [54]

3.1.6.2 Metrics

In the original U-Net paper, only the IOU score was used to evaluate the performance. In my U-Net model, I use three more metrics to evaluate and analyse the performance of my U-Net.

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, the IOU score was used to evaluate the model's performance. IOU score measures the area of overlap between the predicted mask and the original ground truth mask. The IOU score is chosen to evaluate U-Net trained with 2D and 3D datasets for several reasons.

First, the IOU score is easy to analyse. It rewards the true positives and penalise the false positives and false negative of the model based on its formula. Three of these metrics are equally weighted.

Second, the IOU score isn't affected by imbalanced class problems, which is common in medical segmentation images. [54]

Third, the IOU score has a helpful property which is scale invariance. It means that it only focuses on the area of the segmented object regardless of the size; therefore, it is chosen to evaluate the performance of both 2D and 3D U-Net, which might vary in size and orientation. [55]

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, the Dice score was used to evaluate the model's performance. However, the dice score is chosen to evaluate U-Net trained with 2D and 3D datasets for several reasons.

First, the dice score is more sensitive to small changes in the segmentation accuracy (true positives). It is because it puts more weight on rewarding true positives than penalising false positives and false negatives. It makes it a good choice to evaluate U-Net, especially dealing with a complex dataset.

Second, the dice score isn't affected by imbalanced class problems, which is common in medical segmentation images. [54]

Third, the dice score is complementary to the IOU score. Dice score is often used with IOU score because dice score is more sensitive to true positives, while IOU score is more sensitive to false positives and false negatives.

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, the precision score is used to evaluate the model's performance. Precision evaluates the number of true positives in the true positive and false positive.

Precision is good at evaluating false positives of the model. In medical field, false positives can have huge consequences. For example, if a patient is wrongly diagnosed as having a brain tumor, unnecessary treatments or surgery will be applied to the patient. However, it does not consider false negative. Therefore, another metric has to be used to overcome the weakness of precision.

In both BRATS 3D U-Net and Chest X-Ray 2D U-Net, the recall score is used to evaluate the model's performance. Recall evaluates the number of true positives in true positives and false negatives. Recall is good at evaluating false negatives of the model to complement with precision.

Overall, Dice Score and IOU score help detect true positives, Precision helps detect false positives, and Recall helps detect false negatives. All the metrics complement each other so I can find out the strengths and weaknesses of my model.

3.1.6.3 Optimizer

Optimizers are algorithms or techniques that modify your neural network's weights and learning rates to minimise losses and improve accuracy.

The original U-Net paper used Stochastic Gradient Descent as an optimizer. **In both BRATS 3D U-Net and Chest X-Ray 2D U-Net,** Adam (Adaptive Moment Estimation) optimizer is used. Compared to traditional stochastic gradient descent, Adam is unique. Stochastic gradient descent maintains a constant learning rate (alpha) for all weight updates, which does not change throughout training. Adam calculates individual learning rates for various parameters as an adaptive learning rate approach. Adam utilizes estimations of the first and second moments of the gradient to change the learning rate for each weight of the neural network. [56]

Adam combined the benefits of two modified stochastic gradient descent modifications: AdaGrad and RMSProp. AdaGrad, an adaptive gradient algorithm, enhances performance on issues involving sparse gradients (computer vision issues) by maintaining a per-parameter learning rate. Root Mean Square Propagation (RMSProp) also maintains per-parameter learning rates that are adjusted following the mean of recent weight gradient magnitudes. The algorithm performs well on online and non-stationary issues. [57]

Other than that, Adam is rather simple to configure because the default configuration parameters work well for most of the problems.

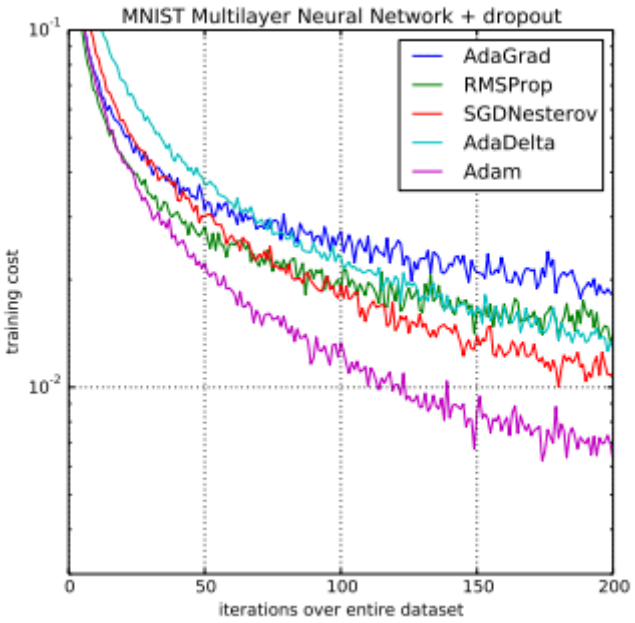


Figure 32: Comparison of convergence speed between different optimizers [58]

D. P. Kingma et al. used Adam for MNIST digit recognition and IMDB sentiment analysis. Based on Figure 32, Adam was proven the fastest to converge among all optimizers.[58]

3.2 Reason why transfer learning is not used.

Model	2D-CNN	3D-CNN	Semi-CNN	
	Params	Params	Pre-Trained Params	Total Params
VGG-16	134.7 M	179.1 M	5.3 M	82.2 M
ResNet-18	11.4 M	33.3 M	0.4 M	31.7 M
ResNet-34	21.5 M	63.6 M	0.8 M	60.5 M
ResNet-50	23.9 M	46.4 M	0.9 M	45.8 M
ResNet-101	42.8 M	85.5 M	0.9 M	84.8 M
ResNet-152	58.5 M	117.6 M	1.4 M	115.6 M
DenseNet-121	7.2 M	11.4 M	0.8 M	10.4 M
DenseNet-169	12.8 M	18.8 M	0.8 M	17.9 M

Figure 33: Number of Parameters of different U-Net backbones[59]

Transfer learning is a machine learning technique that uses other similar models' pre-trained knowledge to speed up training and achieve better results. In U-Net, transfer learning is very common and popular. The encoder layer is replaced by another pre-trained Convolutional Neural Network, and it is called the backbone. The popular backbones include VGG, ResNet, DenseNet, and Inception. However, the backbones significantly increase the size and number of parameters of U-Net. Figure 33 shows the number of parameters of popular backbones can go up to 100 million. A large number of parameters will slow down the training and inference speed and increase computational power. Edge devices have limited memory and processing power. Therefore, model that can run on edge devices is also restricted by the number of parameters. Models with a low number of parameters can run efficiently within the capability of the edge device.

Other than that, transfer learning is not applied in my U-Net because it is hard to modify the pooling layers in the pre-trained backbone. When the backbone is used in U-Net, the convolution and pooling layers of the U-Net are replaced by the pre-trained backbone. It is hard to modify the pooling layers because the modification made might not be compatible with the pre-trained weights and cause the model to perform poorly.

3.3 Modification on Dataset Augmentation

In the original U-Net paper, the authors use elastic deformations to their training data to help the model learn invariance in the dataset to improve the model. **In my 2D and 3D Datasets**, I combine five types of augmentation and apply them to my datasets. The augmentation technique includes vertical and horizontal flip, random rotate, transpose and grid distortion. Lewy, Dominik et al. state that the combination and mixing of augmentation techniques can improve the performance and robustness of the model. [60]

The horizontal and vertical flip technique flips the image horizontally and vertically. The random rotate 90 technique randomly rotates the image by 90 degrees, clockwise or counterclockwise. Transpose is a technique that switches the rows and columns of the pixels. Grid Distortion is a technique that applies distortion to the grid of the image to make the shape of the objects in the image slightly different.

Data augmentation generally reduces overfitting and improves the model's generalization ability as the model can train and validate with more versions of data. Besides, data augmentation also resolves the scarcity of medical images without increasing any operational

costs because segmenting medical images requires a professional radiologist, which is costly. [61]

3.4 3D U-Net – (BRATS)

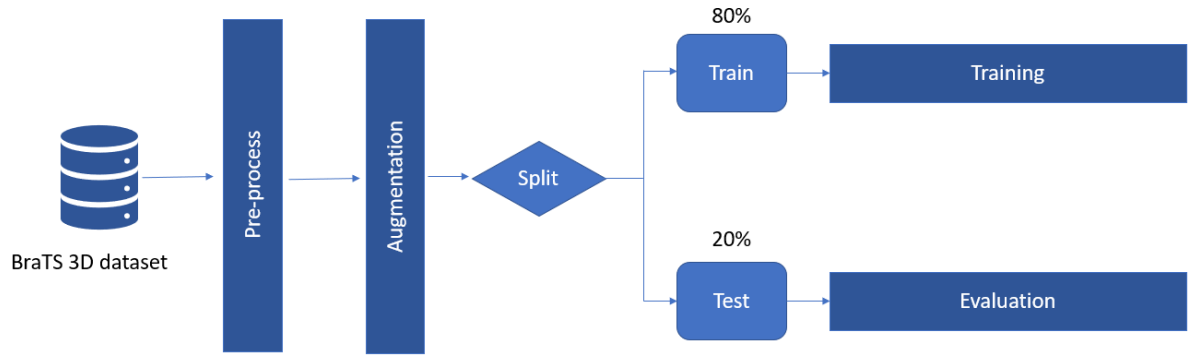


Figure 34: Flowchart of overall steps for 3D U-Net trained with BRATS

For section 3.4, 3D U-Net will be trained with BRATS 3D dataset. The BRATS dataset is multiclass, meaning it has more than one class. The four classes of BRATS include the GD–enhancing tumor (ET), the peritumoral edoema (ED), and the necrotic and non–enhancing tumor core (NCR). They represent different types of brain tumors. Based on Figure 34, the BRATS 3D dataset will be pre-processed and augmented. Then, it will be split into an 80:20 ratio for training and evaluation.

3.4.1 BRATS dataset

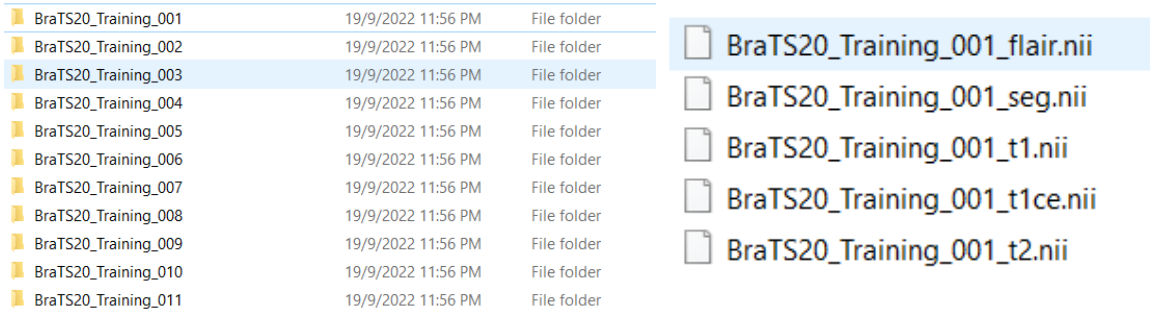


Figure 35: BRATS folder and files

In the BRATS dataset, there are 369 3D Brain Images, each image is obtained in 4 types of MRI Scans, and a segmented mask is provided. When the data was downloaded from Kaggle, it consisted of 369 folders; each folder represents a 3D image. Each folder consists of 4 types of MRI scans: Flair, T1, T1ce, and T2. It also includes a segmented mask called seg. All 3D images are in the NifTi file format. The MRI scans shapes are 255x255x155, and the mask shape is 255x255x155x4.

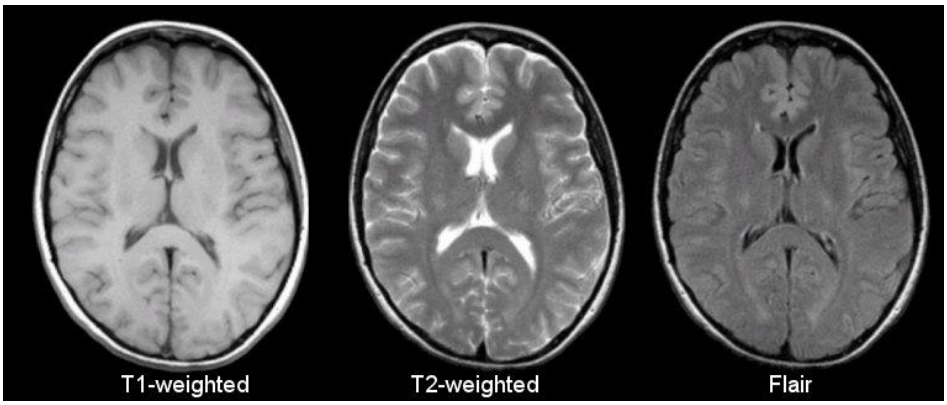


Figure 36: Comparison of T1, T2 and Flair MRI scans [12]

Based on Figure 36, Flair and T2 are darker than T1, making the tumor brighter and more obvious. It means that T1 has less context and will not be used for training.

3.4.2 BRATS pre-processing steps.

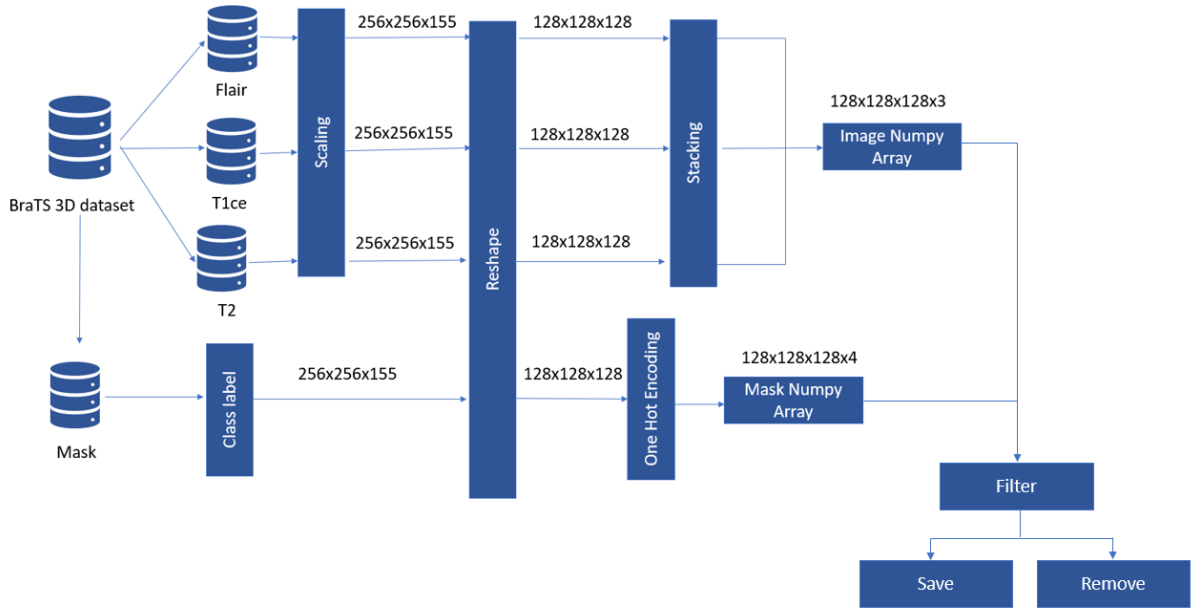


Figure 37: Flowchart of BRATS pre-processing steps.

Based on Figure 37, the BRATS 3D dataset image comprises three types of MRI scans: Flair, T1ce, and T2. The images are scaled, reshaped, and stacked into a numpy array. The class label of the BRATS 3D dataset mask is modified, then the mask is reshaped and converted into one hot encoding. Lastly, the images and masks are filtered before using it for training and evaluation.

3.4.2.1 Scaling

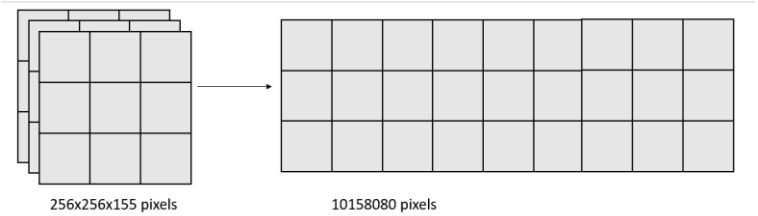


Figure 38: 3D image flattened to 1D image

First, all the 3D images (4 types of MRI scans) were loaded as a numpy array and flattened to 1 dimension.

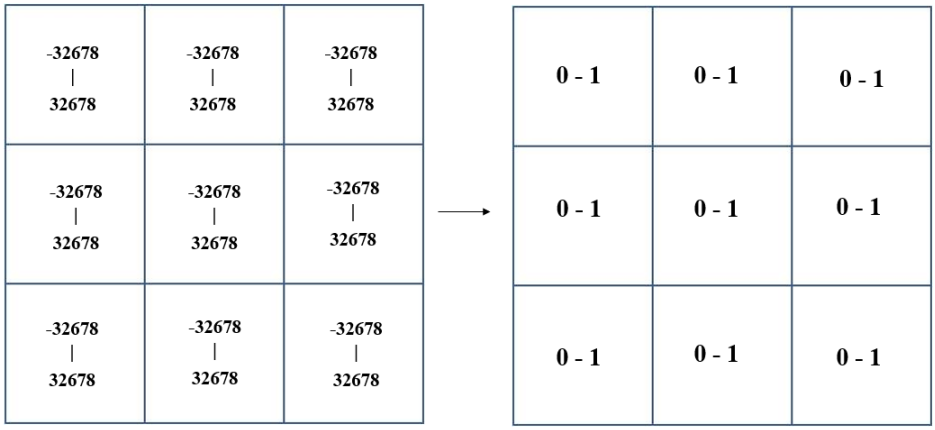


Figure 39: Pixels Scaling

The flattened numpy array is scaled from (-32678 to 32678) pixels to (0 to 1) pixels using MinMaxScaler. For BRATS 3D dataset, the pixels are in a 16-bit integer which ranges from -32678 to 32678, so it is necessary to normalize it before training. Each pixel can have the same scale or magnitude by scaling the pixel values to a fixed range, 0 to 1. It can speed up the convergence of the machine learning algorithm during training, notably for algorithms that employ distance measures like KNN and SVM. [62] Disparities in scales across the pixel values will make the model harder to learn. A model might learn large weight values because of having large input values, such as a spread of hundreds or thousands of unit pixels. Large weight values indicate that the model is unstable, so it may perform poorly during training and be sensitive to input values, leading to a more significant generalisation error. [63]

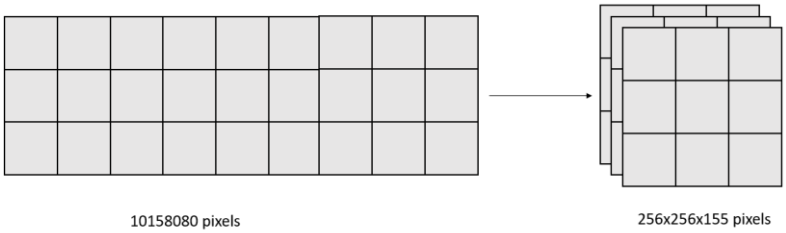


Figure 40: 1D image convert to 3D image.

After scaling, the scaled 1-dimensional numpy arrays are converted back to 3-dimensional.

3.4.2.2 Class label

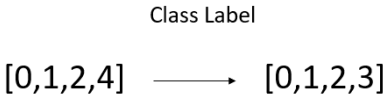


Figure 41: Class label Modification

The 3D masks are loaded as numpy array. Then, the mask arrays are changed from type float64 to uint8. Based on Figure 41, label 4 in the class label of the mask is changed to 3.

3.4.2.3 Reshape

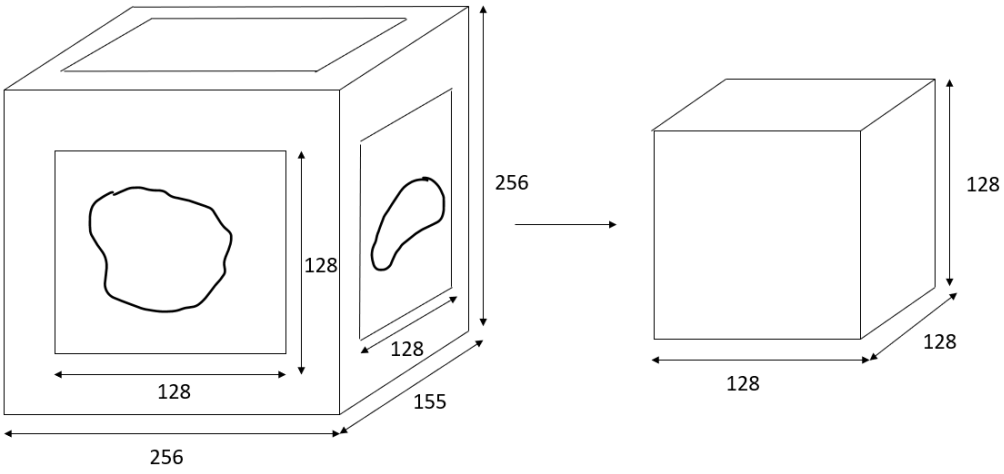


Figure 42: Reshape of 3D image and mask.

The middle part of 3 MRI images (Flair, T1ce, and T2) and mask are cropped together to remove redundant backgrounds or borders, which will affect the result of training. The shape of the image and mask is originally 256x256x155 and is reshaped into 128x128x128.

