School of Computing

FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Deep Learning for Aneurysm Segmentation from Magnetic Resonance Angiogram Images

Bhumit Vijay Adivarekar

Submitted in accordance with the requirements for the degree of M.Sc. Advanced Computer Science (Artificial Intelligence)

2021/2022

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Deliverables 1	*Report*	*SSO (15/10/2021)*
Deliverable 2	*Software codes or URL*	*Supervisor, assessor (15/10/2021)* **URL contained in the report*

Type of Project: __Exploratory Software__

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

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

(Signature of student) ____________________________

Summary

This project aims to develop a 3D medical image segmentation model which can detect aneurysms in a 3D brain magnetic resonance angiogram (MRA) image. An aneurysm is the formation of blood clotting in brain vessels, detecting them manually is a labour-intensive and time-consuming task. Recent advances in the field of computer vision and deep learning introduced some state-of-the-art models which perform better and faster than human experts for 3D medical images. This project discusses the latest trends in this field and medical imaging domain and delivers an exploratory software, where a 3D MRA image is given as input, pre-processed, and passed through the 3D image segmentation models to detect aneurysm and display the aneurysm segmented region to the end-user.

Acknowledgements

I would like to thank my supervisor, Dr. Toni Lassila, for his guidance throughout the project and for helping me grasp complex medical imaging domain-related concepts with ease. I would also like to thank Prof. Ali Gooya for giving me valuable feedback on my project which helped me achieve my goals.

Table of Contents

Chapter 1
Introduction

1.1 Context

A Cerebral Aneurysm or generally called an Aneurysm is a bulge in a blood vessel in the human brain, which can cause serious consequences if it bursts. Aneurysms are present in about 5% of the population with high morbidity and mortality [1]. Around 1 in 12,500 people have a ruptured brain aneurysm in England each year, the aneurysm can be developed by anyone at any age, but it's most common in people over the age of 40 [2]. Currently, an aneurysm is detected by using medical imaging tool called Magnetic Resonance Angiogram (MRA), and then doctors check this MRA image manually to detect an aneurysm, which is a time-consuming, complex, and error-prone task even if the doctor is experienced. An automatic process can significantly make the whole medical diagnosis process efficient.

1.2 Problem Statement

An aneurysm is generally detected using MRA images and manually segmenting the region of the aneurysm. MRA is a non-invasive medical imaging technique used to detect various brain-related diseases. As manually detecting them is a complex and time-consuming task, an accurate automatic segmentation method can prove beneficial for this problem. Such methods can make this manual process more efficient at expense of computing power. Detecting an aneurysm at an early stage is a very difficult task even for experts, advances in medical imaging and deep learning can help solve this problem. This problem could be treated as an Image Segmentation problem as we are trying to segment the aneurysm part from the MRA image. These methods are used to locate objects and boundaries in an image.

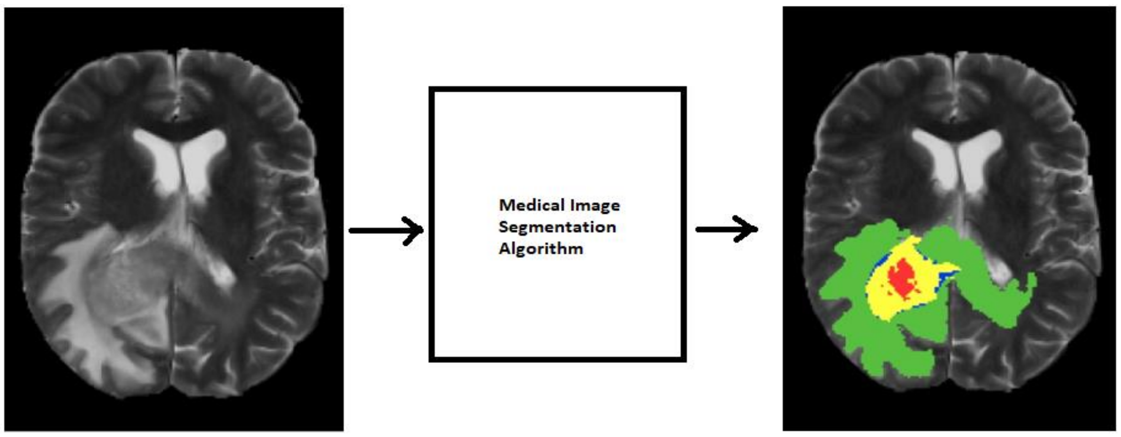

Figure 1. Working of 2D Medical Segmentation Algorithms to segment abnormalities in Brain Medical Image.

A simple working of the medical segmentation approach can be seen in Figure 1. An image of the brain is passed through a segmentation model to get a segmentation mask which can be overlapped on the source image to get the region of abnormalities. To train or teach such models we need an ample amount of labelled data where abnormalities masks are provided, such dataset is created by humans experts. In the case of brain images, they form a complex structure of vessels, and an aneurysm is formed in between these vessels so a 2D image fails to represent this complexity which can lead to inaccurate segmentation, so for our problem, we will be working with 3D images and 3D neural networks.

1.3 Objectives

We have laid down few objectives to achieve our goal of segmenting an aneurysm.

1.3.1 Gaining access to ADAM Challenge Dataset.

One of the objectives is to gain access to this dataset as this is only provided upon successful participation in this competition. This dataset includes 113 images with a total memory size of 33.4 GB, which can be a challenging task to access this dataset.

1.3.2 Literature Review.

This domain of medical imaging is very vast and can be difficult for any person with no domain information. We will be reviewing existing literature on this topic of aneurysm segmentation for any standard or latest existing approaches.

1.3.3 Building and Training Models for the Given Problem.

After doing an extensive review of existing approaches, we will be deciding which deep learning-based segmentation algorithm we will be using.

1.3.4 Evaluation of model on test data.

We will be evaluating the performance of the model based on a suitable metric for a segmentation task like the Dice Similarity Coefficient / Dice Score.

1.4 Deliverables

We will be delivering the complete software code for pre-processing the data, building the model, and all the trained models for segmentation.

1.5 Ethical Issues and Risks Involved

We will be using a dataset that is only made available through participation in the competition, so we are bound by some rules of the competition. We can use this dataset for personal use but cannot distribute this dataset to a third party unless they are added to the competition team.

There is a risk involved in this project as we are dealing with 3D images and 3D neural networks which require high computing power and building and training them is a time-consuming task. If these conditions are not met, then we might not get an accurate model for this task as we will be training these models for less amount of time and with low computing power.

Chapter 2
Background and Literature Review

2.1 Machine Learning

Machine learning is a branch of Artificial Intelligence and Computer Science which deals with a set of algorithms and vast data to imitate the way a human learns, eventually becoming better and better.