3.4.2.4 Stacking

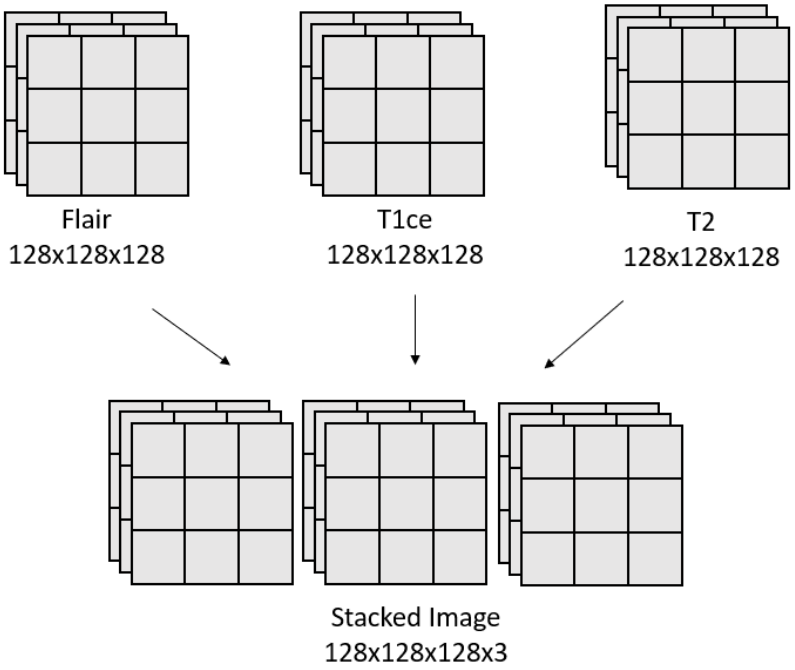


Figure 43: 3 3D images stacked into 4D numpy array.

3 3D MRI images (Flair, T1ce, and T2) numpy array are stacked into a 4-Dimension numpy array and then saved into a numpy file.

3.4.2.3 One Hot Encoding

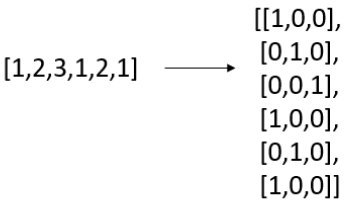


Figure 44: One Hot Encoding

The class integer vector mask is converted to a one-hot encoded matrix using the "to_categorical" function. Categorical variables are frequently used in machine learning as one-hot encoded vectors. A one-hot encoded vector is a binary vector of 0s and 1s, where the index of the 1 corresponds to the categorical class being denoted. Based on Figure 44, there is a vector with a class label of [1,2,3], with three unique classes. If the vector is converted to a one-hot encoded matrix, it will be [[1,0,0], [0,1,0], [0,0,1]]. After converting the mask to one-hot encoded vectors, it is saved into numpy files.

3.4.2.4 Filter

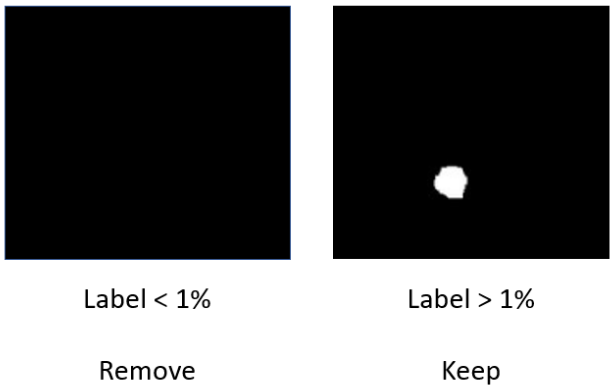


Figure 45: Filtering useless image and mask

All masks are checked to determine whether they have 1% of pixels with labels. It is so that the mask with no context will not be used to train, which is a waste of resources. “np.unique” function was used to obtain the four unique values (4 classes) from the mask and their corresponding counts. The proportion of the count of the first unique value (the background) over the sum of counts of all the unique values is used to check whether the background occupies more than 99% of the image. If the mask doesn’t have more than 1% label, the mask associated with the image is removed. All images and masks are split into 80% training and 20% testing/validation. After filtering, only 344 3D images and masks are left for training and testing.

3.4.3 BRATS Augmentation Steps

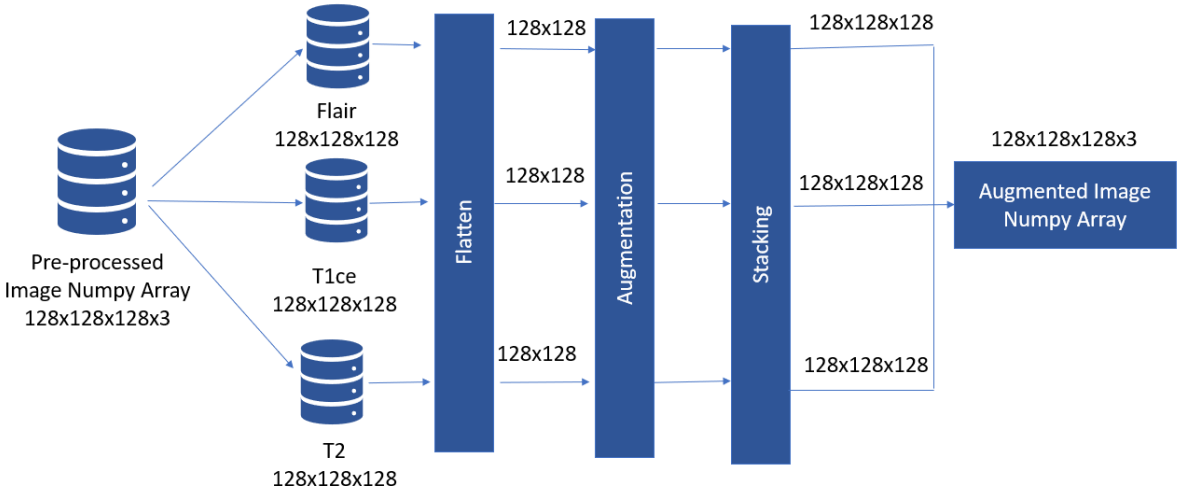


Figure 46: Flow chart of Image Augmentation

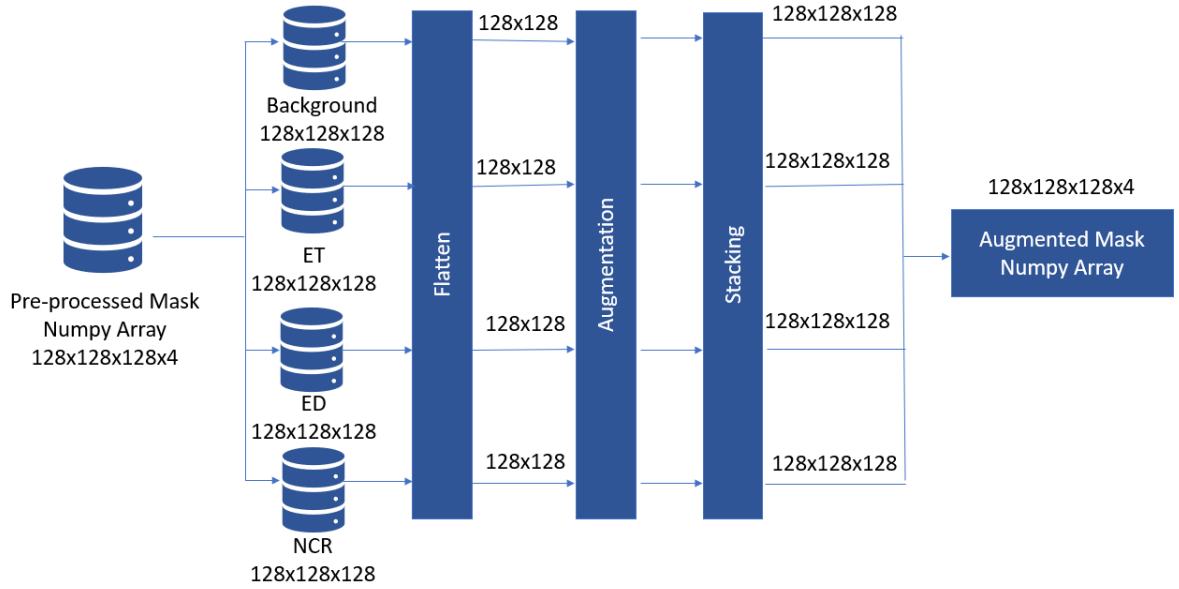


Figure 47: Flowchart of Mask Augmentation

Data Augmentation for the BRATS dataset is more challenging than other datasets because BRATs datasets are in 3D, not 2D. First, the pre-processed 4D image numpy arrays are split into respective MRI scans (Flair, T1ce and T2) and flattened to 2D and saved as a 2D image (PNG format). Then, the 2D images are augmented and converted back to a 2D numpy arrays. The 2D image numpy array is stacked and converted back to a 4D numpy array. Likewise for the mask, the pre-processed 4D mask numpy arrays are split into respective classes (Background, ET, ED, NCR) and flattened to 2D and saved as a 2D image (PNG format). Then, the 2D masks are augmented and converted back to a 2D numpy arrays. The 2D image numpy arrays are stacked and converted back to a 4D numpy array.

3.4.3.1 Flatten

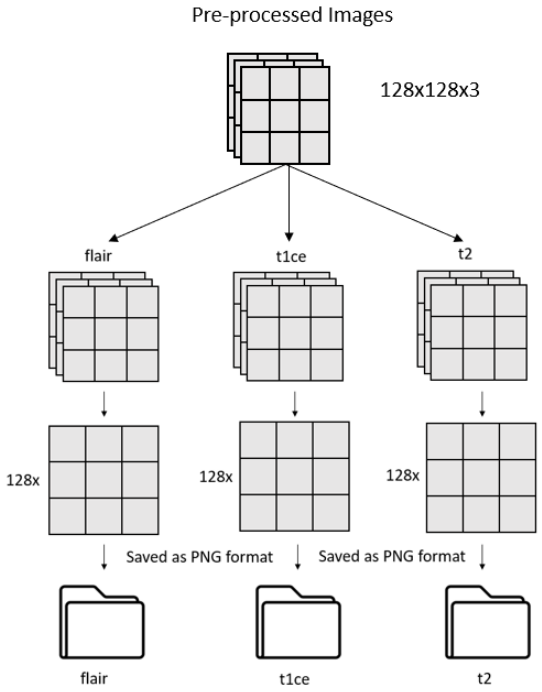


Figure 48: Flatten Pre-processed Image and save as PNG

The pre-processed images from 3 different MRI scans (Flair, T1ce, T2) with the shape of 128x128x128 are loaded. They are saved as a 2D 128x128 image in the respective folders in PNG format. Eventually, 128 2D images with the shape of 128x128 will be saved for each type of MRI scan.

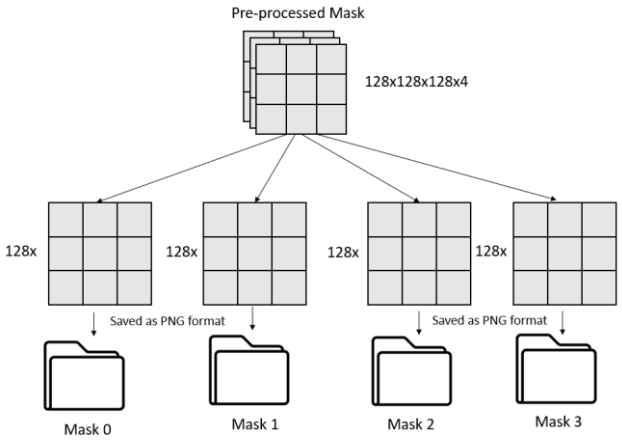


Figure 49: Flatten Pre-processed Mask and save as PNG

The pre-processed masks with the shape of 128x128x128x4 are loaded. They are saved as a 2D 128x128 image in 4 folders in PNG format. Eventually, 128 2D images with the shape of 128x128 will be saved for four types of masks. There are four types of masks because there are four classes.

3.4.3.2 Augmentation

The saved images and mask are loaded for augmentation. The augmentation uses the package Albumentation [64] which can perform spatial-level transform. Spatial-level transformations will augment an input image, as well as extra targets like masks, concurrently. The transform includes Vertical and Horizontal Flip, RandomRotate90, Transpose, and Grid Distortion. The number of images increased from 344 to 713.

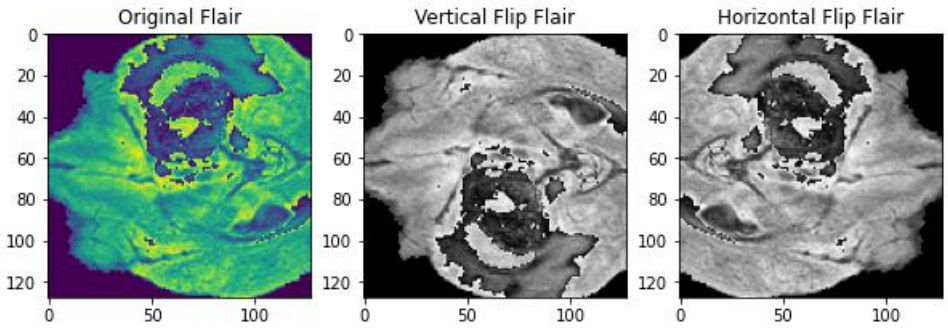


Figure 50: Vertical and Horizontal Flip Flair

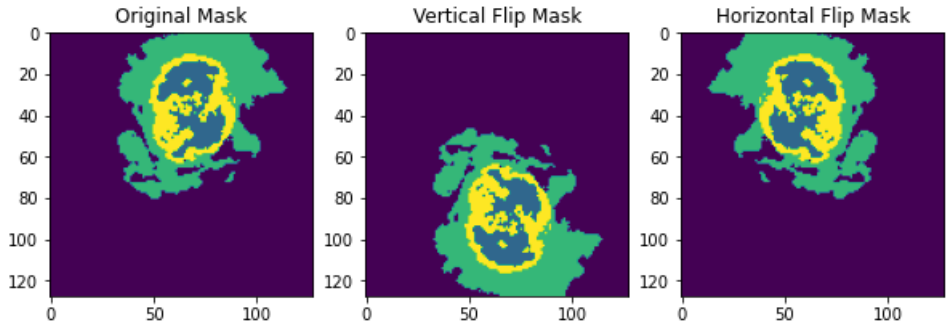


Figure 51: Vertical and Horizontal Flip Mask

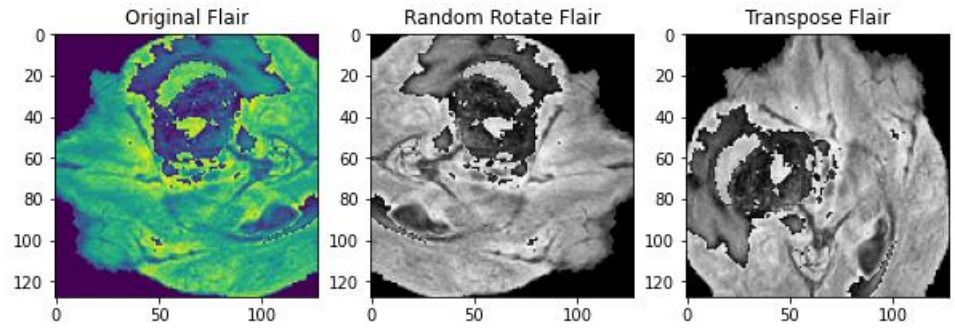


Figure 52: Random Rotate and Transpose Flair

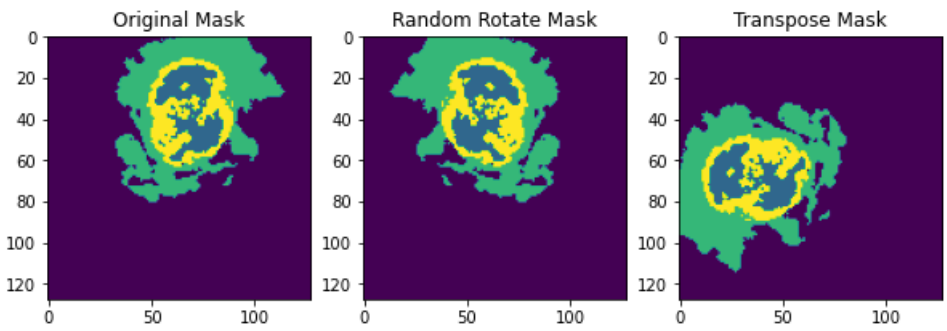


Figure 53: Random rotate and transpose mask

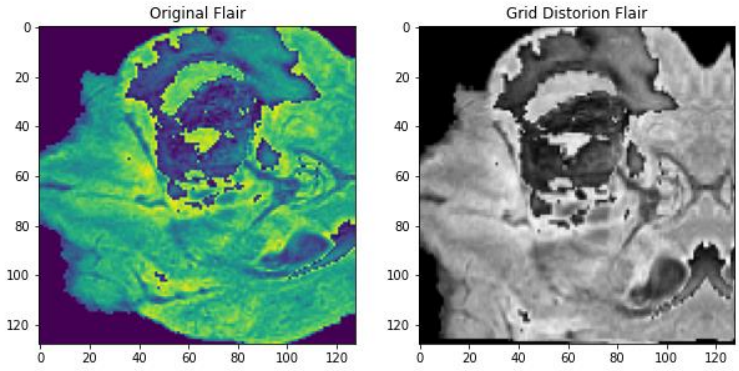


Figure 54: Grid Distortion Flair

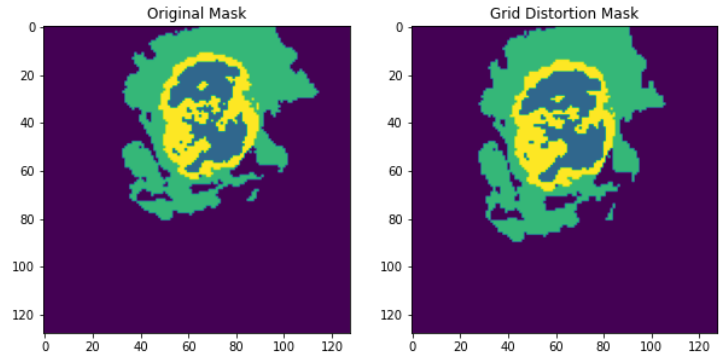


Figure 55: Grid Distortion Mask

3.4.3.3 Stacking

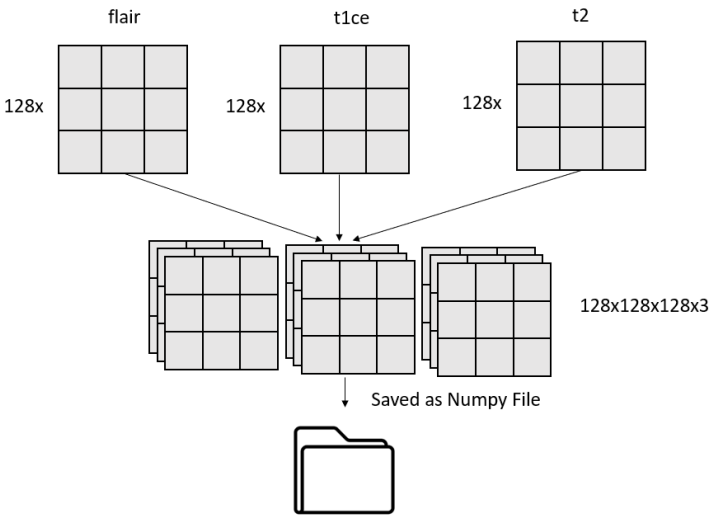


Figure 56: Stacking Augmented Images and save as numpy

The augmented images are stacked and saved as numpy files.

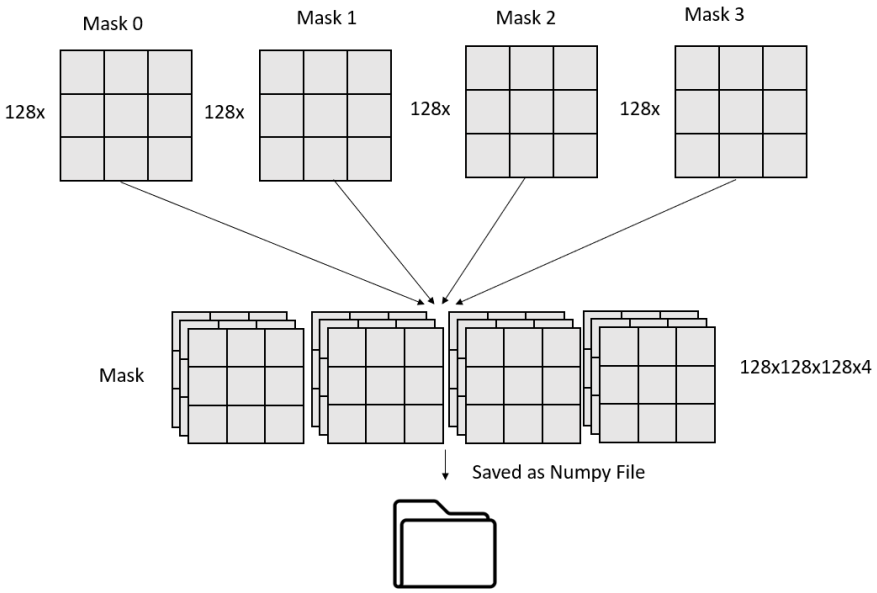


Figure 57: Stacking Augmented mask and save as numpy

The four different types of augmented masks are stacked and saved as one numpy file.

3.4.4 BRATS 3D U-Net Architecture

Table 2: BRATS 3D U-Nets

	3D U-Net 1	3D U-Net 2	3D U-Net 3
Number of Filters (At the start)	16	8	16
Types of Pooling Layer	Max Pooling	Average Pooling	Average Pooling

There are three types of U-Net with the same architecture, but different numbers of filters and types of pooling layers trained for this dataset. The purpose of testing the average pooling with a different number of filters is to identify the characteristics of the average pooling with different number of filters.

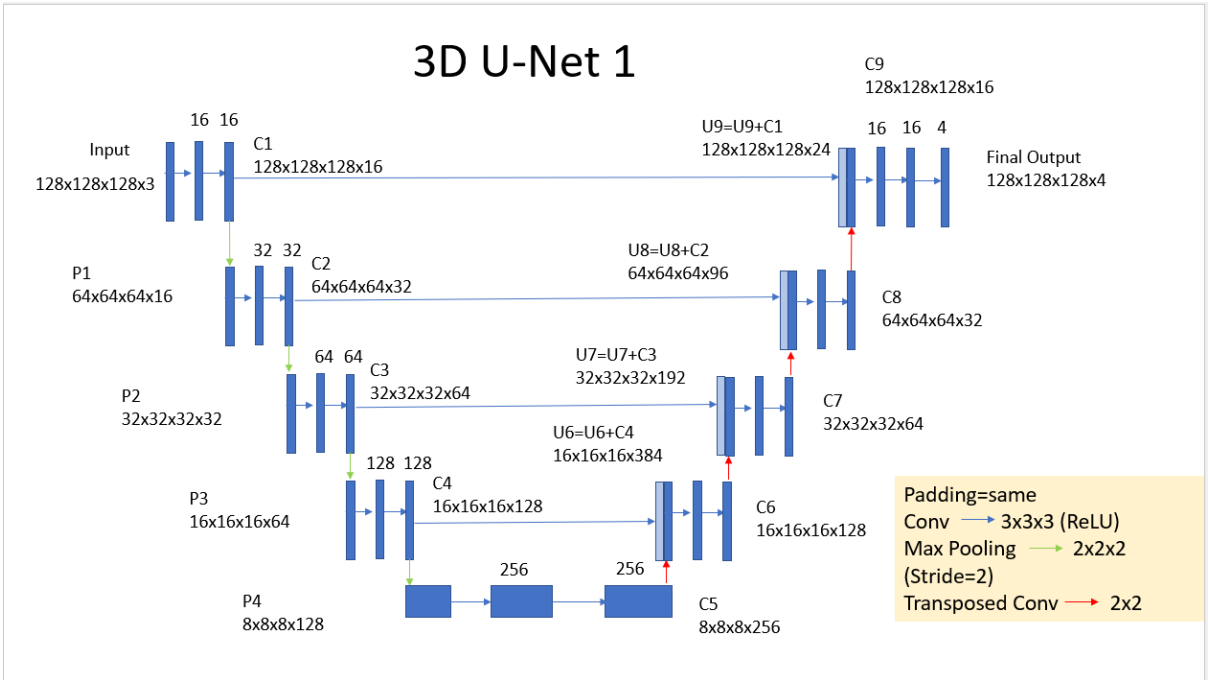


Figure 58: Architecture of U-Net 1

For my 3D U-Net 1, Convolutional layers are the first layer in the encoder network, proceeded by Max-pooling layers. The size of the convolutional kernel is 3x3x3 (for 3D datasets). 16 filters are applied in the encoder's initial layer, and the number is consistently doubled in the encoder's successive levels until it reaches 256 filters. The output of the encoder block is down-sampled by a factor of 2 utilizing max-pooling with a stride value of 2. It results in a two-fold reduction in the image's dimension.

The decoder uses transposed convolution to upsample the encoder's output. The matching feature maps from the encoder network are concatenated with the output produced by transposed convolution layer. Two subsequent convolution layers come after it, and the process is continued until the decoder network is complete. The feature maps continue to reduce with increasing resolution as the image progress through the decoder layers until it reaches the full-size image.

Initially, the shape of the input numpy array is 128x128x128x3, 128x128x128 is the height, width, and depth of the 3D image, and 3 means the three stacked MRI scans. The input numpy array is passed into 16 filters twice, which outputs a feature map with a 128x128x128x16 shape. It means that 16 feature maps with the size of 128x128x128 are produced. The feature maps are passed into pooling layers, and the size of the feature map is reduced from 128x128x128 to 64x64x64 while the number of feature maps remains the same. The resized feature map is passed into filters again, but the number of filters is doubled this time. It means that the number of feature maps is doubled. Then, it is passed into the pooling layer again means that the height and width are divided by 2, but the number of feature maps remains the same. These steps repeat until the number of feature maps reaches 256 and the size of the feature map reaches 8x8x8.

256 feature maps with the size of 8x8x8 are passed into the transposed convolution layer, and the size of the feature map is increased to 16x16x16, and the number of feature maps remains at 256. The 256 upsized feature map in layer U6 is concatenated with the convolutional layer C4 which has 128 feature maps with a size of 64x64. It means that the number of feature maps is increased to 384. 384 feature map is passed into 128 filters, and 128 feature maps with the size of 16x16x16 are obtained. The size of feature maps is doubled and concatenated with the corresponding encoder layer again. Then the feature maps are passed into filters with half the number of previous layers. These steps repeat until the number of feature maps reaches 16 and the size of the feature map reaches 128x128x128, which is the original size of the image. The final 16 feature maps are finally passed into the convolution layer with four filters to output the mask with four classes.

Table 3: Dropout Rate in each layer

Layers	Dropout Rate
C1 and C2	0.1
C3 and C4	0.2
C5	0.3
C6 and C7	0.2
C8 and C9	0.1

The dropout function with different dropout rates was applied in all the convolutional layers, as the Table 3. ReLU (Rectified Linear Unit) was used as an activation function to overcome

the vanishing gradient problem and let the model perform better and learn faster. The softmax activation function was used for multiclass segmentation for the final output layer.

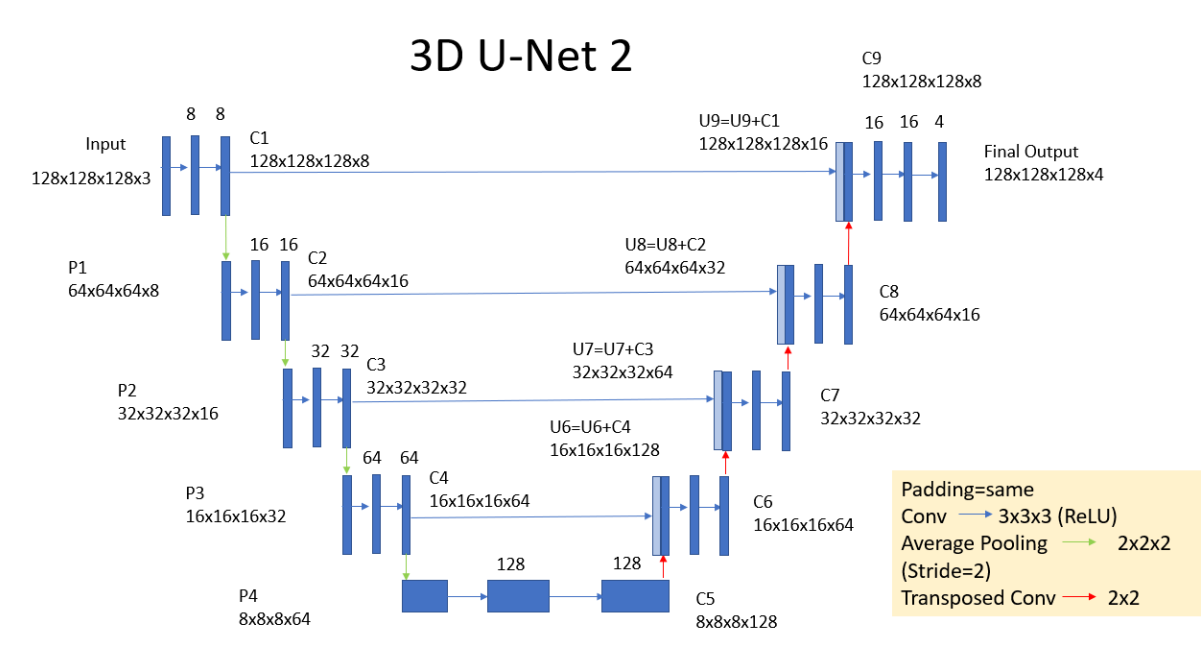


Figure 59: Architecture of U-Net 2

3D U-Net 2 uses the same structure and dropout rate. The only difference is the pooling layer and the number of filters. The pooling layer is changed to Average Pooling. The starting number of feature maps is reduced to 8, and the consequent number of filters is changed accordingly.

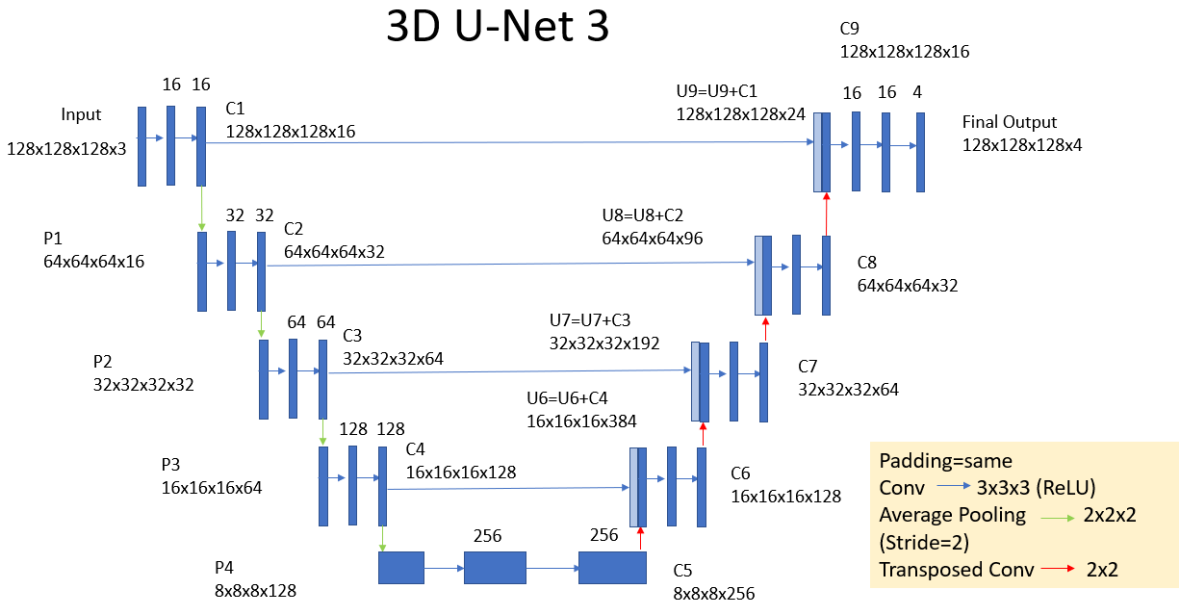


Figure 60: Architecture of U-Net 3

3D U-Net 3 uses the same structure and dropout rate. The only difference is the pooling layer is changed to Average Pooling.

3.4.5 BRATS Model Training Steps

First, the training images/mask and validation images/masks are loaded from respective folders. Four lists are created to save training images/masks and validation images/masks.

```
Training Image List = saves all the training images from training image folder
Training Mask List = saves all the training masks from training mask folder

Batch size =2
L= length of Training Image List
Start an infinite loop:
    Batch start = 0.
    Batch end = batch size.

    While batch start is less than L:
        Limit = Batch end or L (depends on which value is smaller)
        X= load the image from Training Image List from Batch start to Limit.
        Y= load the mask from Training Mask List from Batch start to Limit.

        Returns X and Y (two numpy arrays with batch size of 2)
        Batch start increase by batch size.
        Batch end increase by batch size.
```

Figure 61: Pseudocode for custom data generator

Then, a custom data generator was built to provide training images and masks to the model when training. Based on Figure 61, the training image and mask lists are used to save all the training images and mask filenames from the respective folder. A batch size of 2 is set so the custom data generator can provide two images and masks per loop. An infinite loop starts, the batch start is 0, and the batch end is assigned as batch size. While the batch start is less than the length of the image list, the image and mask from the image list and mask list are loaded from batch start to limit. The limit is either the batch end or the length of the training image list, depending on which is smaller. The two images and masks are returned, and the batch start and batch end increase by the batch size. The reason to set a limit when loading the image and mask is to prevent the image and mask returned from exceeding the total number of training images and masks.

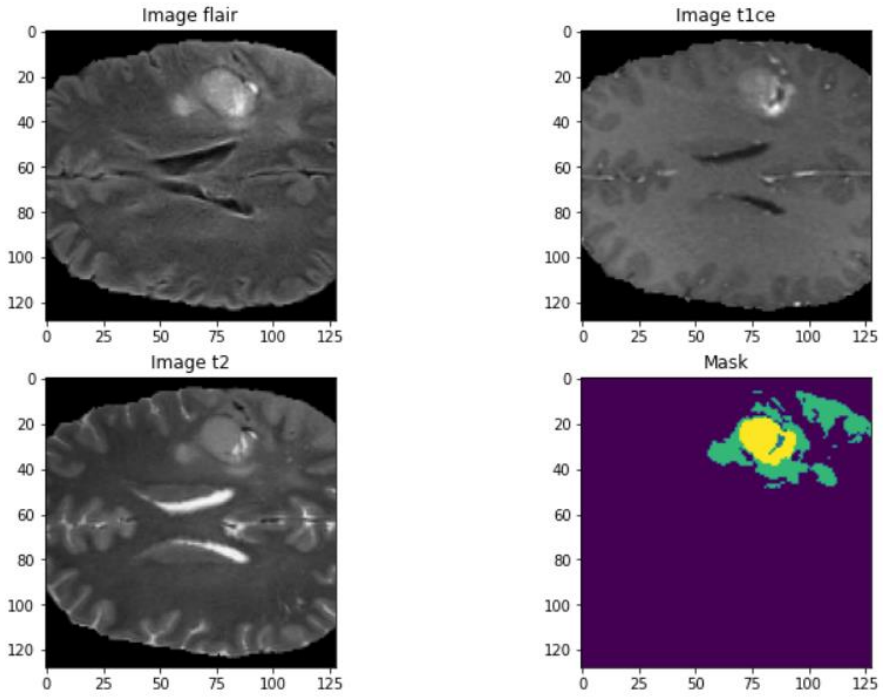


Figure 62: Flair, T1ce, T2 and Mask

A random 3D image and mask are plotted to make sure they are compatible.

Table 4: Training Parameters for 3D U-Net

Batch Size	2
Loss Function	Dice Loss + Focal Loss
Learning Rate	0.0001
Metrics	MeanIOU
Optimizer	Adam

U-Net 1 is compiled with loss function, learning rate, metrics, and optimizer, as shown in Table 4. The steps per epoch are determined by the total length of the training image list divided by batch size. The validation steps per epoch are determined by the total length of the validation image list divided by batch size. U-Net 1 is trained for 10 epochs based on the steps per epoch and validation steps per epoch. The trained model weight is saved in HDF5 file format. The training and validation accuracy and loss plot is also saved for evaluation. U-Net 1 is trained again with 10 epochs until the IOU score converges. The steps above are repeated for U-Net 2 and U-Net 3.

3.4.6 BRATs – Model Evaluation

The mean IOU score of the trained model is evaluated using the validation image and mask.

3.5 2D U-Net – (Chest X-Ray)

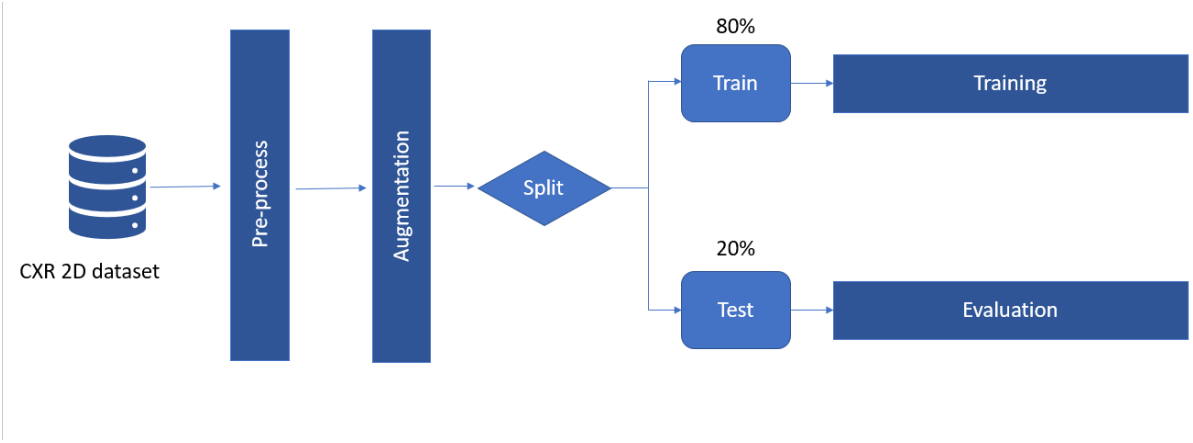


Figure 63: Overall process for 2D U-Net and Chest X-Ray

For section 3.5, 2D U-Net will be trained with Chest X-Ray 2D dataset. The Chest X-Ray dataset is a binary dataset, meaning it has only one class, lung. Based on Figure 63, the Chest X-Ray 2D dataset will be pre-processed and augmented. Then, it will be split into an 80:20 ratio for training and evaluation.

3.5.1 Chest X-Ray Dataset

This Chest X-Ray Dataset comprises two datasets, Montgomery County Chest X-Ray dataset, and ShenZhen Chest X-Ray Dataset.

The Department of Health and Human Services, Montgomery County, Maryland, USA, and the National Library of Medicine collaborated to create this Montgomery County Chest X-Ray Dataset. This dataset consists of 80 normal cases of X-Ray and 58 cases of tuberculosis X-Ray.

The National Library of Medicine, Maryland, USA, and Shenzhen No. 3 People's Hospital, Guangdong Medical College, Shenzhen, China, created the ShenZhen Chest X-Ray Dataset. This dataset consists of 336 normal cases of X-Ray and 326 cases of tuberculosis X-Ray.

In total, there are 800 X-Ray images in this dataset.

3.5.2 Chest X-Ray Pre-processing

94 Redundant X-Ray Images without a mask are removed. 706 images and masks are available for augmentation and training.

3.5.3 Chest X-Ray Augmentation

The augmentation uses Albumentation package [64] to perform spatial-level transform. Spatial-level transformations will augment an input image, as well as extra targets like masks, concurrently. The transform includes Vertical and Horizontal Flip, RandomRotate90, Transpose, and Grid Distortion. The number of images and masks increased from 704 to 1144.