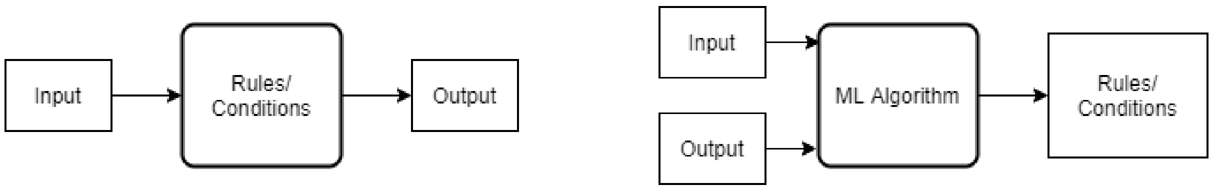

Figure 2. Conventional Programming Approach (left) vs Machine Learning Approach (right)

Normal conventional algorithmic approaches focus on rules and conditions beforehand, but what if we do not have any specific rules and we have a vast amount of data instead, here machine learning algorithms help us to tackle this problem. Here we need to create a machine learning pipeline where we provide inputs and output data and train the machine learning algorithm which will provide a model or in layman terms a set of rules and conditions. This model can be used to get outputs for unknown data inputs. Figure 2 represents a comparison of the conventional approach and machine learning approach. Machine Learning (ML) algorithms can be used to find unknown patterns in the data and draw insights from it. ML algorithms generally can be further divided into two classes of algorithms, Supervised and Unsupervised Learning Algorithms.

2.1.1 Supervised Learning

Supervised Learning uses ML algorithms where labelled data is present i.e., for every input there is a labelled output. For instance, in an image classification problem of dog vs cat images for every image, there would be a label associated with it, representing that image class or category. These algorithms try to map the representation of the images to their respective labels. If this problem were to be solved using a conventional programming approach, we need to write a lot of rules to identify shapes and colour intensities, and various image features, but it’s impossible to write the rule for every feature as there are a lot of variations in dog and cats, for example, different breeds, lighting conditions, etc. Supervised ML approaches make problems like these easy to handle with enough labelled data. In general, there are two major tasks in Supervised ML: classification and regression. Classification is identifying a particular category based on input data. Regression is forecasting values based on certain input parameters.

2.1.2 Unsupervised Learning

Unsupervised Learning uses a set of ML algorithms to analyse and cluster unlabelled datasets. The main purpose of these algorithms is to discover unknown hidden patterns in the data without any human intervention, thus they are called unsupervised. For instance, we have a dataset of food orders around the city, and the shop wants to open new branches in the city, so by using clustering on this dataset of customers addresses or locations, we can cluster them into how many branches the shop want to open, with this they shop can choose an appropriate location for the new branch to increase their revenue. Unsupervised learning includes two main tasks: clustering and association. Clustering is the grouping of unlabelled data based on similarities or differences. Association is finding patterns or relationships between different attributes in a dataset. E.g., recommendations of products based on buying habits of customers.

2.1.3 Deep Learning

Deep learning is the subset of machine learning which generally consists of a neural network with multiple layers. Neural networks use multiple hidden layers to map the input to the output with non-linear representations.

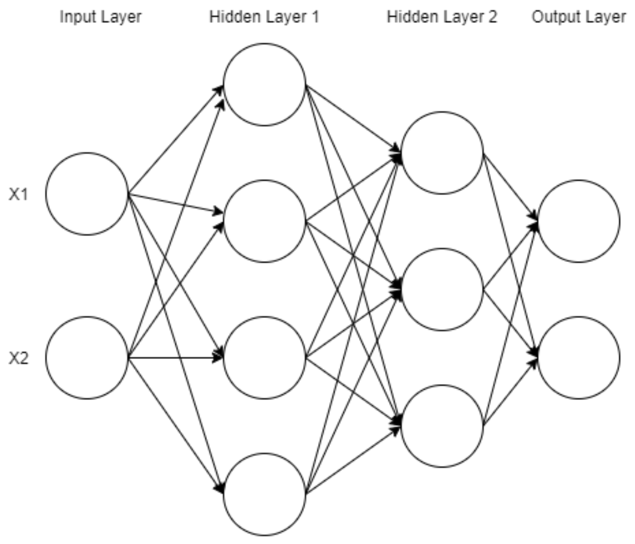

Figure 3. The simple network structure of Deep Neural Network

Figure 3 represents a simple Artificial Deep Neural Network with 2 hidden layers. The main purpose of neural networks is to simulate brain-like behaviour to match its learning behaviour. Deep learning can be expanded to other conventional machine learning techniques like supervised, unsupervised, and reinforcement learning. Deep learning has outperformed Machine learning approaches and shown many successes in other domain-specific tasks in Computer Vision, Natural Language Understanding, Speech Recognition. Many latest AI applications use deep neural networks e.g., google translate, google search, amazon product recommendation, self-driving cars, and numerous other products.

2.2 Convolution Neural Networks

Convolution Neural Networks (CNN) was first introduced by Yann LeCunn [3] to detect objects in a document. Later, CNN was adopted for various tasks, like image recognition, object detection, and various computer vision-related tasks. The underlying convolution operation is the key to its success. An image is made up of pixels which is a matrix, to extract features from an image, a convolution operation is used with different filters.

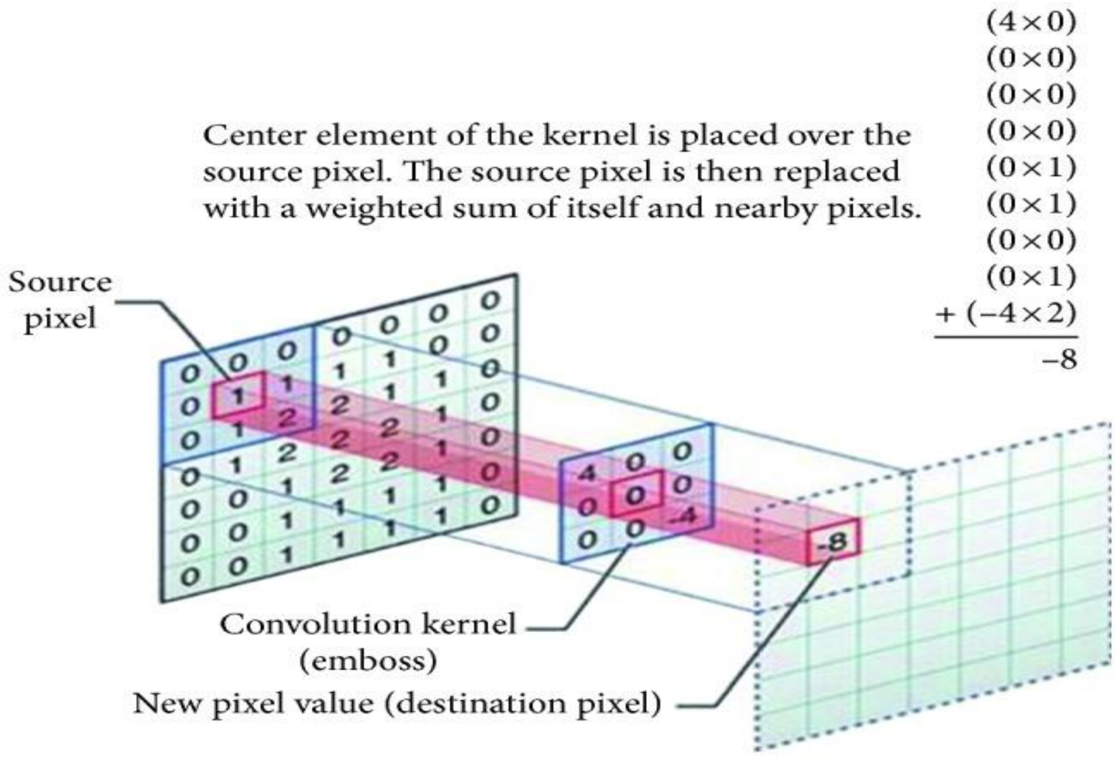
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4×0)
(0×0)
(0×0)
(0×0)
(0×1)
(0×1)
(0×0)
(0×1)
+ (−4×2)
−8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Figure 4. Example of convolution operation on image matrix with filters (kernel)