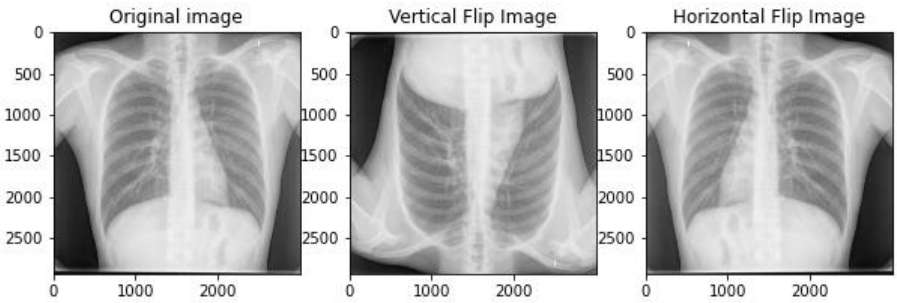


Figure 64: Vertical and Horizontal Flip Image

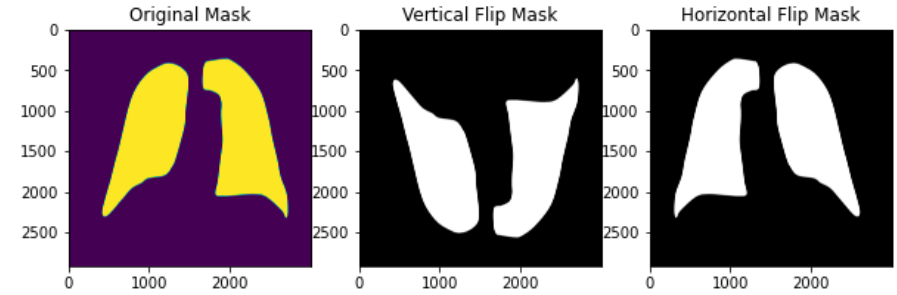


Figure 65: Vertical and Horizontal Flip Mask

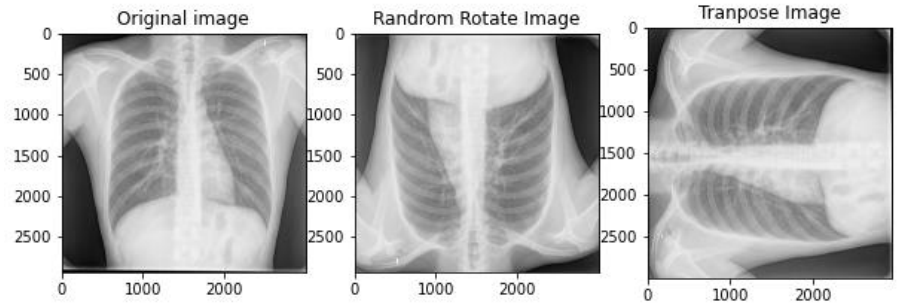


Figure 66: Random Rotate and Transpose Image

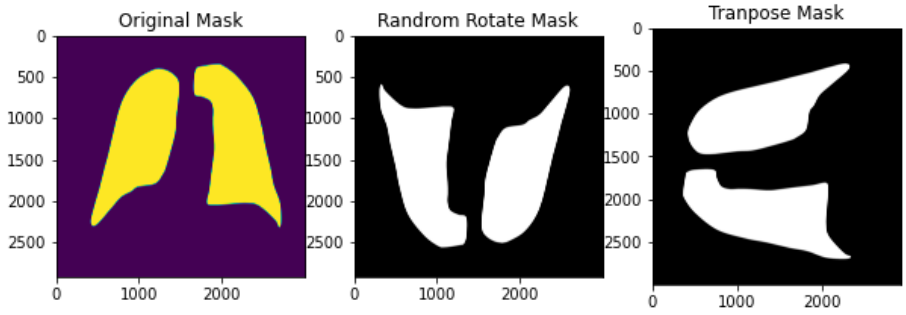


Figure 67: Random Rotate and Transpose Mask

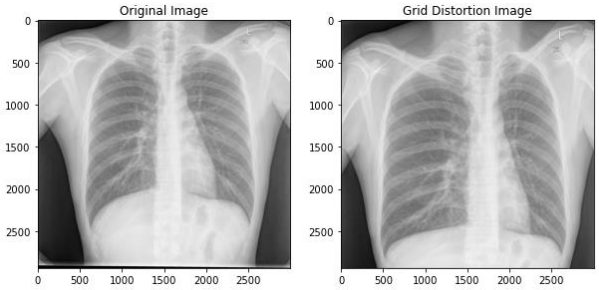


Figure 68: Grid Distortion Image

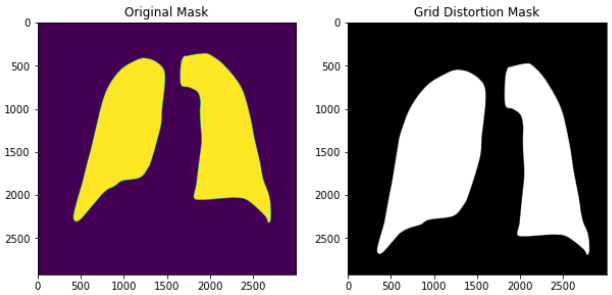


Figure 69: Grid Distortion Mask

3.5.4 Chest X-Ray 2D U-Net Architecture

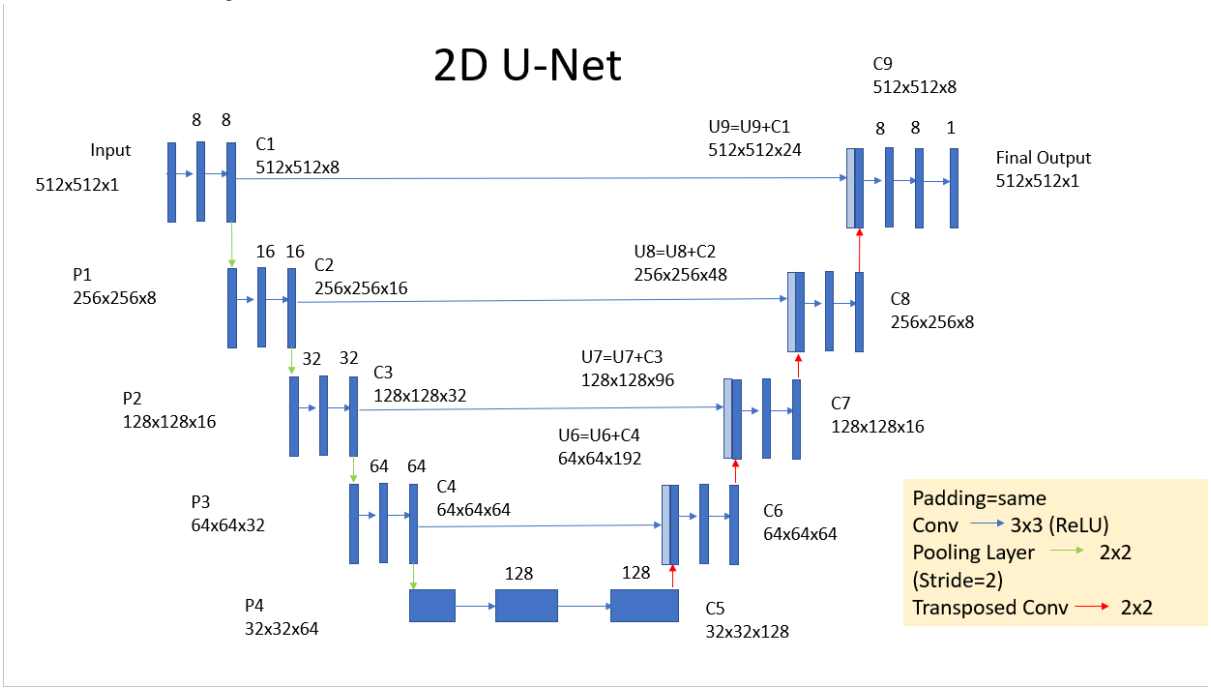


Figure 70: 2D U-Net Architecture

For my 2D U-Net, Convolutional layers are the first layer in the encoder network, proceeded by pooling layers. Four types of pooling layers are trained using the same 2D U-Net architecture. The pooling layers include Max Pooling, Average Pooling, Hybrid Pooling, and Atrous Spatial Pyramid Pooling. The convolutional kernel size is 3x3 (for 2D datasets). 8 filters are applied in the encoder's initial layer, and the number is consistently doubled in the encoder's successive levels until it reaches 128 filters. Utilizing a pooling layer with a stride value of 2, the output of the encoder block is downsampled by a factor of 2. It results in a two-fold reduction in the image's dimension.

The decoder uses transposed convolution to upsample the encoder's output. The matching feature maps from the encoder network are concatenated with the output produced by transposed convolutional layer. Two subsequent convolution layers come after it, and the process is continued until the decoder network is complete. The feature maps continue to reduce with increasing resolution as the image progress through the decoder layers until it reaches the full-size image.

Initially, the shape of the input image is 512x512x1, 512x512 is the height and width of the image, and 1 represents that it is a binary image. The input image is passed into 8 filters twice, which outputs a feature map with a 512x512x8 shape. It means 8 feature maps with a height and width of 512x512 are produced. The feature maps are passed into pooling layers, and the size of the feature map is reduced from 512x512 to 256x256 while the number of feature maps remains the same. The resized feature map is passed into filters again, but the number of filters is doubled this time. It means that the number of feature maps is doubled. Then, it is passed into the pooling layer again means that the height and width are divided by 2, but the number of feature maps remains the same. These steps repeat until the number of feature maps reaches 128 and the feature map size reaches 32x32.

128 feature maps with the size of 32x32 are passed into the transposed convolution layer, and the size of the feature map is increased to 64x64, and the number of feature maps remains at 128. The 128 upsized feature map in layer U6 is concatenated with convolutional layer C4 which has 64 feature maps with the size of 64x64. It means that the number of feature maps is increased to 192. 192 feature map is passed into 64 filters, and 64 feature maps with the size of 64x64 are obtained. The size of feature maps is doubled and concatenated with the corresponding encoder layer again. Then the feature maps are passed into filters with half the number of previous layers. These steps repeat until the number of feature maps reaches 8 and the size of the feature map reaches 512x512, which is the original size of the image. The final eight feature map is finally passed into the convolution layer with one filter to output the binary mask.

ReLU (Rectified Linear Unit) was used as an activation function to overcome the vanishing gradient problem and let the model perform better and learn faster. For the final output layer, the sigmoid activation function was used for binary segmentation. Early stopping is added to the model to prevent overfitting.

3.5.5 Chest X-Ray-Training

All the X-Ray images and masks are loaded and appended to the image and mask respectively and resized from 1298x1298 to 512x512. All the X-Ray images and masks are split into 80% training, 10% validation, and 10% testing.

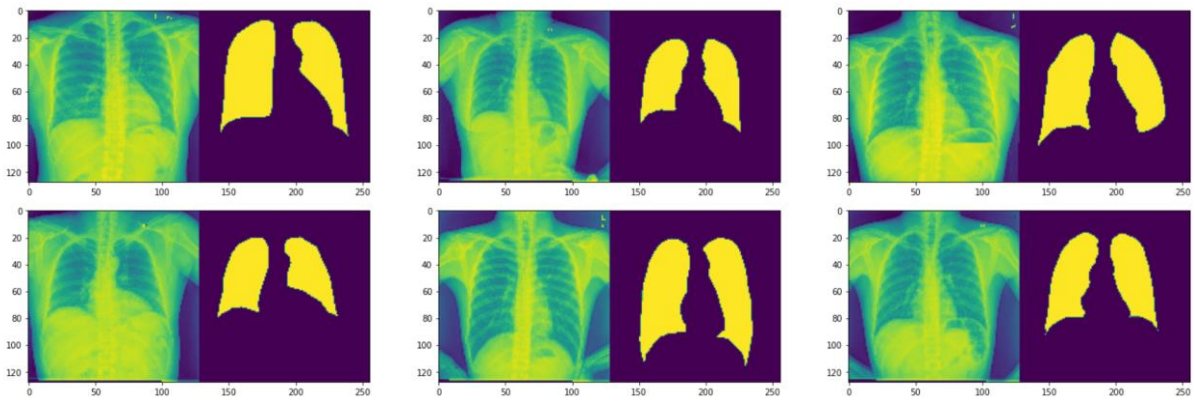


Figure 71: Image and Mask

Three random images and masks are plotted side by side to ensure they are compatible.

Table 5: Training parameters of 2D U-Net

Batch Size	16
Loss Function	Dice Coefficient Loss
Learning Rate	0.0002
Metrics	Dice Coefficient
Optimizer	Adam

The model is compiled with loss function, learning rate, metrics, and optimizer, as shown in Table 5. Three callback functions are used for this model, which are ModelCheckPoint,

EarlyStopping, and ReduceLRonPlateau. The Model Checkpoint callback is used to save the weights of the model with the best validation loss. The Early Stopping callback stops training if the validation loss does not improve after a certain number of epochs. The ReduceLRonPlateau callback reduces the learning rate if the validation loss does not improve after a certain number of epochs.

Four models (4 different pooling layers but the same architecture) are trained with 10 epochs and a batch size of 16 with the original dataset (without augmentation). The trained model weights are saved in HDF5 file format. The pooling layer with the best performance was trained again with the augmented dataset until the score converged.

3.5.6 Chest X-Ray-Model Evaluation

The model is evaluated with five metrics, Accuracy, Dice Score, IOU score, Recall score and Precision Score. The testing images are looped and input into the model to predict the mask. The predicted and actual masks are put into the metrics functions from sci-kit learn to get the score. All the score is appended to the score array, and then the mean of each score is calculated.

3.6 Edge Device for U-Net

There are a few popular edge devices in the market, including Intel NCS2 and Google Coral Dev Board Mini. For processing power wise, both of them can operate at 4 TOPS (trillion operations per second), and both have low power consumption. For size wise, NCS2 has a slightly smaller size (72.5x27x14mm) compared to Coral Dev Board Mini (88x60x24mm). For software support, NCS2 supports different models like Tensorflow, PyTorch, Caffet, and many more, but Google Coral Dev Board Mini can only support Tensorflow lite model.

In June 2022, Owais Ali and other IEEE members presented their work on implementing U-Net on Intel Neural Compute Stick 2 (NCS2) to do binary segmentation on medical datasets like BraTs datasets, ZNSDB datasets for tuberculosis, and Brain MRI datasets. They successfully reduced the 30 million parameter U-net to 0.5 million parameters, and their model has the highest dice score of 0.96 [1]. They show that NCS2 has a huge success in medical semantic segmentation. However, Google Coral Dev Board Mini has yet to be explored by any researchers on how good it is at medical segmentation, so it is a good opportunity to explore it.

After training and comparing 2D U-Net, the best model is selected and converted into a Tensorflow Lite version to be applied on Google Coral Dev Board Mini.

3.6.1 Chest X-Ray Model to TFLite version

The trained model for my Chest X-Ray dataset is saved in HDF5 format and is converted to Tensorflow Lite (TFlite) file format using a TFlite converter. The reason for using TFlite models is that the Google Coral Dev Board Mini only can run inference on TFlite models.

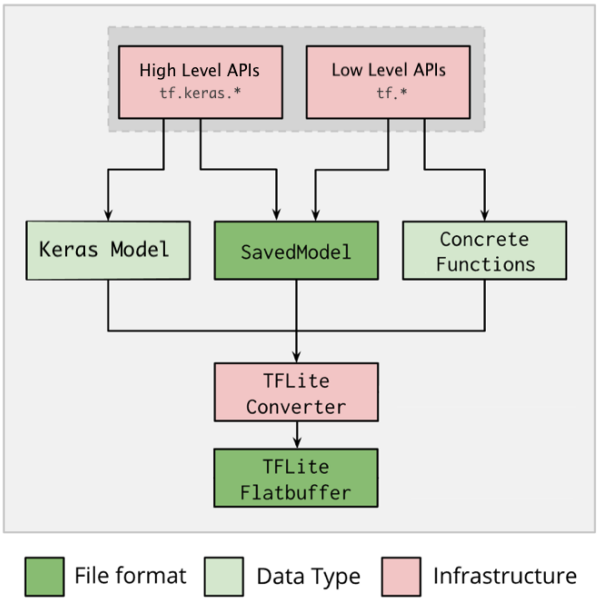


Figure 72: Workflow of Tensorflow Lite Converison [65]

Based on Figure 72, it shows the workflow of converting the model. First, the model is saved using high-level or low-level APIs into the Keras model, saved model, and concrete functions. They are converted into a TensorFlow Lite model (an optimized FlatBuffer format identified by the. tflite file extension) using a TFlite converter.

FlatBuffer is an efficient cross-platform serialization library for C++, C#, and many more languages. FlatBuffer has most of the benefits of a protocol buffer (used to save Tensorflow models) without deserializing the whole data file. Besides, Flatbuffer is memory efficient and fast because it does not need additional allocations for memory to access data.

3.6.2 Coral Dev Board Mini Inference

Table 6: Requirements for Coral Dev Board Mini Inference [66]

Host Computer Operating System	Windows
Python3 version	3.9
Power Supply	2A/5V Power Adapter
Wifi connection	SNS wifi
Connection Hardware	USB-C cables

All requirements listed in Table 6 are prepared. The Mendel development tool is installed onto the host computer. MDT (Mendel Development Tool) is a tool that enables interface with the Dev Board Mini. MDT can identify connected devices, download Linux packages on the board, and start a shell terminal.

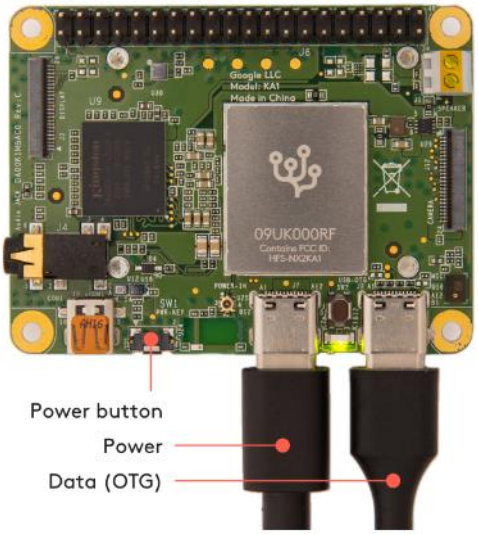


Figure 73: Power Supply and Data Port [66]

The power supply cable is connected to the left plug of the board with a power source of 5V/2A. The data cable is connected to the right plug of the board with the host computer. Once these two cables are connected, the board will automatically boot up within 20 seconds, and the LED will turn green.

```
C:\Users\ADMIN>mdt devices
xenial-umpire          (172.17.2.102)
```

Figure 74: Checking connected device.

Devices connected to the host computer are checked using the "mdt devices" command line. Based on Figure 74, the name of the board and its IP are shown.

```
C:\Users\ADMIN>mdt shell
Waiting for a device...
Connecting to xenial-umpire at 172.17.2.102
```

Figure 75: Connecting to shell of Google Coral Dev Board MIni

The “mdt shell” command connects the board's shell to the host computer.

```
nmcli dev wifi connect <wifi name> password <wifi password> ifname wlan0

nmcli connection edit

set 802-1x.eap peap

set 802-1x.phase2-auth mschapv2

set 802-1x.identity <username>

set 802-1x.password <password>

set wifi-sec.key-mgmt wpa-eap

set 802-11-wireless.ssid <wifi name>
```

Figure 76: Wifi connection commands.

Wifi network is connected to the board using commands in Figure 76 to update the system and download packages and code. Wifi connections are checked using the command “nmcli connection show”.

```
sudo apt-get update

sudo apt-get dist-upgrade

sudo reboot now
```

Figure 77: Update Mendel Software command

The system’s image is updated using command in Figure 77 to make sure the system is up to date. The board is rebooted after updating the system.

```
mendel@xenial-umpire:~$ ls
coral  END_USER_LICENSE  Medical_Semantic_Segmentation
```

Figure 78: Check home directory

A repository is created in GitHub to upload the inference code, a folder with 70 test numpy files converted from a Chest X-Ray Image, and pre-trained model weights in TFlite format. The repository is cloned into the board. The' ls' command checks the files and folders in the current directory. A folder called "Medical_Semantic_Segmentation" is the repository cloned to the board.

```
mendel@xenial-umpire:~/Medical_Semantic_Segmentation$ ls
custom_coral_inference.py  input.npy  model.tflite
```

Figure 79: Check Cloned GitHub Repository

The folder has three files:

- An inference code.
- A folder of 70 numpy files converted from a chest X-Ray Image
- The model weights in the TFlite format

```
mendel@xenial-umpire:~/Medical_Semantic_Segmentation$ python3 custom_coral_inference.py
551.24 ms
```

Figure 80: Run Inference Code and Runtime was returned.

The inference code is run. The inference time and 70 result files in numpy file format are returned. The inference code takes in and outputs a numpy file because the CV2 package is not supported on the board. CV2 package converts images into a numpy array as an input of the model. So, the images are converted to the numpy files in the host computers before uploading to the board.

```
C:\Users\ADMIN>mdt pull /home/mendel/Medical_Semantic_Segmentation/result.npy C:\Users\ADMIN\Desktop
Waiting for a device...
Connecting to xenial-umpire at 192.168.101.2
100% |<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<| /home/mendel/Medical_Semant...
```

Figure 81: Transfer result back to host computer

70 result numpy files are transferred to the host computer using the “mdt pull” command.

3.6.3 Coral Dev Board Mini Result Evaluation

The TFlite model is also evaluated with five metrics, Accuracy, Dice Score, IOU score, Recall score, and Precision Score. There are two folders to store the numpy files for the real mask and predicted mask by the TFlite model. The numpy files in the folders are loaded and put into the function to evaluate the scores. The scores are appended to an empty array, and an average of the score is calculated.

3.7 DeepLabv3 Matlab - Chest X-Ray

The purpose of training DeepLabV3+ on the Chest X-Ray dataset is to compare the results of DeepLabv3+ with my modified U-Net. Many papers compare U-Net and Deeplabv3+ with different medical datasets, but the outcomes differ. Some papers said that U-Net outperformed DeepLabV3+, and others said that DeepLabv3+ outperforms U-Net due to different settings. Therefore, DeepLabV3+ was trained in my project to compare and analyse the results.

3.7.1 Chest X-Ray - DeepLabv3 Matlab Training

The images and mask are converted into binary form because the DeepLabv3+ model in Matlab only accepts images in binary format. The binary images and mask are split into training and testing in the ratio of 80:20.

The image and mask are loaded using imageDataStore and pixelLabelDatastore. ImageDataStore is a function in Matlab to manage and read a collection of image files from a directory. PixelLabelDataStore is a function in Matlab to manage and read a collection of pixel-label images from a directory for semantic segmentation. [67]

A DeepLab V3+ network is created with an input image size of 512x512, four classes, and ResNet18 as a backbone. The image and pixel label data (masks) are combined. Then, a pre-processing transform function is applied to the combined images and masks. For the pre-processing transform function, the image and mask are resized to the image size declared earlier (512x512). Then, the image is converted to RGB because ResNet18 requires RGB image input.

The training options of the DeepLab Network are specified. The optimizer used is SGDM (Stochastic Gradient Descent with Momentum). The normal gradient descent method modifies the network parameters (weights and biases) to minimise the loss function by making small steps toward the negative gradient of the loss function at each iteration. On the contrary, the stochastic gradient descent technique updates the model parameters using a small random subset of the training data, called a mini-batch, instead of the entire dataset. It makes the training process faster and requires less memory. However, the mini-batch updates are a noisy estimate of the updates resulting from using the entire dataset. The algorithm repeats this process over the entire dataset in multiple iterations, called epochs. The size of the mini-batch and the number of epochs can be adjusted for better performance. The stochastic gradient descent method can oscillate along the route leading to the best result. One method to decrease this oscillation is to include a momentum factor in the parameter update, which is called SGDM. [68]

The mini-batch size is set as 8, and the max epochs is set as 3. The model is trained and saved using the command “save net”.

3.7.2 Chest X-Ray – DeepLabv3 Matlab Evaluation

The test images are loaded and resized. The "semanticseg" function is used to segment the test images. A white image for the segmented mask is generated as a layover and saved into the destination folder.

Figure 82: Mask colour conversion

The predicted result is converted to black and white from blue and purple before evaluation. The DeepLabV3 model is evaluated with five metrics, Accuracy, Dice Score, IOU score, Recall score and Precision Score. There are two folders to store the test mask images and predicted mask images by the DeepLabv3 model. The images in the folders are loaded and put into the function to evaluate the scores. The scores are appended to an empty array, and an average of the score is calculated.

4.0 Results and Discussion

4.1 BRATS Results

4.1.1 Training Score of 3D U-Net Max Pooling with 16 filters

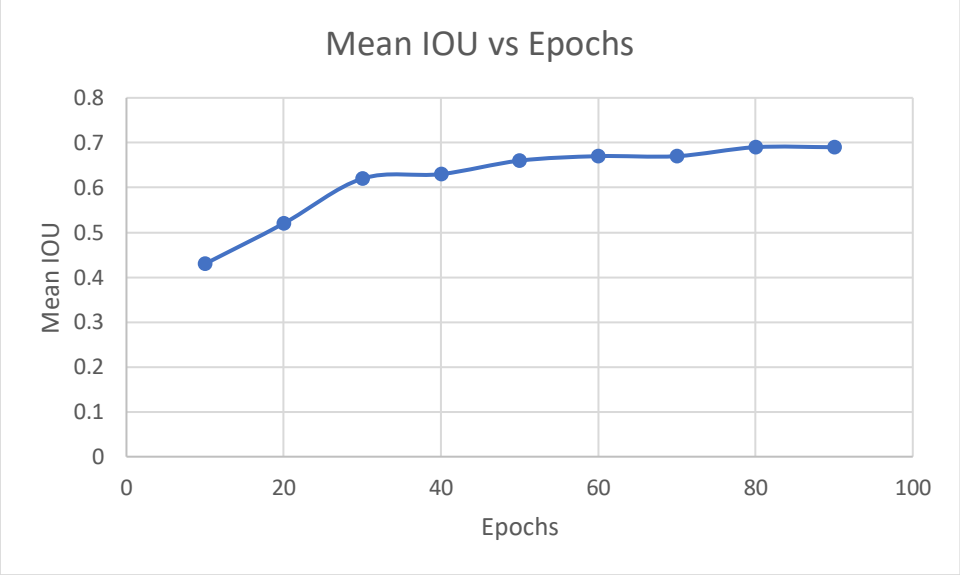


Figure 83: Mean IOU vs Epochs of 3D U-Net with Max Pooling and 16 filters

Based on Figure 83, the mean IOU score starts to converge at 60 epochs.

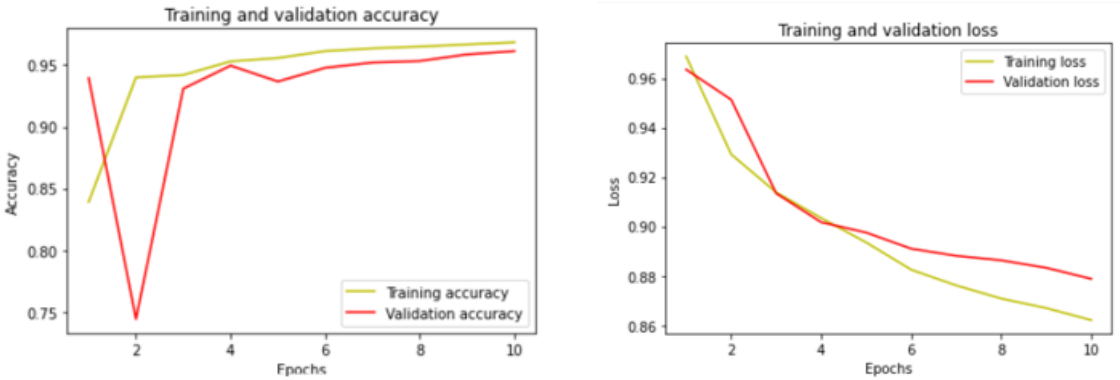


Figure 84: Training and Validation Accuracy and Training and Validation Loss at 10 epochs

Based on Figure 84, the training accuracy started at 0.85 and converged at 0.95. The validation accuracy drops from 0.95 to 0.75 and then converges back to 0.95. The training and validation losses started from around 0.96 and decreased to 0.86. It shows that the model is still learning and improving at 10 epochs.

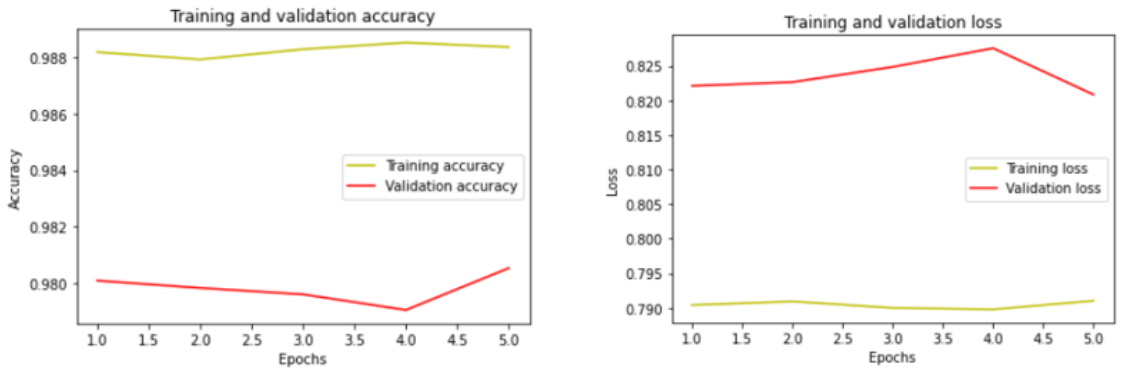


Figure 85: Training and Validation Accuracy and Training and Validation Loss at 70 epochs

Based on Figure 85, the training/validation accuracy and loss do not have obvious changes. It shows that the model is not learning and improving at 70 epochs. This observation matches Figure 82, that the IOU score converges at 60 epochs.

4.1.2 Training Score of 3D U-Net Average Pooling with 8 filters

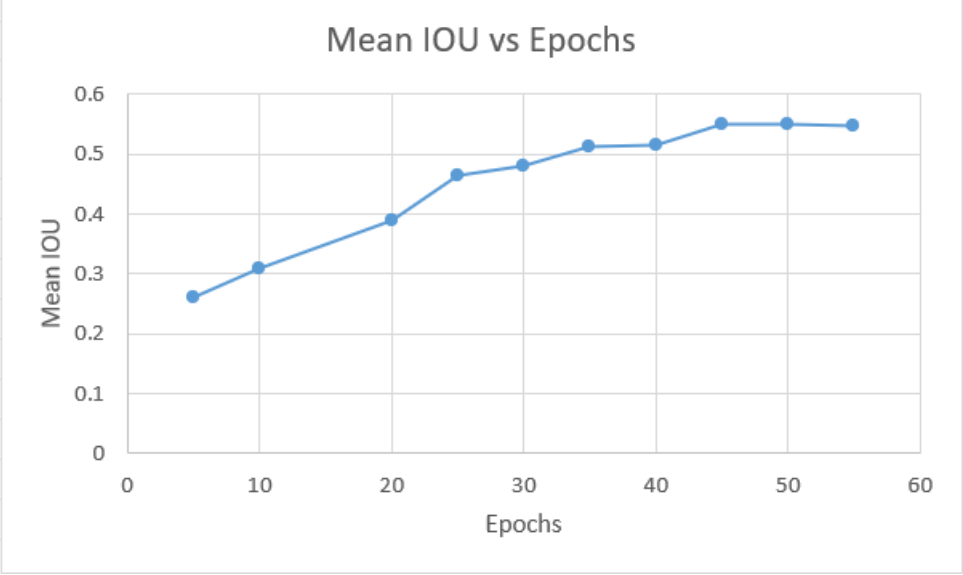


Figure 86: Mean IOU vs Epochs of 3D U-Net with Average Pooling and 8 filters

Based on Figure 86, the mean IOU score converges at 45 epochs.

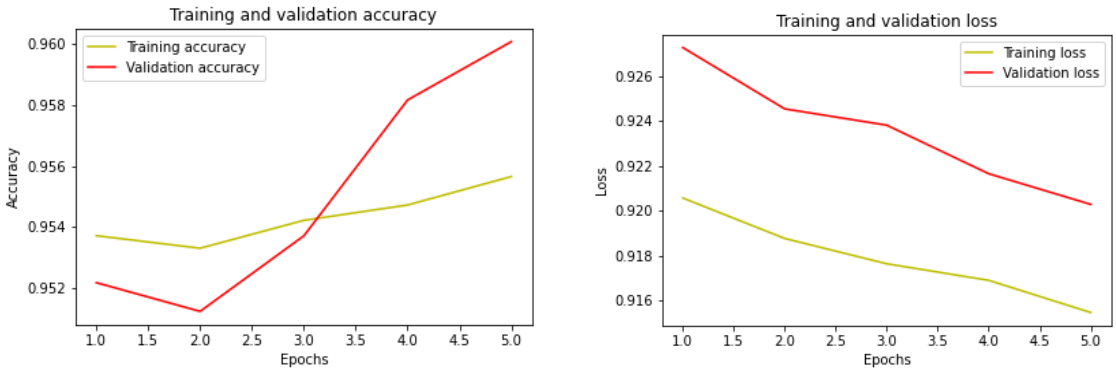


Figure 87: Training and validation accuracy and Training and Validation loss at 10 epochs

Based on Figure 87, the training accuracy started from 0.954 and increased to 0.956, while the validation accuracy increased from 0.952 to 0.96. The training loss started from 0.920 and decreased to 0.916, while the validation loss decreased from 0.926 to 0.922. It shows that the model is still learning and improving at 10 epochs.

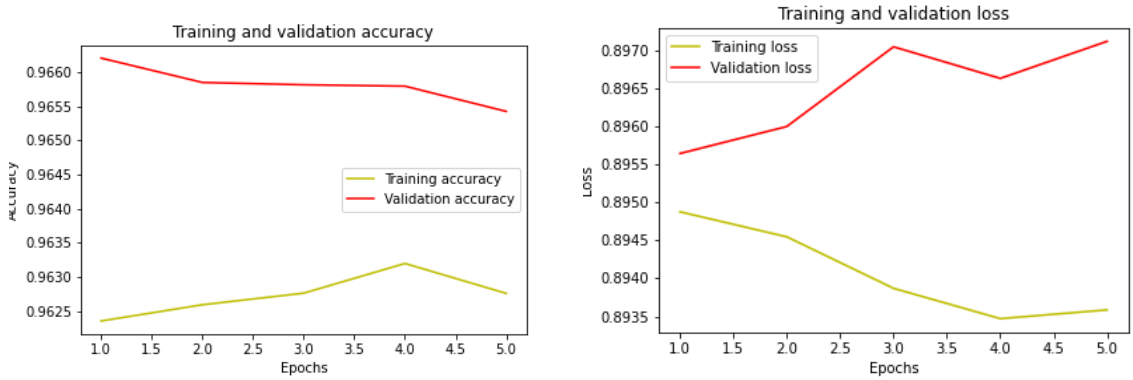


Figure 88: Training and Validation Accuracy and Training and Validation Loss at 45 epochs

Based on Figure 88, the training accuracy started from 0.9625 and increased slightly to 0.9630, while the validation accuracy started at 0.966 and decreased to 0.9655. The training loss started from 0.8950 and decreased to 0.8935, while the validation loss started from 0.8955 and increased to 0.897. It shows that the model is already overfitting at 45 epochs because the training accuracy and loss are improving while the validation accuracy and losses get worse performance.

4.1.3 Training Score of 3D U-Net Average Pooling with 16 filters

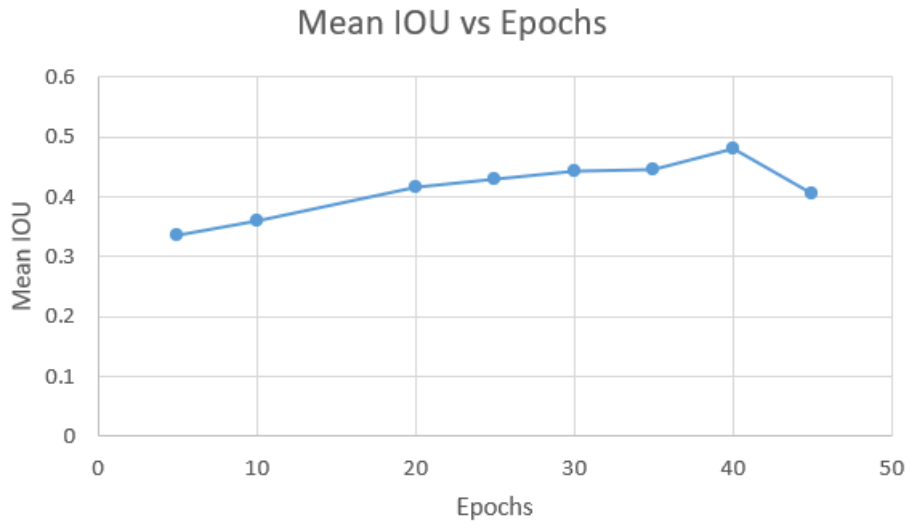


Figure 89: Mean IOU vs Epochs of 3D U-Net with Average Pooling and 16 filters

Based on Figure 89, the mean IOU score converges at 35 epochs and starts overfitting at 45 epochs.

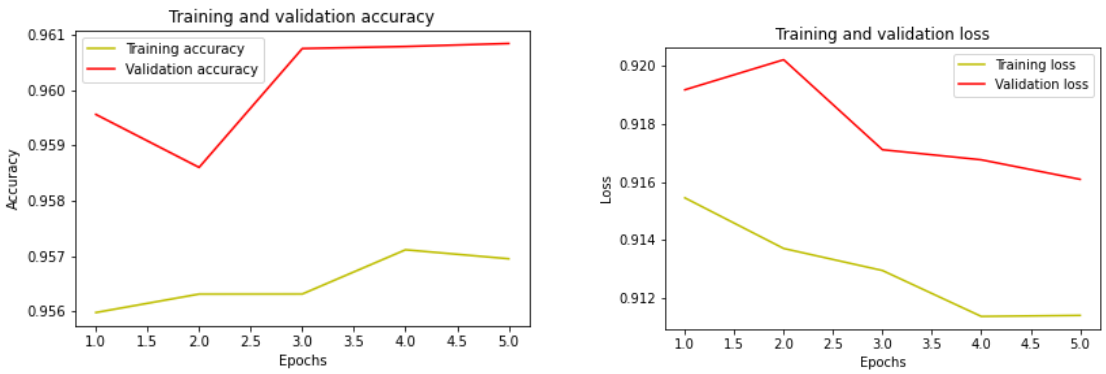


Figure 90: Training and Validation Accuracy and Training and Validation Loss at 10 epochs

Based on Figure 90, the training accuracy started from 0.956 and converged at 0.957, while the validation accuracy started at 0.960 and converged at 0.961. The training loss started from 0.916 and converged at 0.912, while the validation loss decreased from 0.919 to 0.916. It shows that the model is learning and improving at 10 epochs.

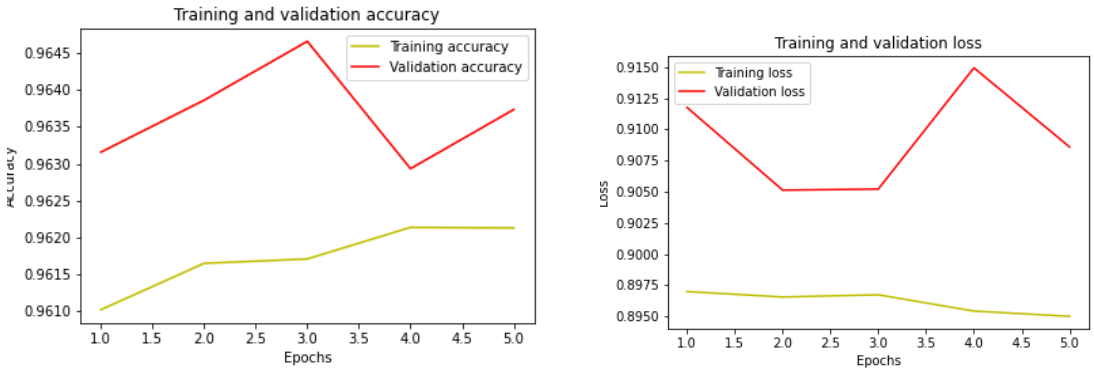


Figure 91: Training and Validation Accuracy and Training and Validation Loss at 45 epochs

Based on Figure 91, the training accuracy started from 0.9610 and increased to 0.9615, while the validation accuracy went up and down and eventually got the same score. The training loss started from 0.8975 and converged at 0.8950, while the validation loss went up and down and eventually got the same score. It shows that the model is not learning and improving at 45 epochs.

4.1.4 Discussion of the performance of different U-Nets on BRATS

Table 7: Highest Mean IOU score of different U-Net

U-Net	Highest Mean IOU score
U-Net 1 (Max pooling ,16 filters)	0.71
U-Net 2 (Average pooling ,8 filters)	0.45
U-Net 3 (Average pooling ,16 filters)	0.35

In conclusion, U-Net with max pooling layer and 16 filters performs the best and gets the highest mean IOU score of 0.7. U-Net with average pooling layer has overfitting issues and converges at 0.35 (16 filters) and 0.45 (8 filters) mean IOU Score. Several reasons might cause max pooling layer to outperform average pooling layer.

The first reason is that the strong feature, like the edges of the tumor is important to determine the position of the tumour and for accurate segmentation. Max pooling layer preserves strong features like sharp edges in the images, while average pooling layers might blur these strong features. So, max pooling will be more effective in preserving important features like edges of brain tumors for better segmentation. [16]

In some brain MRI images, the brain tumor only appears as a small part of the image while the rest are background. In this situation, background pixels will dominate the average pooling elements because all the pixels are averaged out and cause it to overlook the small brain tumour parts. On the other hand, max pooling is good as it preserves strong features of the brain tumor parts, even if they are small. [69]

4.1.5 Discussion on how to overcome the overfitting of U-Net

In the previous discussion in section 4.1.1, my U-Net with average pooling layers has overfitting issues. Therefore, there is a proposed pooling method to reduce the overfitting issue. Nirthika Rajendran et al. suggest that stochastic pooling can reduce overfitting in CNN in medical image analysis. Stochastic pooling chooses a value randomly from a probability distribution instead of normal max pooling, which chooses the largest value in a predetermined zone. It adds noise to the pooling process and aids in lowering network overfitting. The benefits of stochastic pooling for medical image analysis are covered in the original paper, especially on the limited training data. When working with medical images with significant noise and fluctuation levels, stochastic pooling can minimize the model's sensitivity to minor changes in the input. [69]

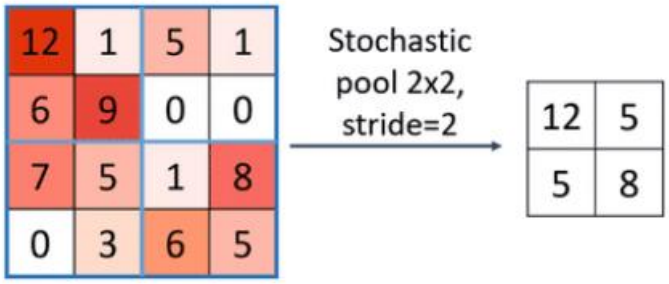


Figure 92: Stochastic Pooling process [70]

4.1.6 Results of U-Net on BRATS

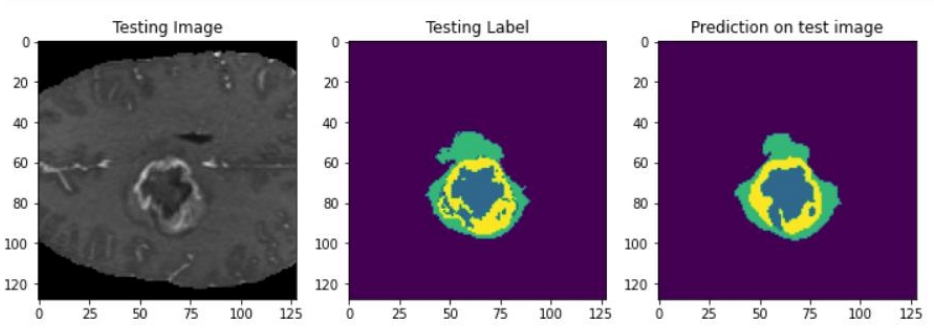


Figure 93: Testing Images, Ground Truth Mask and Prediction Mask

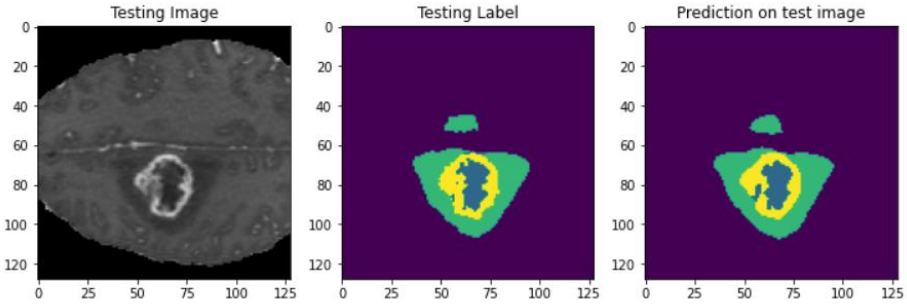


Figure 94: Testing Images, Ground Truth Mask and Prediction Mask

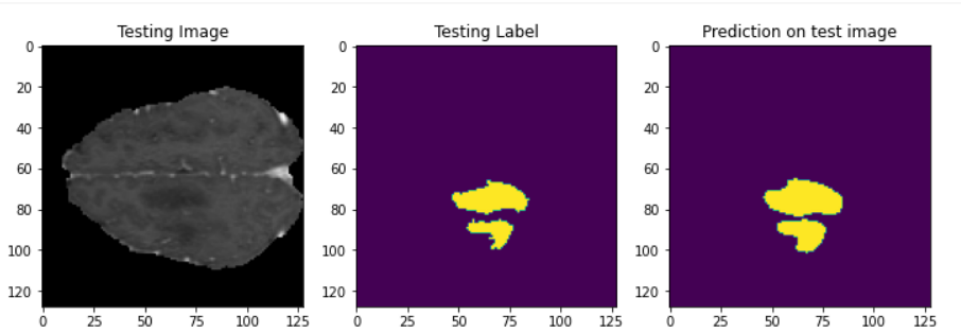


Figure 95: Testing Images, Ground Truth Mask and Prediction Mask

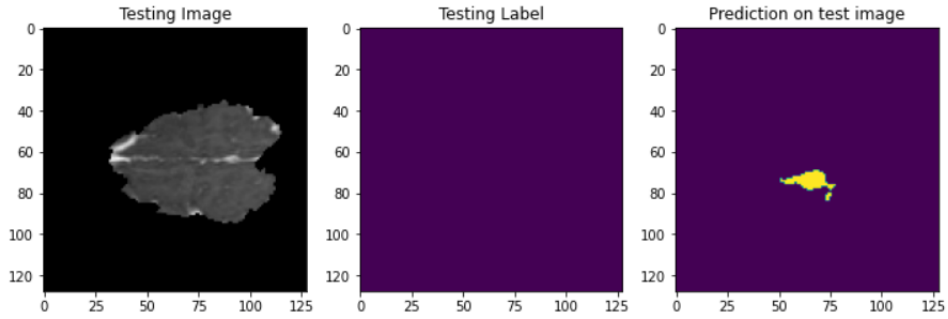


Figure 96: Testing Images, Ground Truth Mask and Prediction Mask

Based on Figure 96, the model has a wrong prediction by predicting healthy brain parts as a tumor (false positive). Based on Figures 93,94 and 95, the prediction by the model is slightly different from the ground truth mask.