In CNN, the convolution operations are performed on the image to extract its features with many different kernel/filters [4]. Figure 4 represents how convolution operation works on the image matrix taken from [5]. CNN layers mostly work as feature extractors, initial CNN layers extract shapes, edges, and global information while deeper layers extract many lower detail features [4]. Max pooling layer is also added after convolution to reduce the dimension and select important features. Max pooling selects the feature with max value from the matrix and passes it to the next layers. In the end, it is combined with an ANN to classify images from the representations learned.

2.3 U-Net Architecture

In recent years, deep convolutional networks have outperformed many visual recognition tasks. In 2015 O. Ronneberger et al released a novel architecture for medical image segmentation called U-Net based on CNN [6]. Figure 5 shows the standard U-Net Architecture taken from [6].

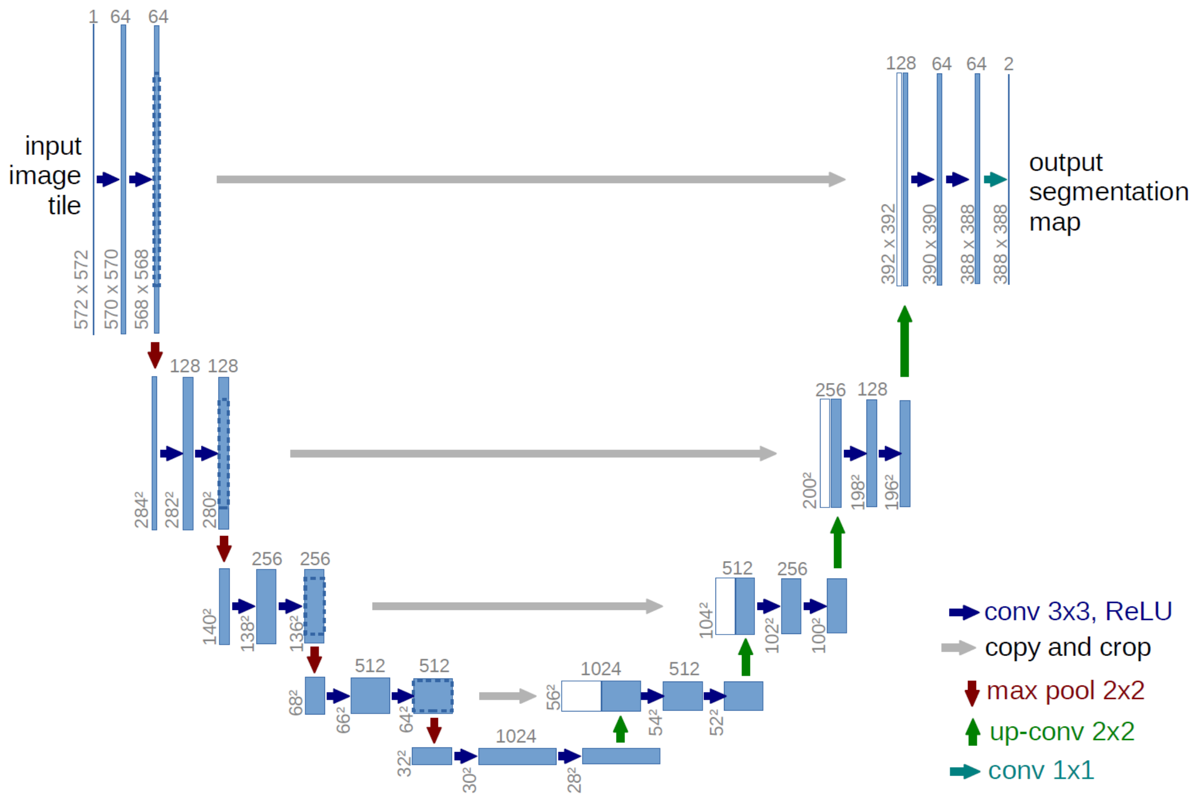

Figure 5. Standard U-Net Architecture with contracting (left) and expansive (right) path

This architecture is called U-Net because of the way it's built and shown in figure 5. It consists of two paths contracting path (left) and expansive path (right) with skip connection shown in grey colour in the above figure 5. The input image is downscaled with subsequent convolution and max-pooling layers that's why the left side is called contracting path and the features are again upsampled on the right side i.e., on the expansive path to get the segmentation mask. Here while expansion the output from convolution layers of the left side is concatenated while upsampling by skip connections. Up-sampling is the reverse of the max-pooling operation.

U-Net was one of the first deep neural networks used to perform medical image segmentation. After this breakthrough, this architecture was adopted for various other image segmentation tasks in other domains. Modifications of U-Net are still widely used in medical image segmentation with few improvements and modifications in network architectures. The standard U-Net was used for 2D image segmentation, later it was adopted for 3D medical images, which also yielded state-of-the-art results in medical image segmentation tasks.

2.4 Visual Attention

Visual Attention is a part of our everyday life that even we do it subconsciously sometimes or do with purpose. For instance, focusing on the road while driving, focusing on the text while reading, attention is what we do most of the time. But defining it in a mathematical way or in terms of computer science or while training a deep neural network is quite challenging. For neural networks especially, it needs to learn, for example, the most important place to look at a picture is to focus on objects rather than the background, so attention is needed to solve this issue.