4.1.7 Comparison different models on BRATS

Table 8: Comparison of different models on BRATS

	My U-Net	DeepLabv3+	MM-LinkNet
Number of Parameters	10 million	41 million	31 million
Training Images	590 (3D images)	-	-
Testing Images	148 (3D images)	-	-
Epochs	90	80	80
Accuracy	0.989	0.996	-
Dice Score	0.827	0.92	0.777
IOU Score	0.724	0.93	0.717
Recall	0.824	-	0.766
Precision	0.84	-	0.896

4.1.8 Discussion of results of different models on BRATS

The IOU score is 10% lower than the Dice Score for my U-Net. Both scores measure the similarity between the predicted and ground truth segmentation masks. However, they are calculated differently. Dice Score emphasize more on true positive rate because it rewards the model when it predicts the positive pixels (real tumor). Dice scores give greater weights for true positives than false positives and negatives. It is proven in the dice score formula; the numerator only has the overlap between predicted and ground truth segmentation which is the true positives and does not affect by the true negative. IOU score also rewards the model for true positive prediction, but false positive and false negative contribute more to the IOU score than the dice score. The false positive and false negative contribute more to the denominator. Therefore, the IOU score is usually lower than the dice score because of the false positive and false negative.

Based on Table 8, my U-Net's dice score is lower than DeepLabV3+ but higher than MM-LinkNet. It means that my U-Net is performing alright even though it has a lower dice score than DeepLabV3+ because the number of parameters of DeepLabV3+ is four times of my U-Net. However, there is still room for improvement in getting higher true positives. [71], [72]

Based on Table 8, my U-Net's IOU score is similar to MM-LinkNet but much lower than DeepLabV3+. It means that my model predicts less true positives and more false positives or false negatives. IOU score performs even worst when the false positive or false negative is high. In order to get a high IOU score, true positives must be increased, false positives and false negatives must be low.

Recall and Precision are often used together to detect errors. Recall is used to detect false negatives, and Precision is used to detect false positives. Based on Table 7, my U-Net's Recall score is higher than MM-LinkNet, but my U-Net's Precision score is lower than MM-LinkNet. It means that I need to work on lowering the false positives. In the previous discussion, I'm uncertain whether false positives or negatives contribute to low IOU scores. However, a low precision score proves that false positive is higher than false negative.

In conclusion, my 3D U-Net needs to improve on getting higher true positives and lower false positives.

4.1.9 Discussion of how to increase true positive and reduce false positive for my 3D U-Net

There are several ways to improve the true positive and false positive of my 3D U-Net model.

The first way is to adjust the class weight of the loss function. Phan, Huy et al. mention that the weighted loss aims to address the imbalanced dataset problem. [73] So, having a proper weight loss function is important. The loss function used **in my 3D U-Net** is Dice Loss, and the weight of the loss function for each class is distributed equally. There are four classes for my dataset: one background class and three different tumor classes. The weights of the tumor classes can be increased while the weight for the background class can be reduced. It can penalise the model more when it mispredicts the tumor to increase my true positive and reduce my false positive.

The second way is to adjust the contribution of two loss functions in total loss. **In my 3D U-Net**, I use dice loss plus focal loss as the total loss. Since the purpose of the focal loss is only to address the class imbalance problem by assigning more weights to easily misclassified examples, the contribution of the dice loss can be increased compared to the focal loss. For instance, I can use dice loss plus 0.5 multiplied by the focal loss as the total loss. By increasing the contribution of dice loss in a total loss, the model will emphasise more on rewarding true positives and penalising false positives.

The third way is to fine-tune the hyperparameters of the U-Net, such as learning rate, batch size and types and values of regularisation. The way to fine-tune is to experiment with different hyperparameters values and find the best combination. **In my 3D U-Net**, I use only dropout instead of other regularisation methods. Other popular regularisation methods include L1, L2 and Elastic-Net regularisation.[74] Different dropout and regularisation rates for each method can be tested to determine which suits U-Net best.

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Figure 97: Regularization methods and Test Classification Error[75]

Based on Figure 97, it shows that L2 regularization combined with dropout has lower classification error than using L2 regularization alone. So, it is worth investigating the effect of a different combination of regularization methods on U-Net.[75]

4.2 Chest X-Ray Results

4.2.1 Comparison of 2D U-Net with 4 different Pooling Layer

Table 9: Comparison of different pooling layers

	Max Pooling	Average Pooling	Hybrid Pooling	ASPP
Accuracy	0.941	0.956	0.950	0.955
Dice score	0.866	0.902	0.887	0.898
Jaccard score	0.756	0.824	0.801	0.818
Recall	0.767	0.83	0.808	0.827
Precision	0.997	0.992	0.991	0.985

All the pooling layers are trained with the same original dataset and 10 epochs. Among all the pooling layers, average pooling layer performs the best in terms of all score metrics.

Average pooling outperforms max pooling in Chest X-Ray Dataset because there are no imbalanced class data in this dataset. Unlike in the BRATS dataset, some of the data has imbalanced class data where the background pixels dominate the image, the Chest X-Ray Dataset has equal distribution in lung pixels and background pixels in the image. Average pooling gives equal importance to all the pixels in the image and works well when no imbalanced class data is dominated by background pixels in the dataset. Max pooling selects the maximum value in the feature map and discards all the other values. It can help to find tiny tumors in the BRATS dataset but lose important information in Chest X-Ray Dataset.

The average pooling, ASPP and hybrid pooling have similar results, and they all outperform max pooling. ASPP and hybrid pooling have good generalization ability, which is mentioned in their original paper, thus reducing the overfitting of U-Net.

Although average pooling performs the best among all other pooling layers tested, it still has a similar problem faced in the U-net trained with the BRATS dataset: low IOU score and Recall. It means the true positive is low, and the false negative is high.

To improve the U-Net performance on Chest X-Ray, U-Net with average pooling was selected to train with an augmented dataset.

4.2.2 Training scores for 2D U-Net with average pooling layer

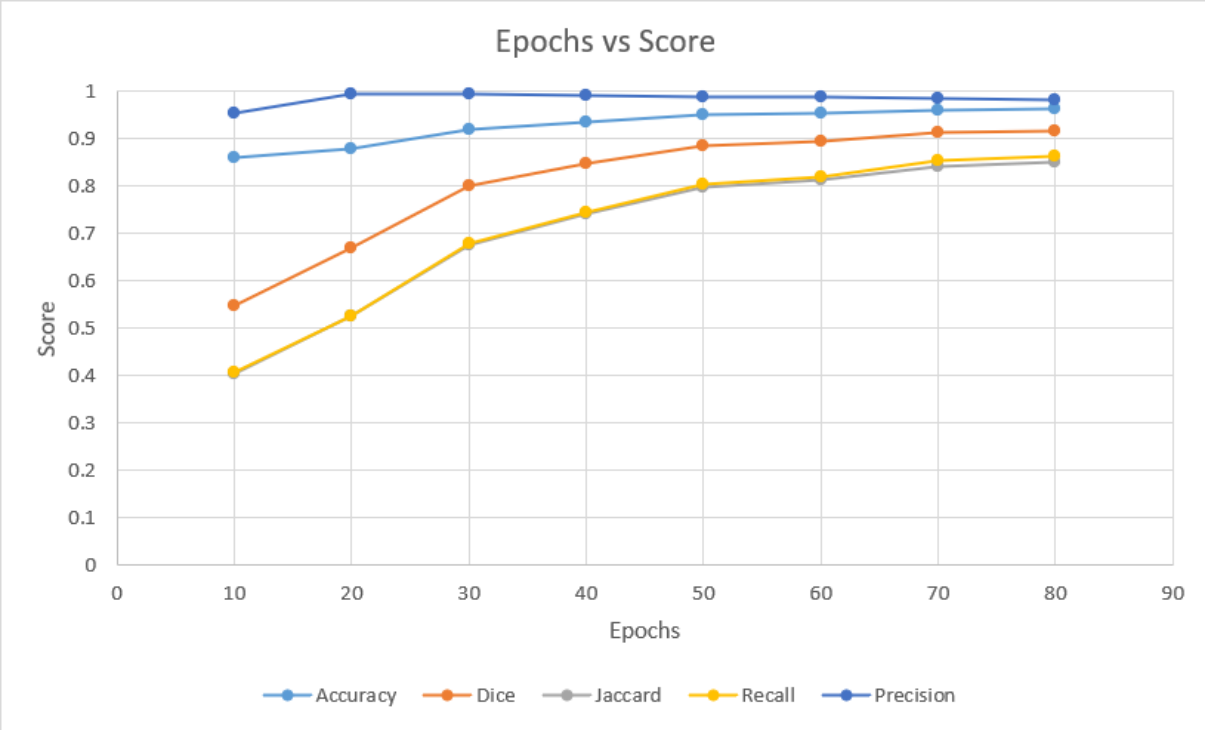


Figure 98: Epochs vs Score of 2D U-Net with average pooling

Based on Figure 98, the accuracy and precision are roughly the same throughout the training. The Dice score starts from 0.5 and converges at around 0.9 at 60 epochs. The Jaccard and Recall score starts from 0.35 and converges at around 0.85 at 60 epochs.

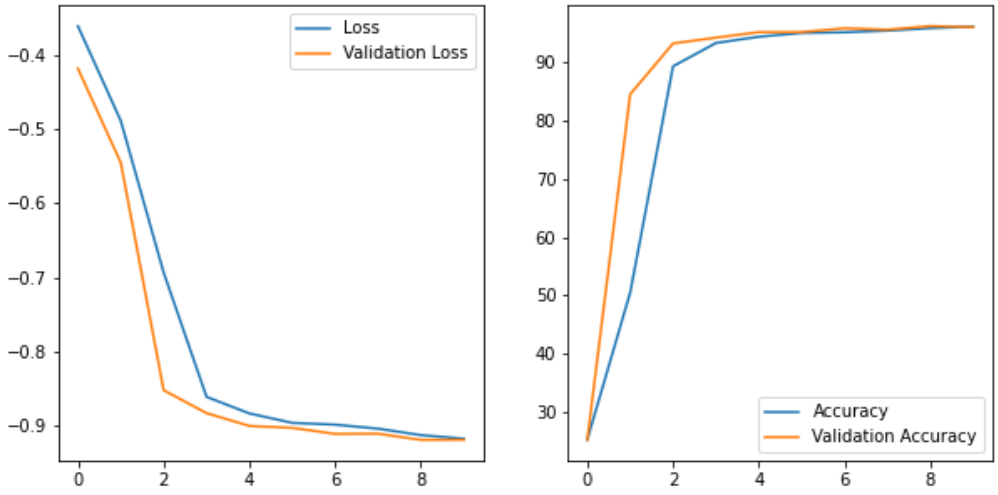


Figure 99: Training and Validation Accuracy and Loss at 10 epochs

Based on Figure 99, the training and validation loss starts from -0.4 and converges at -0.9. The training and validation accuracy starts at 20 and converges at 90. It shows that the model is still learning and improving at 10 epochs.

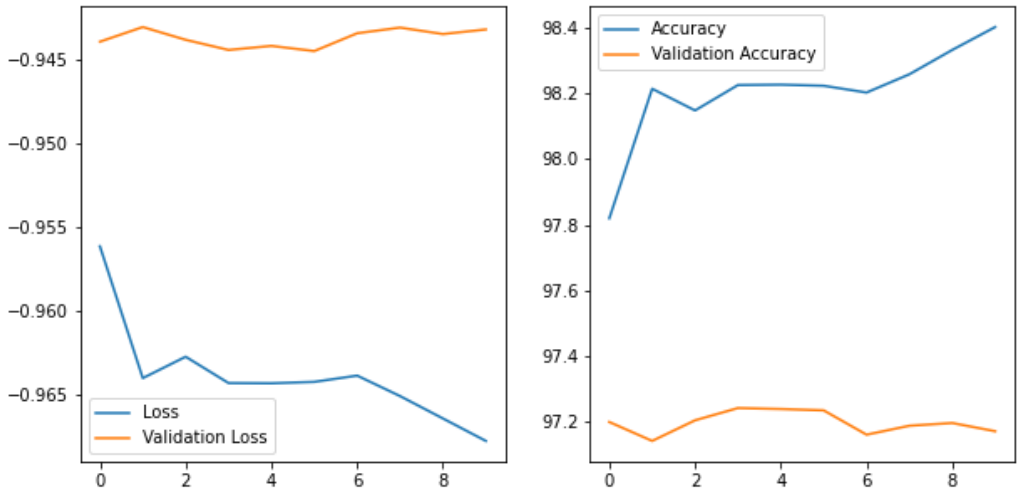


Figure 100: Training and Validation Accuracy and Loss at 60 epochs

Based on Figure 100, the training loss remains at around -0.945, and the validation loss starts from -0.955 and reduces to -0.967. The training accuracy starts at 97.8 and increases to 98.4, while validation accuracy remains at around 97.2. It shows that the model is not learning and improving much at 70 epochs because the validation loss and accuracy remain the same. This observation matches another observation in Figure 97 that the dice score, IOU score, recall score, and precision score converge at 60 epochs.

4.2.3 DeepLabV3+ in Matlab training on Chest X-Ray Dataset

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Mini-batch Loss	Base Learning Rate
1	1	00:00:12	53.86%	0.7827	0.0100
1	50	00:02:42	97.43%	0.0736	0.0100
1	100	00:05:13	97.25%	0.0695	0.0100
2	150	00:07:48	98.06%	0.0492	0.0100
2	200	00:10:32	98.24%	0.0455	0.0100
3	250	00:13:03	98.04%	0.0447	0.0100
3	300	00:15:28	98.36%	0.0402	0.0100
3	312	00:15:59	98.47%	0.0382	0.0100

Figure 101: Mini Batch Accuracy and Loss over Epochs

DeepLabV3+ in Matlab was used to train the same augmented dataset with 3 epochs. Based on Figure 101, the mini-batch accuracy successfully increases from 53.86% to 98.47%, and the mini-batch is reduced from 0.7827 to 0.0382.

4.2.4 Result of Original Image and Mask and Predicted Mask of U-Net with Average Pooling

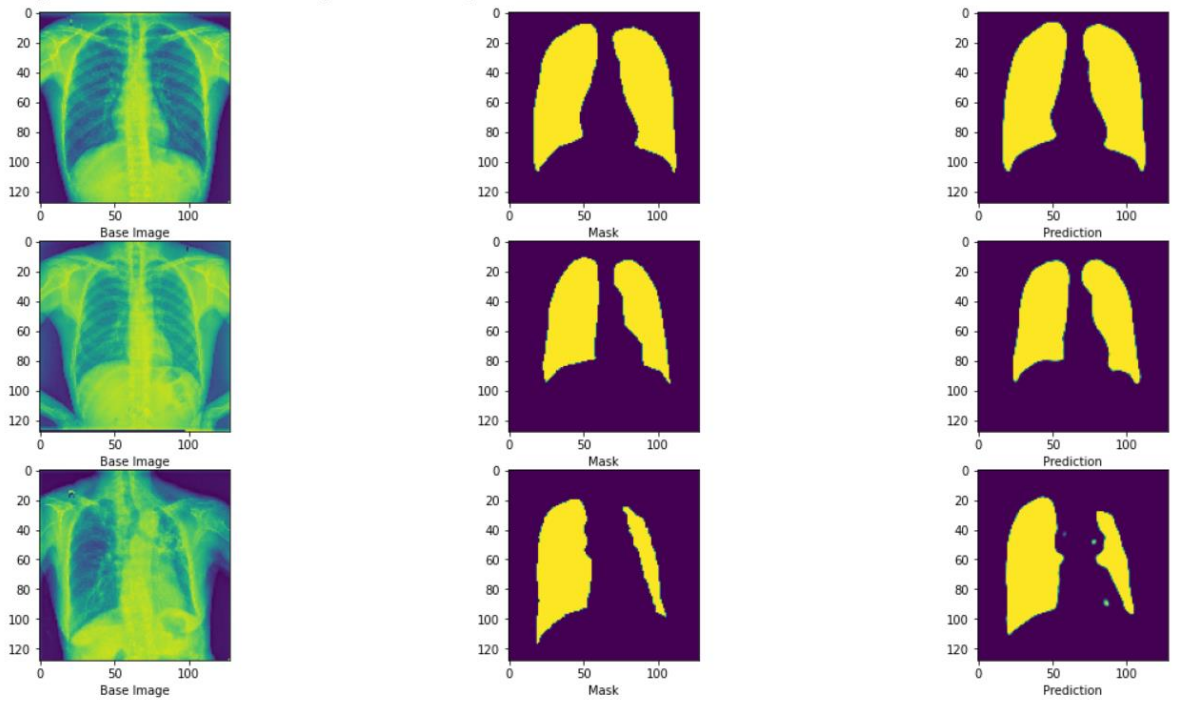


Figure 102: Test Image, Ground Truth Mask and Predicted Mask

Based on Figure 102, the first and second prediction of the model is similar to the ground truth mask, while the third prediction has a small false positive at the right lung compartment.

4.2.5 Comparison between DeepLabV3 from Matlab and My U-Net with Average Pooling

Table 10: Comparison between DeepLabV3+ and U-Net

	DeepLabV3	My U-Net with average pooling
Epochs	3	80
Size	98 MB	9.2MB
Training Images	838	838
Test Images	210	210
Accuracy	0.966	0.963
Dice score	0.978	0.917
Jaccard score	0.957	0.850
Recall	0.978	0.863
Precision	0.978	0.982

Based on Table 10, DeepLabv3+ performs better overall, especially in Dice score, Jaccard score and Recall. However, the size of DeepLabv3+ model is 10 times bigger than the modified U-Net. Other than the size of the model, several differences between these two models might cause the difference in the score.

The first difference is the architecture. U-net uses an encoder and decoder to capture and localize the feature. In contrast, DeepLabV3+ uses Feature Pyramid Network (FPN) with atrous convolutions and atrous spatial pyramid pooling (ASPP) to capture and localize features. Due to the complex structure of DeepLabV3+, it can capture more complex features, but the model size is bigger. On the other hand, U-Net has a simpler structure and fewer parameters.

The second difference is the optimizer used. DeepLabV3+ uses SGDM optimizer, while U-Net uses Adam optimizer. SGDM speed up the model's training and reduce the oscillation in training by training the model in batches and using momentum parameter. Adam modifies the learning rate every epoch to speed up the training process and improve the model's performance. The optimizers' performance depends on the dataset's type or size. Adam is good at handling datasets with noisy and sparse gradients, but not for SGDM.

The choice between DeepLabV3+ and U-Net depends on the available computational resource. In conclusion, it is a trade-off between performance and computational resources. However, U-Net still has huge potential to improve with the same model size.

4.2.6 Comparison between CNN based U-Net and My U-Net with Average Pooling

Table 11: Comparison with my U-Net and CNN based U-Net

	My U-Net	CNN based U-Net
Number of Parameters	0.7 million	8.6 million
Datasets	1. ShenZhen (SZCX), 2. Montgomery County (MC)	1. ShenZhen (SZCX), 2. Montgomery County (MC), 3. Japanese Society of Radiological Technology (JSRT)
Image Size	512x512	256x256
Training Images	838	755
Testing Images	210	189
Epochs	80	50
Batch Size	16	8
Accuracy	0.963	0.938
Dice Score	0.917	0.918
Jaccard Score	0.850	0.91
Recall	0.863	0.983
Precision	0.982	-

Based on Table 11, my U-Net performs similarly in dice score but has a lower Jaccard score and recall score. However, the number of parameters for CNN-based U-Net is ten times more than the number of parameters of my U-Net.[76]

If the dice scores are similar, the number of true positives predicted is similar because true positives contribute a lot to the dice score. However, my U-Net's Jaccard score is 10% less than CNN-based U-Net. It means my false positive or false negative is higher since the true positives are similar. My U-Net's recall score is 10% less than CNN-based U-Net, which means my false negative is higher.

In conclusion, my U-Net has higher false negatives and needs to improve on getting lower false negatives.

4.2.7 Discussion on how to reduce false negative for my 2D U-Net

There are several ways to reduce the false negative of my 2D U-Net model.

The first way is optimal thresholding. For binary segmentation, the sigmoid function is the activation function, which outputs a value between 0 and 1. A threshold is needed to determine whether it is the object of interest or background. Normally, the threshold is set at 0.5. However, 0.5 threshold value is not always the best. Therefore, optimal thresholding is a post-processing technique used to find the best threshold value for the model.

Xu Yanzhe et al. proposed U-Net with optimal thresholding using Otsu's thresholding to detect small blobs in medical images. It outperforms other thresholding techniques and shows a lot of room for improvement. [77]

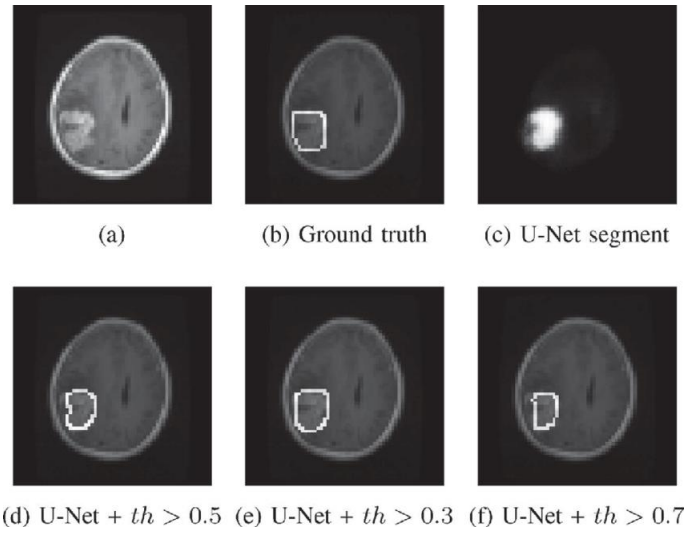


Figure 103: Brain Tumor Segmentation with different threshold [78]

Isunuri, Bala Venkateswarlu, et al. proposed adaptive thresholding to improve brain tumor segmentation. It outperforms other models in recall score and dice score significantly. Based on Figure 103, it shows the effect of different thresholds to the output of the segmentation.[78]

The second way is to fine-tune the hyperparameters of the U-Net, such as learning rate, batch size, and types/values of regularization. **In my 2D U-Net,** only early stopping is used as a type of regularization. Other popular regularization methods include dropout, L1, L2, and Elastic-Net regularization can be combined with early stopping to get a better effect. [74] As mentioned in an earlier discussion in section 4.1.6, combining different regularizations can improve the model's performance. Still, experiments are needed because different models and datasets have different properties that will affect the performance of regularization.