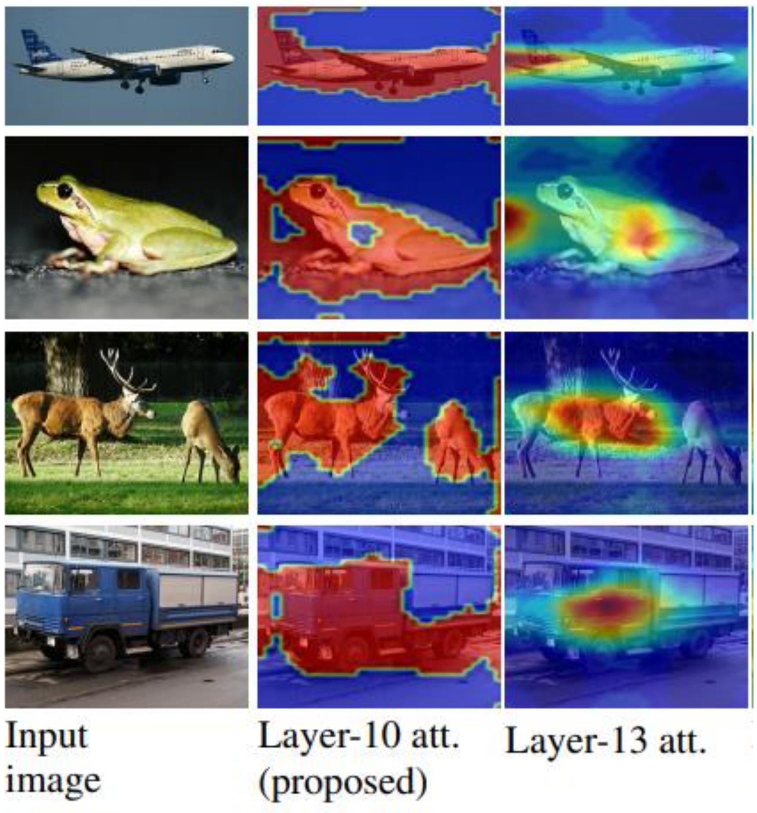

Figure 6. Attention on images on different layers.

Attention mechanism was first introduced by Vaswani et al in 2017 [7] in his research paper. It was first applied in Natural Language Processing task for Neural Machine Translation which greatly improved the results for translations. Later this research was adopted in the domain of computer vision. In computers, visual attention is used over spatial domain i.e., all over the image while in NLP it is used over sequence or time domain. Figure 6 represents how attentions maps look over the image in [8]. In [8] Jetley et al. trained a visual attention model consisting of convolution layers. Using attention for images greatly improves the accuracy of the model but has a huge trade-off as it uses a lot of matrix operation which demands high memory and good hardware.

In computational terms, attention is applied by multiplying all neighbouring values and weighing them against the labels and previous layer representations. On contrary, using an attention mechanism is hard on images as there are two dimensions to deal with rather than one dimension in NLP-like tasks. Figure 7 displays how attention is computed visually for videos over both spatial and time domains from Non-Local Neural Networks [9]. The 3D images can be treated as a similar problem as of [9] videos-based attention. The depth (z-axis) of the 3D images can be considered as time domain and x and y-axis as spatial domain. This approach though demands a lot of computation resources and it's not feasible for our problem.

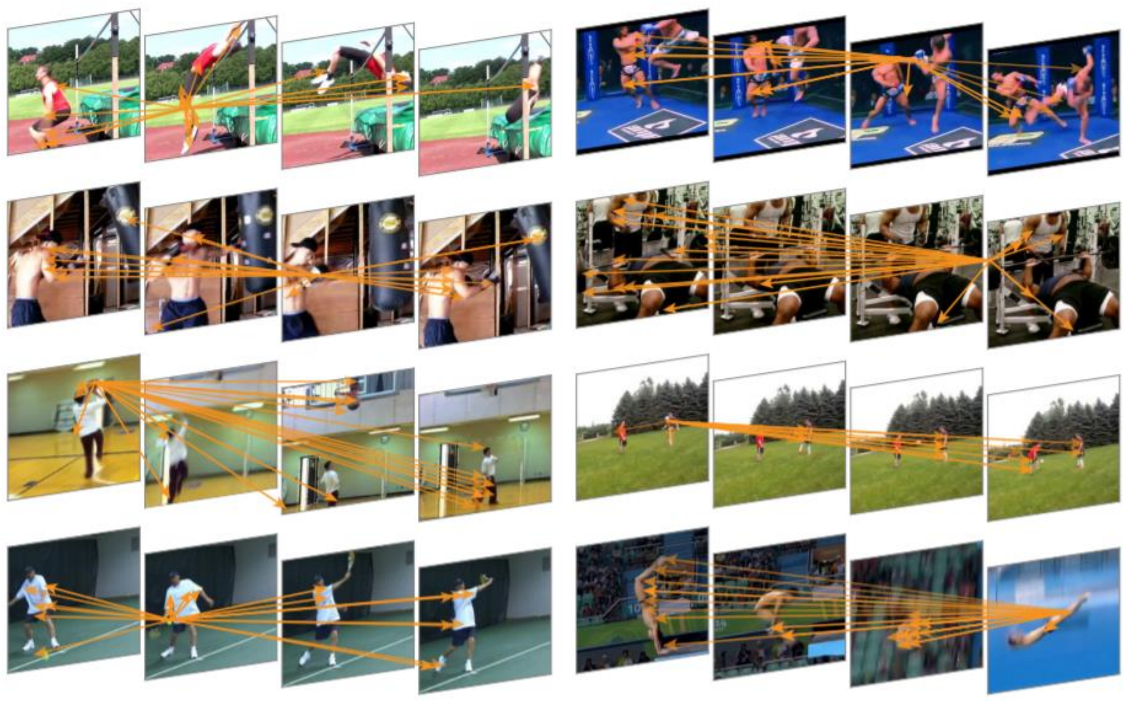

Figure 7. Attention on images over spatial and time domain

For our problem statement, we need an efficient attention mechanism to focus more on the aneurysm remnant in the MRA image. As we are dealing with 3D images, and the aneurysm is very small and hard to find it visually, we need to use an attention mechanism to attend more toward the complex arteries formation in the brain images and find the small aneurysm remnant from this complex image.

2.5 Conventional Approach for Medical Image Segmentation

Image segmentation is one of the most important parts of medical image processing. It is a process of extracting the region of interest (ROI) through an automatic or semi-automatic process. In medical imagining, image segmentation is used to segment tissues, body organs, and abnormalities in them. Some of the applications consist of cancer detection, analysis of cardiac images, tumour detection and segmentation, etc. In earlier days medical images were being segmented using rules and condition-based methods. [10] describes all of these methods in detail, we will discuss some of them as follows.