Another regularization method worth mentioning is batch normalization. S. Ioffe, et al. address the internal covariate shift issue by normalizing layer inputs. By eliminating internal covariate shift, the model can be trained faster. Batch normalization allows the model to train with higher learning rates and care less about the initialization of the model. The authors also mention that

batch normalization will sometimes eliminate the need for dropout. However, batch normalization with other regularization methods like dropout and L2 regularization must be dealt with carefully. The author also removed dropout and reduced L2 regularization to speed up the batch-normalized model. The use of dropout and L2 regularization might cause side effects to the model since they are all doing the same job. The authors tested Inception-V3, a classification model with batch normalization, which improved its performance significantly. Batch normalization lets the classification model achieve the same accuracy with 14 times fewer training steps. Besides, they also used ensemble batch-normalized networks to improve the performance of ImageNet classification by surpassing the accuracy of human raters by achieving a 4.9% top validation error. [79]

Figure 104: Image before and after morphological processing [80]

The third way to reduce false negatives is using post or pre-processing techniques like the morphological method. Morphology comprises techniques that may be applied either as a post-processing technique on the output of the Image Segmentation stage or as a pre-processing technique on the input data. When segmentation is done, morphological operations may be applied to the segmented image to correct flaws and provide details about the structure and shape of the image, as illustrated in Figure 104. [80]

Setyawan, Robert et al. study the effect of morphological pre-processing on the Fuzzy C-means model to segment Brain Tumor MRI images. It shows that the MRI images produced with morphological method can produce a more detailed image with less noise. The model also shows a 3% improvement in accuracy after applying the morphology processing method.[81]

4.3 Results of Coral Dev Board Mini Inference

4.3.1 Result of Ground Truth Mask and Predicted Mask on Coral Dev Board Mini

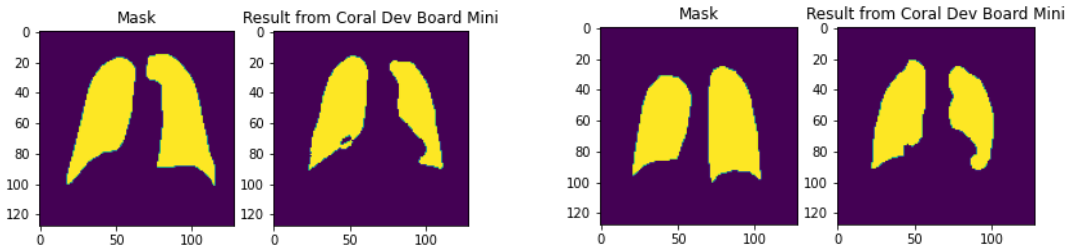


Figure 105: Comparison between Ground Truth Mask and Predicted Mask on Coral Dev Board Mini

4.3.2 Comparison of Predicted Mask on Host Computer and Predicted Mask on Coral Dev Board Mini

Table 12: Comparison of predicted mask on host computer and edge device

	Predicted Mask on Host Computer	Predicted Mask on Coral Dev Board Mini
Model Type	Tensorflow	Tensorflow Lite
Model Size	9158 KB	2998 KB
Accuracy	0.963	0.955
Dice score	0.917	0.904
Jaccard score	0.85	0.832
Recall	0.863	0.854
Precision	0.982	0.950

After converting the model from Tensorflow version to Tensorflow Lite, the model size is reduced from 9158KB to 2998 KB. The model scores only drop around 1%, which is acceptable. Unfortunately, no paper runs segmentation experiments on edge devices using Chest X-Ray Dataset. However, Owais Ali et al. modified U-Net to test the performance on Intel NCS-2 (a popular edge device). They scored a highest dice score of 0.96 on BRATS and 0.94 on Heart MRI Images, and 0.74 on ZNSDB (Ziehl-Nelsen-stained sputum smear microscopy). It is hard to compare the performance because they use different datasets and metrics.

Since my model had a similar score before and after converting to Tensorflow Lite, my Tensorflow Lite model is still comparable with CNN-based U-Net based on previous discussions in section 4.2.6.

4.4 Discussion on improving performance for both U-Net

The first way is ensemble methods. Ensemble methods are common in statistics and machine learning. It combines several algorithms or models to create one optimal prediction model to achieve higher predicted performance than any individual algorithms or models. For example, several U-Net can be trained with different hyperparameters and architecture. The predictions of the different U-Nets can be averaged to be the final prediction. It can reduce the overfitting and bias of the model. However, ensemble method is costly in terms of computational resources and time because multiple models need to be trained. Other than that, ensemble methods are hard to interpret, and any mistake in one of the models might cause bad results than an individual model. Therefore, the trade-off and the risk must be considered.

There are several popular ensemble methods. Bagging and Boosting Ensemble method will be explained. In these cases, U-Net or other model is represented as a classifier.

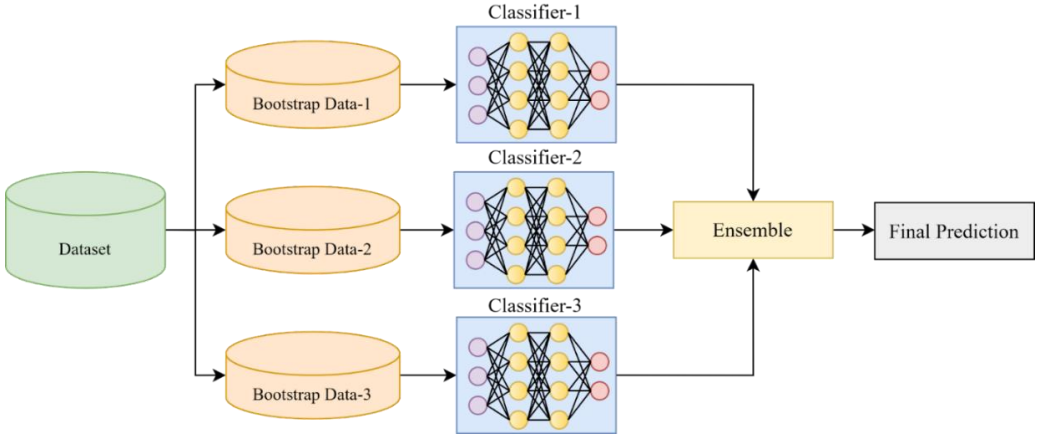


Figure 106: Bagging Ensemble Mechanism [82]

Based on Figure 106, it shows a bagging ensemble mechanism. This bagging ensemble method processes the data in parallel. The dataset is separated randomly into bootstrap data. Some data points will reappear in different bootstrap data. The classifiers are trained with different bootstrap data and all the predictions are ensembled. The average of the ensembled predictions is calculated as the final prediction.[82]

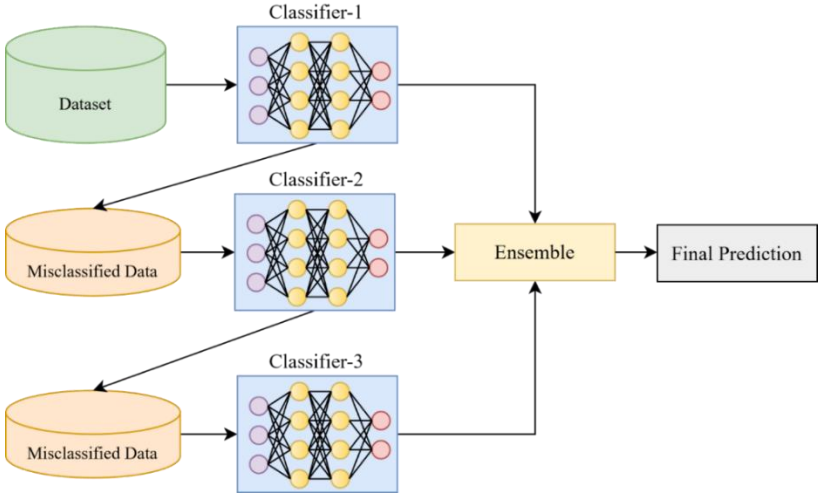


Figure 107: Boosting Ensemble Mechanism [82]

Based on Figure 107, it shows a boosting ensemble mechanism. This boosting ensemble method processes the data sequentially. The whole dataset is input to classifier 1 and the predictions are evaluated. If classifier 1 fails to make accurate predictions, the misclassified data will be the input of classifier 2. Classifier 2 can focus on the misclassified data feature's space and learn the mistakes. The same steps are repeated and eventually all the predictions of the classifiers are ensembled. The average of the ensembled predictions is calculated as the final prediction.[82]

Rohit Kundu proposed a new method based on ensemble learning segment aerial images. The paper created a decision tree using the new ensemble algorithm, and it shows better accuracy than other methods. [83]

The second way is transfer learning. Although transfer learning might increase the model's parameters, it is the easiest and fastest way to achieve good results. Transfer learning in U-Net is simple: replacing the encoder path with pre-trained CNN models. Training with these pre-trained models has several benefits.

The first benefit is that these pre-trained models only require a small number of labeled datasets for training because the pre-trained model already learned to capture important features from large datasets; only a small number of data is needed to fine-tune these pre-trained models.

The second benefit is that transfer learning can improve efficiency when training multiple similar models. Complex models take a long time to train, but with transfer learning, the model does not need to train from scratch whenever a similar model has already pre-trained. The time and computational resources can be used to train multiple similar models instead of just one.

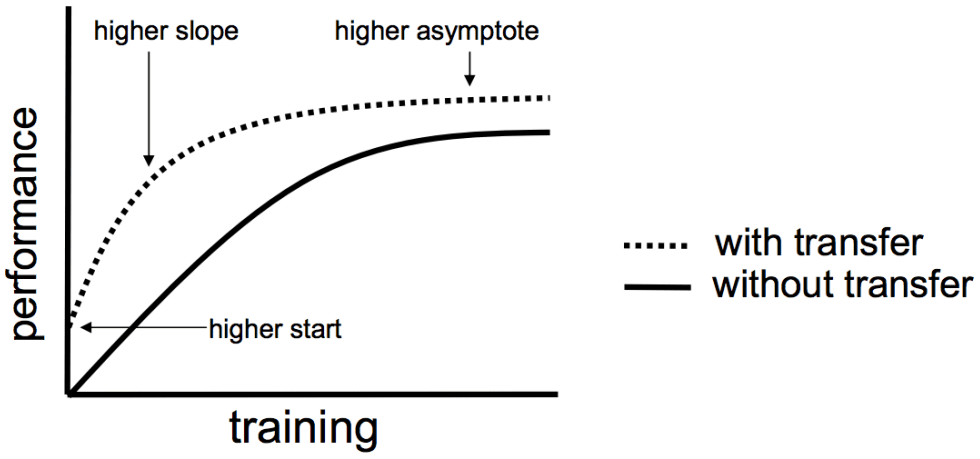


Figure 108: Performance vs Training with and without transfer learning [84]

Based on Figure 108, the model with transfer learning has a higher start, slope and asymptote. It means that the model has better performance at the start of the training, a better rate of improvement and converges at better results.

Ivana Zadro Matovinovic et al. use ResNet34 as the backbone for U-Net and compare the results with traditional U-Net. The U-Net with pre-trained ResNet34 performs better in terms of dice score by 2%. [85]

5.0 Conclusion

This paper discusses the importance of medical semantic segmentation and edge device to the community. Then, this paper discusses different medical semantic segmentation techniques and models. This project aims to investigate and compare the effects of different pooling layers on medical segmentation models and datasets and optimize their performance. U-Net was selected as the model because it is robust and simple to modify. 2 types of datasets were chosen: the 3D BRATS dataset and the 2D Chest X-Ray dataset. Two types of datasets were selected to investigate the effect of pooling layer when training with datasets with different properties like dimensions and class imbalance. The trained 2D U-Net was implemented in the edge device, Google Coral Dev Board Mini. The results are compared with the results on the host computer. Another model, DeepLabv3+ was trained with Chest X-Ray dataset to compare it with 2D U-Net. Comparison with other models was also made for both 2D and 3D U-Net to find the weaknesses that need to be improved. Lastly, the ways to improve the weaknesses are discussed.

5.1 Achievements of this project

In this project, two datasets are augmented: the 3D BRATS dataset and the 2D Chest X-Ray dataset. 2 types of U-Net which are 2D and 3D U-Net are trained respectively. Two 3D U-Net with different pooling layers, including max pooling and average pooling were trained and compared. The 3D U-Net with max pooling outperforms 3D U-Net with average pooling. 3D U-Net with average pooling has overfitting issues. Four 2D U-Nets with different pooling layers, including max pooling, average pooling, hybrid pooling and Atrous Spatial Pyramid Pooling (ASPP), were trained and compared. 2D U-Net with average pooling outperforms all the pooling layers. The 3D U-Net with max pooling achieved 0.989 in accuracy score, 0.827 in Dice score, 0.724 in IOU score, 0.824 in Recall score, and 0.84 in Precision score. The 2D U-Net with average pooling achieved 0.963 in accuracy score, 0.917 in Dice score, 0.85 in IOU score, 0.863 in Recall score, and 0.982 in Precision score.

From the results and discussion, I know that max-pooling performs well in a class-imbalanced dataset like the BRATS dataset and average pooling performs well in normal dataset like the Chest X-Ray dataset. Hybrid pooling and Atrous Spatial Pyramid Pooling (ASPP) perform similarly to average pooling in Chest X-Ray dataset.

Other than that, pre-trained DeepLabV3+ in Matlab was also used to train with Chest X-Ray dataset so that it can be compared with my 2D U-Net with average pooling. The trained DeepLabV3+ achieved 0.966 in accuracy score, 0.978 in Dice score, 0.957 in IOU score, 0.978 in Recall score and 0.978 in Precision score.

The 2D U-Net is implemented on Google Coral Dev Board Mini and achieved 0.955 in accuracy score, 0.904 in Dice score, 0.832 in IOU score, 0.854 in Recall score and 0.95 in Precision score. The model size is reduced from 9158kB to 2998kB.

The number of parameters of both 3D and 2D U-Net is reduced to 10 million and 0.7 million, which is significantly lower than the original U-Net with 34 million parameters.

5.2 Deliverables

Deliverable 1 is discussed in section 2.6, 2.7, 2.8 and 2.9. In section 2.6 and 2.7, different types of medical datasets are discussed. In section 2.8, different types of models for medical segmentation are discussed. In section 2.9, edge device is discussed.

Deliverable 2 is discussed in section 3.4.2, 3.4.3, 3.5.2 and 3.5.3. In section 3.4.2 and 3.4.3, the pre-processing and augmentation steps of BRATS dataset is discussed, and the augmentation results are shown. In section 3.5.2 and 3.5.3, the pre-processing and augmentation steps of Chest X-Ray dataset is discussed, and the augmentation results are shown.

Deliverable 3 is discussed in section 3.4.4, 3.4.5, 3.5.4 and 3.5.5. In section 3.4.4 and 3.4.5, the training and testing steps for BRATS dataset is discussed. In section 3.5.4 and 3.5.5, the training and testing steps for Chest X-Ray dataset is discussed. The result of the 3D U-Net is shown in section 4.1.3, and the result of the 2D U-Net is shown in section 4.2.4. Both of the results are decent.

Deliverable 4 is discussed in section 3.2, where modification of filter in U-Net is discussed to reduce the number of parameters.

Deliverable 5 is discussed in section 4.1.1 and 4.2.1. In section 4.1.1, the comparison between max pooling layer and average pooling layer in 3D U-Net has been discussed. In section 4.2.1, the comparison between max pooling layer, average pooling, hybrid pooling and ASPP in 2D U-Net has been discussed.

Deliverable 6 is discussed in section 4.1.4, 4.2.5, and 4.2.6. In section 4.1.4, the comparison between my 3D U-Net, DeepLabv3+ and MM LinkNet has been discussed. In section 4.2.5, the comparison between my 2D U-Net and DeepLabv3+ has been discussed. In section 4.2.6, the comparison between my 2D U-Net and CNN-based U-Net has been discussed.

Deliverable 7 is discussed in section 4.3.1 and 4.3.2. In section 4.3.1, the results between the original 2D U-Net and Tensorflow Lite 2D U-Net have been shown. In section 4.3.2, the result comparison of these 2 models has been discussed.

5.3 Reflection and improvements

There are some weaknesses in this project that needs to be reflected and improved.

The first weakness is time planning. One of the big limitations of this project is that my 3D U-Net only tested with max pooling and average pooling due to lack of time and computational resources. For 3D U-Net to converge, around 10 to 16 hours is needed for training. Due to the restriction of Google Colab, only 2 hours only training is allowed (with GPU) per day. The model can be trained more efficiently by planning how much training must be done at a certain date. For example, 8 days are required for one 3D U-Net to converge. To train 4 3D U-Nets with 4 different pooling layers till they converge requires around 32 days. 3D U-Net with Hybrid pooling and Atrous Spatial Pyramid Pooling can be trained with sufficient time. Other than better time planning, other platforms like high-performing computing and Google Cloud can be used to speed up the training process. However, learning to use other platforms can be challenging and time-consuming especially for new learners.

The second weakness is quantization. There is two ways to convert the Tensorflow model to Tensorflow Lite, which includes using a Tensorflow lite converter and quantization. Tensorflow Lite Converter directly converts the model to Tensorflow Lite version by reducing the model size and complexity without reducing the precision of the model so it can be used for inferencing on an edge device like Google Coral Dev Board Mini. Quantisation is a technique that significantly reduces the size of the model but also reduces the precision of the

parameters of the model. There are two types of quantization which include post-training quantisation and quantization-aware training. Quantization-aware training execute quantization when training, while post-training quantization executes quantization after the training process. Quantization-aware training performs better because this technique helps preserve the model's precision. In this project, post-training quantisation is used to convert the model since it was easier, but the converted model is performed badly with only 0.3 dice score. It is possible that too much information was lost during the post-quantization and I need to use quantization-aware training instead. However, quantization-aware training can only be done by retraining the model. But due to time restrictions, the TensorFlow model was converted using the Tensorflow Lite Converter. [86], [87]

5.4 Future work

With the current progress in this project, some improvements can be made to understand more about the factors that affect the performance of the models. First, more types of pooling layers like Lp pooling, Stochastic Pooling and Global Pooling can be trained with U-Net and compared. Other than pooling layers, other structures in U-Net like convolutional layers are worth investigating as well. Piao S et al. proposed U-Net with dilated convolution, which outperformed the original U-Net by 5.43% in the IOU score.[88] Other than dilated convolution, there are different types of convolutional layers like Depthwise Separable Convolutional Layer, Grouped Convolutional Layer and many more can be compared and investigated to further improve U-Net in the future.

In the results and discussion, hyperparameter tuning was mentioned to improve the performance of U-Net. However, experiments need to be done to find out the best parameter for each U-Net or dataset. The hyperparameters like batch size, learning rate, optimizers and regularization methods have significant impact on the performance of the U-Net. Experiments can be done with different combinations of hyperparameters, and the performance can be compared and investigated in the future. The advantage of hyperparameter tuning is that it can improve the performance of the model without increasing the number of parameters or size of the U-Net.

As a whole, many different structures and parameters can be compared and investigated to improve the performance of the U-Net. Unlike transfer learning, these modifications in structures and hyperparameter tuning do not increase the size of U-Net and use less computational power than ensemble learning.

Besides, for edge device inferencing, an GUI (Graphical User Interface) can also be developed in the future to let users like doctor or radiologist to run segmentation easily.

6.0 References

[1] O. Ali, H. Ali, S. A. A. Shah, and A. Shahzad, “Implementation of a Modified U-Net for Medical Image Segmentation on Edge Devices,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2022, doi: 10.1109/TCSII.2022.3181132.

[2] S. Porter, “UK hospital first to use AI cancer treatment tool, Healthcare IT News,” 2020. : https://www.healthcareitnews.com/news/emea/uk-hospital-first-use-ai-cancer-treatment-tool#:~:text=Addenbrooke’s%20Hospital%20in%20Cambridge%20is,the%20treatment%20of%20cancer%20patients. (accessed Oct. 23, 2022).

[3] S. Daley, “40 ai in healthcare examples improving the future of medicine, Built In,” 2020. https://builtin.com/artificial-intelligence/artificial-intelligence-healthcare (accessed Oct. 23, 2022).

[4] “Intel works with University of Pennsylvania in using privacy-preserving AI to identify brain tumors ,” *Intel*, 2020. https://newsroom.intel.com/news/intel-works-university-pennsylvania-using-privacy-preserving-ai-identify-brain-tumors/ (accessed Oct. 23, 2022).

[5] “Image Segmentation: The Basics and 5 Key Techniques,” *Datagen*. https://datagen.tech/guides/image-annotation/image-segmentation/ (accessed Feb. 18, 2023).

[6] “X-Ray vs. CT vs. MRI,” *Envision Radiology*. https://envrad.com/difference-between-x-ray-ct-scan-and-mri/#:~:text=X-rays%20and%20CT%20scans,CT%20scan%20or%20x-ray. (accessed Mar. 18, 2023).