2.5.1 Region-based Methods

A region is composed of pixels with some distinct values and its neighbours, and the boundary is made from differences between two regions. Most methods are based on region and boundary properties as they are visually distinguishable. One of the most common region-based methods is Thresholding. Thresholding is the simplest and fastest segmentation method based on a simple assumption that images are formed from regions with different grey levels. With this threshold value, the image is divided into different parts, this threshold value is obtained by analysing the histogram of the pixel intensities. After obtaining the histogram then a threshold can be set to divide regions of the image i.e., foreground and background. Figure 8 showcases how the threshold is set by analysing the histogram, the dotted vertical line represents the threshold to be set to split foreground and background.

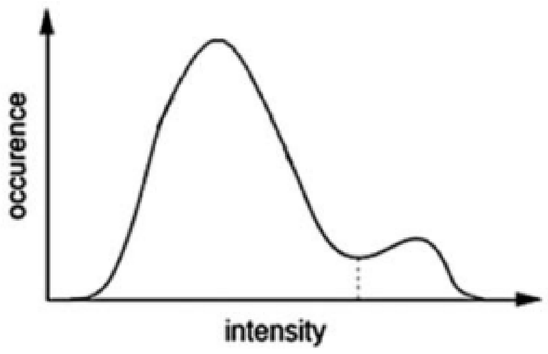

Figure 8. Histogram of intensities for setting the threshold

For coloured images, the image is converted to grayscale first and then thresholding is applied. For our 3D medical images, these methods hardly work as there is no clear boundary between aneurysm and blood vessels, so in such a scenario thresholding fails to show good results.

2.5.2 Classification Methods

These methods try to classify each pixel into a category, the only requirement is for target labels for the input images, i.e., target segmentation masks. K-Nearest Neighbour is the most common method used for image segmentation. This algorithm looks at its neighbour to obtain a segmentation mask and calculates the distance and tries to map it to the target mask. The challenge in this method is to select an optimal number of neighbours and type of distance function. These conventional ML algorithms fail as it only looks at nearby neighbours and does not consider the whole spatial information.

2.5.3 Clustering Methods

The operation of clustering methods is similar to classification methods, but the only difference is they do not require training data. K-Means Clustering has a widely used algorithm for image clustering, it selects the most prominent colours from the image, and based on the number of clusters we specify, it divides the image into these number of clusters. Figure 9 displays how the number of clusters (k) affect the segmentation results [10].

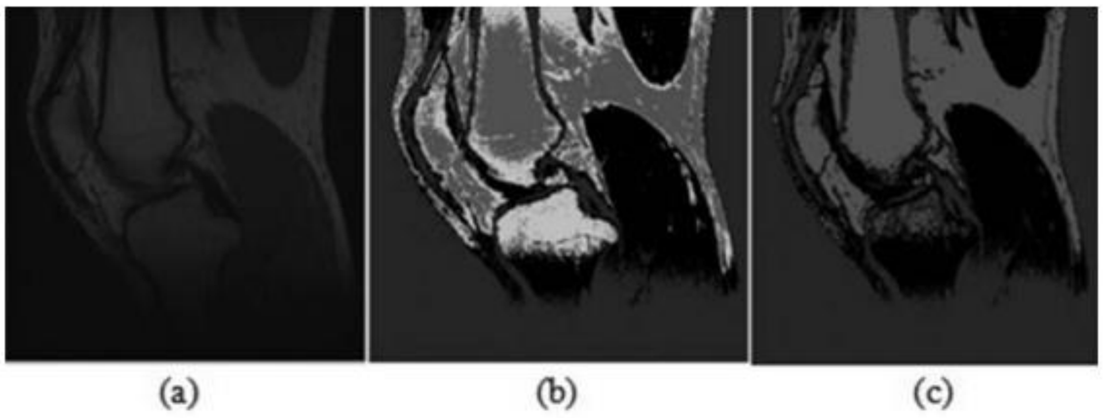

Figure 9. K-Means clustering results (a) original Knee MRI image (b) when k=4 clusters (c) when k=3 clusters

As seen in figure 9 clusters determine the image colour segments, (b) shows more colour clusters while (c) loses some information regarding the centre part of knee bone. This method fails for 3D segmentation as it’s the aneurysm that gets blended with the colour distribution of arteries which makes it difficult to distinguish.

2.6 Deep Learning-based 2D Approach

2D image segmentation tasks on different types of standard and medical imaging datasets have shown great success since the advent of U-Net [6] in 2015. But as simple as this approach was for 2D images, it was as hard for 3D images. 3D images include one more dimension of complexity around the z-axis i.e., the depth/volume of the image. 3D images make more sense compared to 2D images, as the 3D image represents more of a real-world representation of an object. Training model on 3D images is a complex and compute-intensive task while 2D images are simple and easy to train, but they are not as accurate as of its 3D counterpart. But 2D images sliced from 3D images could be trained, but they lose a myriad of global information and structure of the whole image [11]. It's also challenging to slice into 2D images from 3D images for training labels as the segmentation masks are mostly black i.e., they are least useful to train the model and only a few depth channels have a small segmentation mask, this inconsistent slicing can generate imbalanced data which can lead to biased model towards one class.

2.7 3D Deep Architectures for Medical Image Segmentation

As 2D U-Net for medical image segmentation started to gain its stronghold, many researchers tried to do the same for 3D images using 2D patches, but they were less accurate as discussed in the previous section. Later down the year after U-Net was published, in 2016 Özgün et al. resealed a new 3D U-Net based architecture for volumetric images [12]. Figure 10 represents the standard 3D U-Net Architecture from [12]. The 3D U-Net looks exactly the same as of 2D U-Net it's just expanded over the third axis to capture volumetric features in medical images.

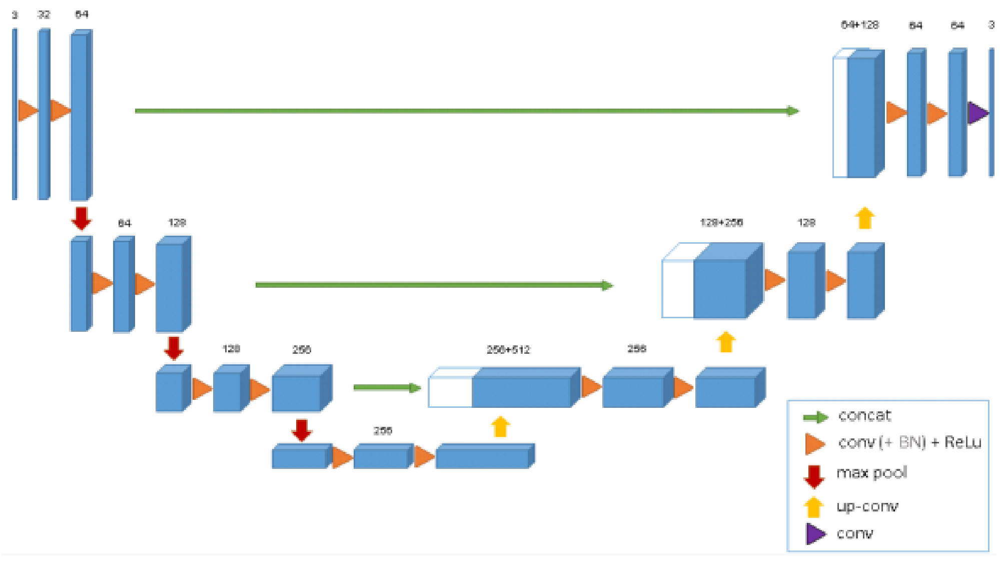

Figure 10. 3D U-Net Architecture. Blue boxes represent feature maps, and the number of channels is denoted above them.

This research was a breakthrough in medical imaging, and it was adopted by many follow-up researchers modifying the architecture and adding visual attention mechanism to improve on its base architecture model. [11], [13], [14] used an attention mechanism on top of the 3D-UNet model with modifications to adapt to the specific domain of the human organ. When attention mechanism was added on top of the 3D U-Net model it gradually outperformed its baseline but on other hand increased the computation complexity by adding an attention mechanism. In early 2021 NVIDIA AI Labs introduced a 3D U-Net architecture variant inspired by Transformers [7], which has shown better results than all previous versions and is more compute-intensive. Overall, all of the 3D U-Net variants achieved a higher dice score than standard 3D U-Net resulting in better segmentation results.

Chapter 3
Software Design and Timeline

3.1 Software Design and Requirements

The exploratory software we are about to develop is a deep learning-based model and it's specifically dealing with 3D medical images so the system should be of high capability as building and training these models is compute-intensive where we need GPUs to train our model and CPUs to deploy our models. The overview of our software is shown in figure 11 which displays in general how we are going to build our exploratory software.

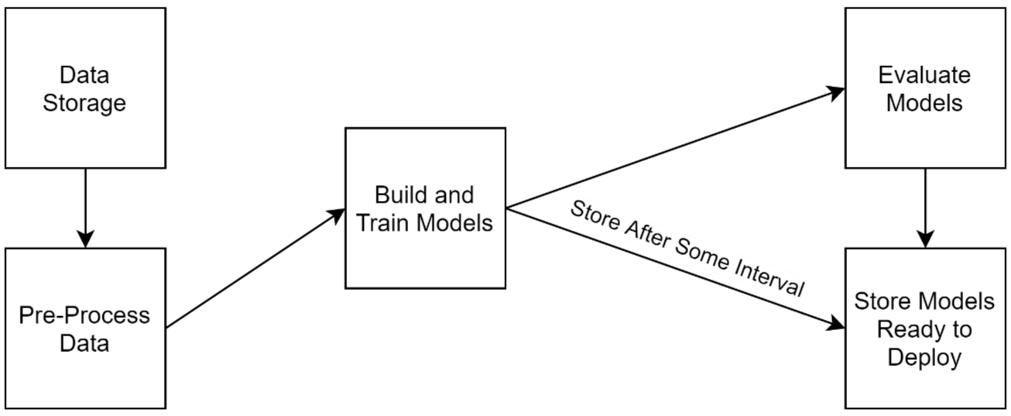

Figure 11. Overview of software for our project.

3.1.1 Data Storage

It is a data repository where the data we need is stored, we have 3D medical images in this data repository. We maintain this repository and this data is not directly tampered with as it can be of huge importance, we copy this data and filter out required data for our specific purpose.

3.1.2 Pre-Process Data

We cannot feed raw image data directly to the model, it first needs to be pre-processed i.e., it needs to be standardized and necessary transformations should be applied before feeding to the model. This is one of the crucial steps that determine the performance of the models, the better pre-processing we perform the higher chances of our model performing well.

3.1.3 Build & Train Models

In this step, we build our deep learning models and train them on pre-processed data. To make these models learn better we need to run them for a specific set of iterations called epochs. In ideal cases, the number of epochs is directly proportional to the accuracy of the model. We will be using PyTorch and MONAI package to build and train these models.

3.1.4 Store Models

While we are training models, we can store iteratively for future use based on specific evaluation metrics. These models can serve a purpose to determine the training behaviour or as a checkpoint. After training is completed, we can save the final model and compare it with previous models to check for improvement.

3.1.5 Evaluate Models

To check the performance of the model, we need an evaluation metric rather than relying on accuracy. We will be using Dice Score or Dice Similarity Coefficient to evaluate our models.

3.2 Use Case of our Model

The saved models could be used in a system that can interact with the client in our case the doctors or medical staff who will upload the raw MRA image to the system which can be on a remote server or built in-house. The system can be standalone and do not require anything else other than input image with a built-in data pipeline and trained models. The input image will be automatically pre-processed and passed through the model to get an output segmentation mask. The whole use case architecture can be seen in figure 12.

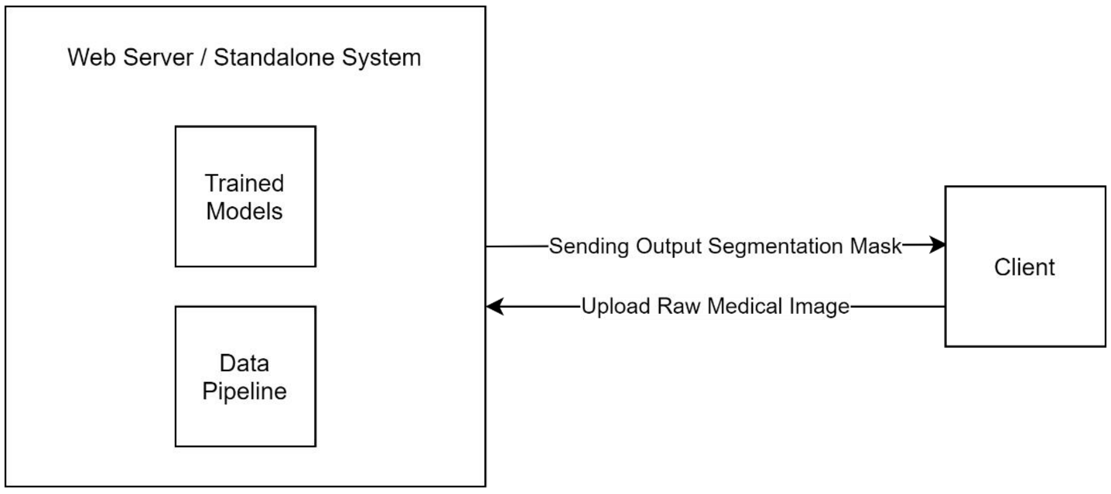
Web Server / Standalone System
Trained Models
Data Pipeline
Sending Output Segmentation Mask
Upload Raw Medical Image
Client