[7] Kate Bush, “medical imaging (radiology),” *Whatls.com*. https://www.techtarget.com/whatis/definition/medical-imaging (accessed Mar. 18, 2023).

[8] A. Singh *et al.*, “Semantic segmentation of bone structures in chest X-rays including unhealthy radiographs: A robust and accurate approach,” *Int J Med Inform*, vol. 165, Sep. 2022, doi: 10.1016/j.ijmedinf.2022.104831.

[9] T. Agrawal and P. Choudhary, “Segmentation and classification on chest radiography: a systematic survey,” *Visual Computer*, 2022, doi: 10.1007/s00371-021-02352-7.

[10] S. Hou, T. Zhou, Y. Liu, P. Dang, H. Lu, and H. Shi, “Teeth U-Net: A segmentation model of dental panoramic X-ray images for context semantics and contrast enhancement,” *Comput Biol Med*, vol. 152, Jan. 2023, doi: 10.1016/j.compbiomed.2022.106296.

[11] F. Hržić, I. Štajduhar, S. Tschauner, E. Sorantin, and J. Lerga, “Local-entropy based approach for X-ray image segmentation and fracture detection,” *Entropy*, vol. 21, no. 4, Apr. 2019, doi: 10.3390/e21040338.

[12] “Magnetic Resonance Imaging (MRI) of the Brain and Spine: Basics,” 2020. https://case.edu/med/neurology/NR/MRI%20Basics.htm (accessed Oct. 23, 2022).

[13] Furqan Shaukat, Gulistan Raja, Ali Gooya, and Alejandro F. Frangi, “Fully automatic detection of lung nodules in CT images using a hybrid feature set,” Apr. 2017, Accessed: Feb. 17, 2023. [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/10.1002/mp.12273

[14] Lucas O. Teixeira, Rodolfo M. Pereira, and Diego Bertolini, “Impact of Lung Segmentation on the Diagnosis and Explanation of COVID-19 in Chest X-ray Images,” Oct. 2021, Accessed: Feb. 17, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/21/21/7116

[15] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.4038

[16] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015. doi: 10.1007/978-3-319-24574-4_28.

[17] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid Scene Parsing Network,” Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.01105

[18] A. Chaurasia and E. Culurciello, “LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation,” Jun. 2017, doi: 10.1109/VCIP.2017.8305148.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation,” Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.02611

[20] N. Siddique, P. Sidike, C. Elkin, and V. Devabhaktuni, “U-Net and its variants for medical image segmentation: theory and applications.”

[21] Brien Posey, “What is edge computing? Everything you need to know,” *TechTarget*, Oct. 2020. https://www.techtarget.com/searchnetworking/definition/edge-device#:~:text=An%20edge%20device%20is%20any,%2D%2D%20or%20exit%20%2D%2D%20points. (accessed Mar. 19, 2023).

[22] BILL BITHER, “WHAT IS AN EDGE DEVICE AND WHY IS IT ESSENTIAL FOR IOT?,” *machinemetrics*, Jan. 07, 2021. https://www.machinemetrics.com/blog/edge-devices#:~:text=There%20are%20two%20types%20of,data%20source%20for%20industrial%20automation. (accessed Mar. 19, 2023).

[23] “What is Edge AI?,” *CDVIAN*. https://www.advian.fi/en/what-is-edge-ai#:~:text=Edge%20AI%20devices%20include%20smart,cameras%20that%20use%20video%20analytics. (accessed Mar. 19, 2023).

[24] Vanessa Braunstein, “How Edge Computing is Transforming Healthcare,” *Nvidia Developer*, Oct. 26, 2021.

[25] Naif Almusallam, Abdulatif Alabdulatif, and Fawaz Alarfaj, “Analysis of Privacy-Preserving Edge Computing and Internet of Things Models in Healthcare Domain”, Accessed: Feb. 17, 2023. [Online]. Available: https://www.hindawi.com/journals/cmmm/2021/6834800/

[26] Md. Golam Rabiul Alam, Md. Shirajum Munir, Md. Zia Uddin, Mohammed Shamsul Alam, Tri Nguyen Dang, and Choong Seon Hong, “Edge-of-things computing framework for cost-effective provisioning of healthcare data,” *J Parallel Distrib Comput*, pp. 54–60, Jan. 2019.

[27] “How Edge Computing Is Driving Advancements in Healthcare Analytics,” *Intel*. https://www.intel.com/content/www/us/en/healthcare-it/edge-analytics.html (accessed Feb. 17, 2023).

[28] Todd J. Traver, “5 considerations for reliable edge computing in the age of IoT,” Jan. 04, 2018. https://www.datacenterdynamics.com/en/opinions/5-considerations-for-reliable-edge-computing-in-the-age-of-iot/ (accessed Feb. 17, 2023).

[29] F.-X. W. Sakib Mostafa, *Chapter 3 -Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images*. 2021.

[30] Harshall Lamba, “Understanding Semantic Segmentation with UNET,” *Medium*, Feb. 17, 2019. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47#:~:text=Transposed%20convolution%20(sometimes%20also%20called,an%20image%20with%20learnable%20parameters (accessed Mar. 07, 2023).

[31] Swapna K E, “Convolutional Neural Network,” *Developers Breach*. https://developersbreach.com/convolution-neural-network-deep-learning/ (accessed Apr. 06, 2023).

[32] Yugesh Verma, “Guide to Different Padding Methods for CNN Models,” Sep. 04, 2021. https://analyticsindiamag.com/guide-to-different-padding-methods-for-cnn-models/ (accessed Mar. 07, 2023).

[33] Jason Brownlee, “A Gentle Introduction to the Rectified Linear Unit (ReLU),” *Machine Learning Mastery*, Aug. 20, 2020.

[34] Divyanshu Mishra, “Transposed Convolution Demystified,” *Towards Data Science*, Mar. 10, 2020.

[35] “What is Transposed Convolutional Layer?,” *GeeksforGeeks*, Jan. 24, 2023. https://www.geeksforgeeks.org/what-is-transposed-convolutional-layer/ (accessed Mar. 07, 2023).

[36] “Confusion matrix,” *The Science of Machine Learning*. https://www.ml-science.com/confusion-matrix (accessed Apr. 06, 2023).

[37] Ekin Tiu, “Metrics to Evaluate your Semantic Segmentation Model,” Aug. 10, 2019.

[38] Kukil, “Intersection over Union (IoU) in Object Detection & Segmentation,” *LearnOpenCV*, Jun. 28, 2022. https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/ (accessed Apr. 06, 2023).

[39] Nigel M. Parsad, “Deep Learning in Medical Imaging V,” *Medium*, Jun. 14, 2018. https://medium.datadriveninvestor.com/deep-learning-in-medical-imaging-3c1008431aaf (accessed Apr. 06, 2023).

[40] “Precision and recall,” *Wikipedia*. https://en.wikipedia.org/wiki/Precision_and_recall#:~:text=Precision%20(also%20called%20positive%20predictive,are%20therefore%20based%20on%20relevance. (accessed Apr. 06, 2023).

[41] “Max Pooling,” *paperswithcode*. https://paperswithcode.com/method/max-pooling (accessed Apr. 06, 2023).

[42] “Average Pooling,” *paperswithcode*. https://paperswithcode.com/method/average-pooling#:~:text=Average%20Pooling%20is%20a%20pooling,used%20after%20a%20convolutional%20layer. (accessed Apr. 06, 2023).

[43] Rafay Qayyum, “Introduction To Pooling Layers In CNN,” Aug. 16, 2022. https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn (accessed Mar. 07, 2023).

[44] Z. Tong and G. Tanaka, “Hybrid pooling for enhancement of generalization ability in deep convolutional neural networks,” *Neurocomputing*, vol. 333, pp. 76–85, Mar. 2019, doi: 10.1016/j.neucom.2018.12.036.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “LNCS 8691 - Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” 2014. [Online]. Available: http://arxiv.org/abs/1406.4729v1.

[46] Jason Brownlee, “Softmax Activation Function with Python,” *Machine Learning Mastery*, Oct. 19, 2020.

[47] Jason Brownlee, “A Gentle Introduction To Sigmoid Function,” *Machine Learning Mastery*, Aug. 25, 2021. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/ (accessed Mar. 10, 2023).

[48] Deepak Battini, “Implementing Drop Out Regularization in Neural Networks,” *Tech-Quantum*, Nov. 03, 2018. https://www.tech-quantum.com/implementing-drop-out-regularization-in-neural-networks/ (accessed Apr. 06, 2023).

[49] Jason Brownlee, “A Gentle Introduction to Dropout for Regularizing Deep Neural Networks,” *Machine Learning Mastery*, Aug. 06, 2019. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (accessed Mar. 10, 2023).

[50] Ramazan Gençay, “Early Stopping,” *Papers With Code*. https://paperswithcode.com/method/early-stopping (accessed Apr. 06, 2023).

[51] Jason Brownlee, “A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks,” *Machine Learning Mastery*, Aug. 06, 2019. https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/ (accessed Mar. 12, 2023).

[52] “A Beginner’s Guide to Focal Loss in Object Detection!,” *Analytics Vidhya*, Aug. 28, 2020. https://www.analyticsvidhya.com/blog/2020/08/a-beginners-guide-to-focal-loss-in-object-detection/ (accessed Mar. 01, 2023).

[53] K. Doi and A. Iwasaki, “The effect of focal loss in semantic segmentation of high resolution aerial image,” in *International Geoscience and Remote Sensing Symposium (IGARSS)*, Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 6919–6922. doi: 10.1109/IGARSS.2018.8519409.

[54] B. Murugesan, K. Sarveswaran, V. S. Raghavan, S. M. Shankaranarayana, K. Ram, and M. Sivaprakasam, “A Context Based Deep Learning Approach for Unbalanced Medical Image Segmentation,” in *Proceedings - International Symposium on Biomedical Imaging*, IEEE Computer Society, Apr. 2020, pp. 1949–1953. doi: 10.1109/ISBI45749.2020.9098597.

[55] Hamid Rezatofighi and Hamid Rezatofighi, “Generalized Intersection over Union A Metric and A Loss for Bounding Box Regression,” *Stanford*, 2018.

[56] Vitaly Bushaev, “Adam — latest trends in deep learning optimization.,” *TowardsDataScience*, Oct. 22, 2018.

[57] Jason Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,” *Machine Learning Mastery*, Jan. 13, 2021.

[58] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[59] “Number of training parameters in millions(M) for VGG, ResNet and DenseNet models.,” *ResearchGate*. https://www.researchgate.net/figure/Number-of-training-parameters-in-millionsM-for-VGG-ResNet-and-DenseNet-models_tbl1_338552250 (accessed Apr. 07, 2023).

[60] D. Lewy and J. Mańdziuk, “An overview of mixing augmentation methods and augmentation strategies,” *Artif Intell Rev*, Mar. 2022, doi: 10.1007/s10462-022-10227-z.

[61] Pragya Soni, “Data augmentation: Techniques, Benefits and Applications,” *analytic steps*, Jan. 09, 2022. https://www.analyticssteps.com/blogs/data-augmentation-techniques-benefits-and-applications (accessed Mar. 21, 2023).

[62] Jeff Hale, “Scale, Standardize, or Normalize with Scikit-Learn,” Mar. 04, 2019. https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02 (accessed Feb. 25, 2023).

[63] Jason Brownlee, “How to Use StandardScaler and MinMaxScaler Transforms in Python,” Aug. 28, 2020. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ (accessed Feb. 25, 2023).

[64] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information (Switzerland)*, vol. 11, no. 2, Feb. 2020, doi: 10.3390/info11020125.

[65] “Convert TensorFlow models,” *Tensorflow*, Jun. 11, 2022. https://www.tensorflow.org/lite/models/convert/convert_models (accessed Apr. 07, 2023).

[66] “Get started with the Dev Board Mini,” *Coral*. https://coral.ai/docs/dev-board-mini/get-started/#7-run-a-model-using-the-pycoral-api (accessed Apr. 07, 2023).

[67] “deeplabv3plusLayers.” https://uk.mathworks.com/help/vision/ref/deeplabv3pluslayers.html (accessed Feb. 26, 2023).

[68] “Matlab Algorithm Documentation.” https://uk.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu80qkw-3_head (accessed Feb. 26, 2023).

[69] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, “Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study,” *Neural Computing and Applications*, vol. 34, no. 7. Springer Science and Business Media Deutschland GmbH, pp. 5321–5347, Apr. 01, 2022. doi: 10.1007/s00521-022-06953-8.

[70] “Max pooling and different Stochastic pooling approaches,” *ResearchGate*. https://www.researchgate.net/figure/Max-pooling-and-different-Stochastic-pooling-approaches-a-the-standard-max-pooling-b_fig2_358274210 (accessed Apr. 07, 2023).

[71] S. Ahuja, B. K. Panigrahi, and T. K. Gandhi, “Fully automatic brain tumor segmentation using DeepLabv3+ with variable loss functions,” in *Proceedings of the 8th International Conference on Signal Processing and Integrated Networks, SPIN 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 522–526. doi: 10.1109/SPIN52536.2021.9566128.

[72] G. Ramasamy, T. Singh, and X. Yuan, “Multi-Modal Semantic Segmentation Model using Encoder Based Link-Net Architecture for BraTS 2020 Challenge,” *Procedia Comput Sci*, vol. 218, pp. 732–740, 2023, doi: 10.1016/j.procs.2023.01.053.

[73] H. Phan, M. Krawczyk-Becker, T. Gerkmann, and A. Mertins, “DNN and CNN with Weighted and Multi-task Loss Functions for Audio Event Detection,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.03211

[74] Aqeel Anwar, “Types of Regularization in Machine Learning,” *Medium*, Feb. 04, 2021. https://towardsdatascience.com/types-of-regularization-in-machine-learning-eb5ce5f9bf50 (accessed Mar. 09, 2023).

[75] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” 2014.

[76] S. Arvind, J. V. Tembhurne, T. Diwan, and P. Sahare, “Improvised light weight deep CNN based U-Net for the semantic segmentation of lungs from chest X-rays,” *Results in Engineering*, vol. 17, Mar. 2023, doi: 10.1016/j.rineng.2023.100929.

[77] Y. Xu *et al.*, *U-Net with optimal thresholding for small blob detection in medical images*.

[78] B. V. Isunuri and J. Kakarla, “Fast brain tumour segmentation using optimized U-Net and adaptive thresholding,” *Automatika*, vol. 61, no. 3, pp. 352–360, Jul. 2020, doi: 10.1080/00051144.2020.1760590.

[79] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Feb. 2015, [Online]. Available: http://arxiv.org/abs/1502.03167

[80] Prateek Chhikara, “Understanding Morphological Image Processing and Its Operations,” *Medium*, Mar. 20, 2022. https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756 (accessed Mar. 12, 2023).

[81] R. Setyawan, A. Almahfud, C. A. Sari, D. Rosal, I. M. Setiadi, and H. Rachmawanto, “MRI Image Segmentation using Morphological Enhancement and Noise Removal based on Fuzzy C-means; MRI Image Segmentation using Morphological Enhancement and Noise Removal based on Fuzzy C-means,” 2018.

[82] Rohit Kundu, “The Complete Guide to Ensemble Learning,” *V7*, Mar. 02, 2023. https://www.v7labs.com/blog/ensemble-learning (accessed Mar. 09, 2023).

[83] Y. ZhiWei, Y. Yu, and X. Xiao, “Image segmentation based on ensemble learning,” in *CCTAE 2010 - 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, 2010, pp. 423–427. doi: 10.1109/CCTAE.2010.5543712.

[84] “Performance graph : with and without Transfer Learning.,” *ResearchGate*. https://www.researchgate.net/figure/Performance-graph-with-and-without-Transfer-Learning_fig2_345904103 (accessed Apr. 07, 2023).

[85] Ivana Zadro Matovinovic, Sven Loncaric, Julian Lo, Morgan Heisler, and Marinko Sarunic, *Transfer Learning with U-Net type model for Automatic Segmentation of Three Retinal Layers In Optical Coherence Tomography Images*.

[86] Vaibhav Nandwani, “Inside Quantization Aware Training,” *Medium*, May 26, 2021. https://towardsdatascience.com/inside-quantization-aware-training-4f91c8837ead (accessed Mar. 22, 2023).

[87] Sayak Paul, “A Tale of Model Quantization in TF Lite,” *Weights & Biases*, Oct. 11, 2022. https://wandb.ai/sayakpaul/tale-of-quantization/reports/A-Tale-of-Model-Quantization-in-TF-Lite--Vmlldzo5MzQwMA (accessed Mar. 22, 2023).

[88] S. Piao and J. Liu, “Accuracy Improvement of UNet Based on Dilated Convolution,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Nov. 2019. doi: 10.1088/1742-6596/1345/5/052066.

7.0 Appendix

7.1 Gantt Chart

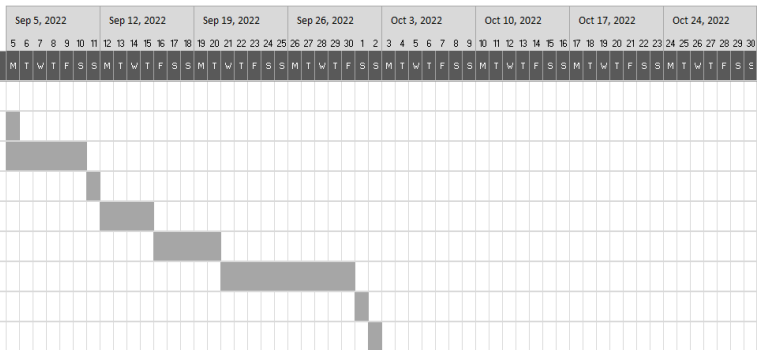
Medical Segmentation for Edge Devices

Tham Shung Yan

Project Start: Mon, 9/5/2022

Display Week: 1

TASK	PROGRESS	START	END
Milestone 1 - Objective 1			
Meet with supervisor to discuss project outline	100%	9/5/22	9/5/22
Find, Study and Decide Core Journal	100%	9/5/22	9/10/22
Study the difference between NCS2 and TPU	100%	9/11/22	9/11/22
Implement segmentation pre-built U-Net using Camvid d	100%	9/12/22	9/15/22
Find and study medical datasets	100%	9/16/22	9/20/22
Study and modify U-Net Structures	100%	9/21/22	9/30/22
Request Quotation for Coral Edge TPU	100%	1/10/2022	1/10/2022
Write out Project Specification	100%	2/10/2022	2/10/2022



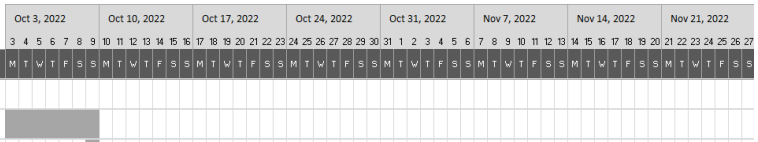
Medical Segmentation for Edge Devices

Tham Shung Yan

Project Start: Mon, 9/5/2022

Display Week: 5

TASK	PROGRESS	START	END
Milestone 2 - Objective 2 & 3 & 4			
BraT dataset augmentation	100%	10/3/22	10/9/22
Study loss function and metrics in segmentation	100%	10/9/22	10/9/22
Project Outline and Planing Report Wrtiting	100%	10/10/22	10/16/22
Review and Improve Project Outline	100%	10/17/22	10/21/22
Train and evaluate 3D U-Net using different pooling layer	100%	10/22/22	10/30/22
Meet with project moderator	100%	10/31/22	10/31/22



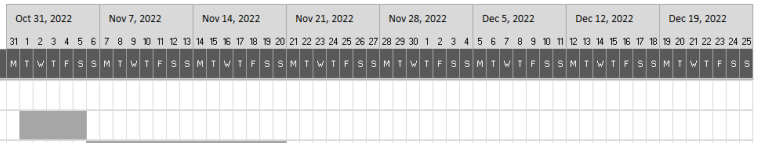
Medical Segmentation for Edge Devices

Tham Shung Yan

Project Start: Mon, 9/5/2022

Display Week: 9

TASK	PROGRESS	START	END
Milestone 3 - Objective 2 & 3 & 4			
Lung Segmentation dataset research and augmentation	100%	11/1/22	11/5/22
Research and evaluate 2D U-Net with different pooling la	100%	11/6/22	11/20/22
Train and evaluate 2D U-Net with augmented lung segm	100%	11/21/22	11/30/22



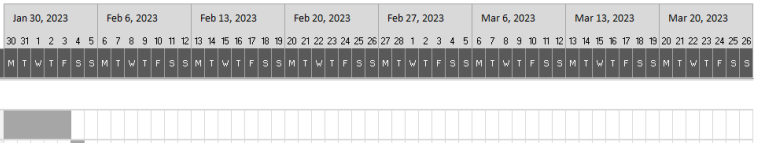
Medical Segmentation for Edge Devices

Tham Shung Yan

Project Start: Mon, 9/5/2022

Display Week: 22

TASK	PROGRESS	START	END
Milestone 4 - Thesis Writing -Objective 5, 6, 7			
Train and test Deeplab and make comparison	100%	1/30/23	2/3/23
Convert trained 2D U-Net Model to tflite version	100%	2/4/23	2/4/23
Implement and Evaluate 2D U-Net on Coral Dev Board M	100%	2/5/23	2/14/23
Thesis Methodology Writing	100%	2/15/23	2/28/23
Thesis Result and Discussion Writing	100%	3/1/23	3/10/23
Thesis Literature Review Writing	100%	3/10/23	3/16/23
Thesis Introduction and Conclusion Writing	100%	3/17/23	3/20/23



Medical Segmentation for Edge Devices

Tham Shung Yan

Project Start: Mon, 9/5/2022

Display Week: 29

TASK	PROGRESS	START	END
Milestone 5 - Final Thesis review and Presentation, Roadshow, Viva preparation			
Final Thesis Submission Review and Improvement	100%	3/21/23	4/13/23
Assesor Presentation Preparation	100%	4/14/23	4/19/23
Moderator Presentation Preparation	100%	4/20/23	4/25/23