Figure 12. Simple Use Case of our Trained models

3.3 Project Timeline

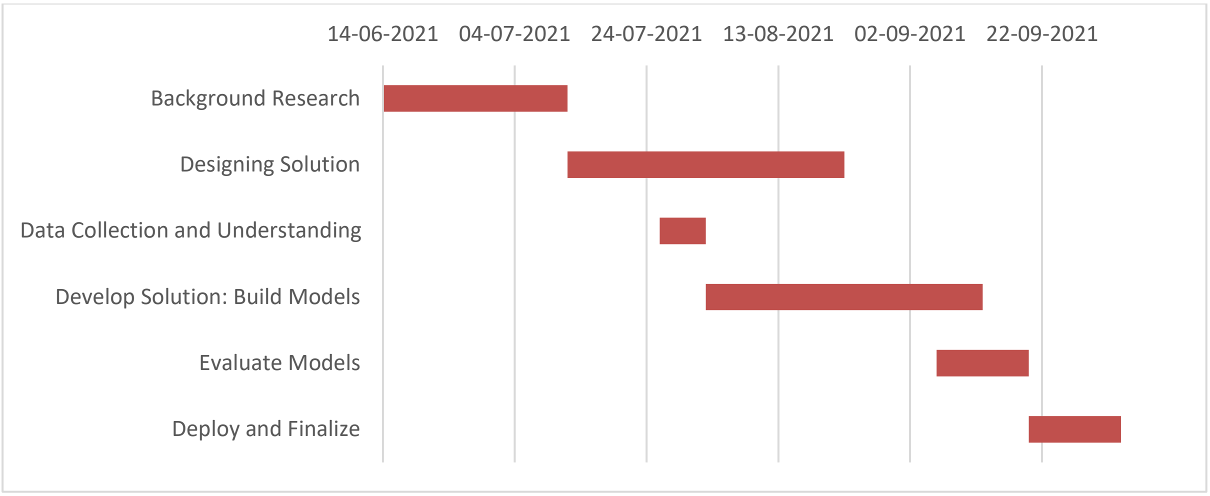

Figure 13. Project Timeline

We will be using Agile Methodology for this project where we can flexibly backtrack to the previous step to prototype our models and experiment faster. We will take 20% of our time to do background research, grasping the domain knowledge and latest trends in this field. We will take around 30% of the time to design the solution, deciding what pre-processing techniques to be used, which deep learning models to use. We will short period for understanding data from the competition and around 30% of the time for building and training the models. Training the models is a very time-consuming process, so we downsize the images and train for a limited number of iterations. The rest of the time is allotted for evaluation of models on the test set and finalizing the code and packaging it. Figure 13 shows the Gantt Chart of our software lifecycle, all of these steps will be discussed in detail in further sections.

Chapter 4
Methodology

We will be following the agile methodology for this project. A lot of experiments have been carried out for this project with an agile methodology by continuously testing new methods and fine-tuning our approach as we experiment. The overall workflow of the project can be understood by checking figure 14.

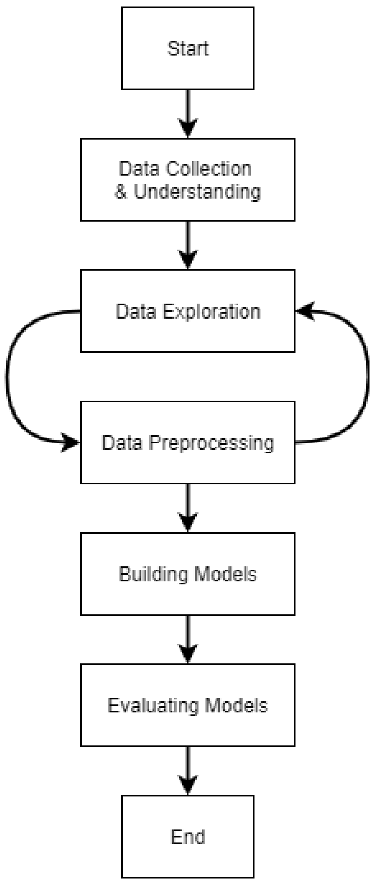

Figure 14. General workflow followed in this project.

We will be using the Python programming language to build exploratory software. We will also use various scientific computation packages like NumPy, SciPy, and medical imaging libraries like SimpleITK and Torchio. For building deep learning models, we will be using PyTorch which is a popular deep learning framework, and MONAI which is built on top of PyTorch for fast prototyping of medical imaging models. We will discuss these libraries more as we proceed in our report.

4.1 Data Collection and Understanding

The dataset is provided by ADAM Aneurysm Detection and Segmentation Challenge [16]. The main aim of this challenge is to develop an automatic detection and segmentation method on Time-of-Flight Magnetic Resonance Angiogram (TOF-MRA) images. Other systems for detecting aneurysm exists using a different modality of medical images, but they contain some artifacts. TOF-MRA is increasingly used in clinical routines, especially for an aneurysm that is smaller in size, each of different shapes, and can occur at any different location. To access this dataset, we first need to register for this competition, even though the competition is over in late 2020. The train set can be downloaded after successful registration for the competition, we will be splitting this train set for training and validation purposes. The total size of the train set is around 33GB containing 113 compressed files, each file containing multiple files which will be described below.

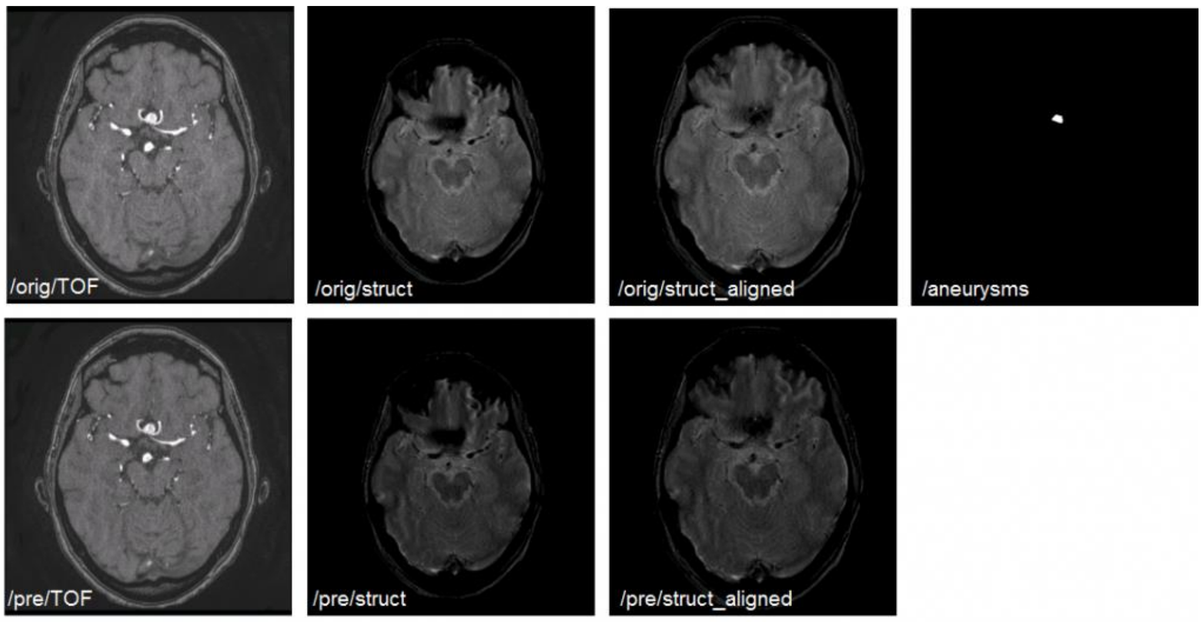

Figure 15. Types of medical images in the dataset (Source: ADAM Challenge).

Each compressed file is divided into sub-folders “orig”, “pre”. The “orig” folder contains all of the original images which are anonymized. The structural image has been aligned to the TOF images. The “pre” folder contains all the pre-processed images to correct bias field inhomogeneities. The “location.txt” is a text file containing the voxel coordinates x, y, z, and the radius of the aneurysm. If the file is empty, it is representing the corresponding MRA image is healthy i.e., no aneurysm. If there is more than one line, it is representing a different aneurysm location in the MRA image. Figure 15 represents all types of images present in the given dataset.

4.2 Data Exploration

Data exploration is a crucial step that helps us know about the general overview of the dataset. It is one of the initial steps in data analysis in which data is analysed and certain characteristics of datasets are described. It generally consists of checking the distribution of the variables/attributes, different data types, outliers, and abnormal values in the dataset. The analysis performed in this step can be used in the Data Preprocessing step to standardize and better process the data.

After getting access and downloading the dataset, we analysed the dataset. There are a total of 113 files in the dataset out of which only 93 images are of the aneurysm. The dataset is a bit unbalanced with the majority of images are of the aneurysm. Figure 16. Showing the distribution of the ADAM Challenge dataset.

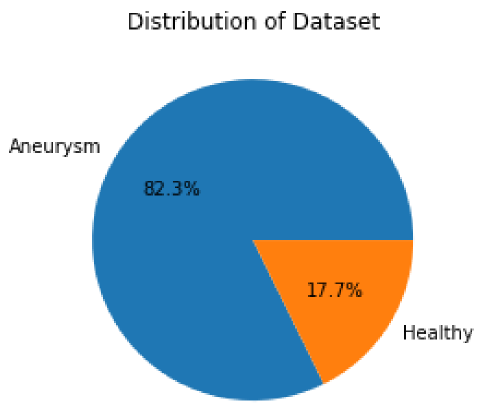

Figure 16. Distribution of ADAM Dataset

We can augment the dataset to increases the size of imbalanced classes or keep it as it is or evaluate our model using proper metrics. If we are augmenting the data operations like scaling intensity, flipping, and rotating the images could help. The available files for a single image in the dataset have been described in the previous section, for our segmentation task we will be using “orig/TOF.nii.gz” images to train our deep learning model and “aneurysms.nii.gz” files as our target segmentation maps. Figure 17 shows one of the example images from the dataset and their aneurysm mask image.

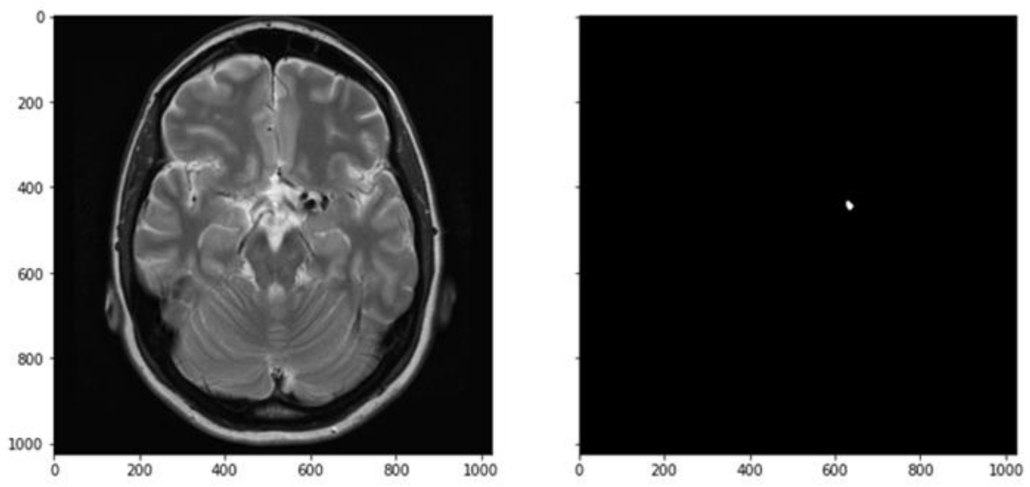

Figure 17. 2D Slice of a TOF-MRA Image (left) and aneurysm mask (right).

The aneurysm is only present in a few channels and the rest of the segmentation mask is all black with no data in them. We checked the distribution of the location of the aneurysm and plotted a histogram shown in figure 15 representing on what slices of 3D image the aneurysm is located.

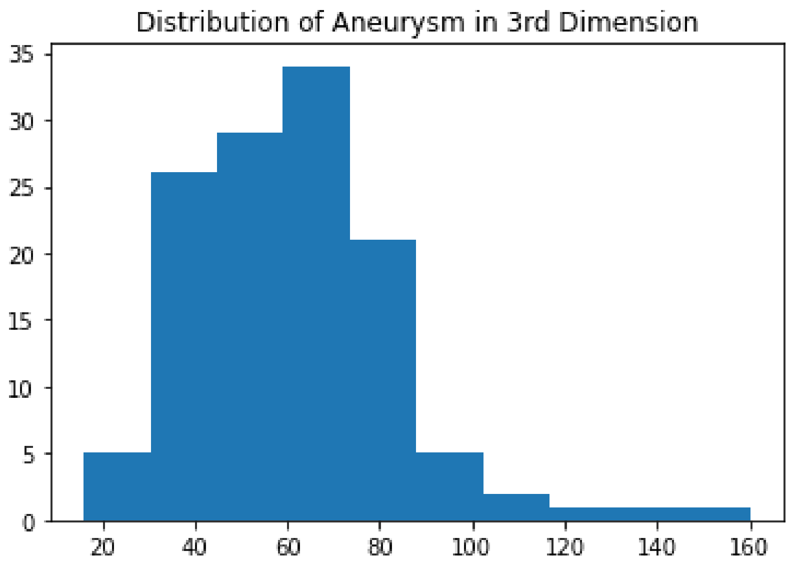

Figure 18. Distribution of Aneurysm location in the 3rd Dimension.

It can be inferred from figure 18 that most of the aneurysm for the images is present between 40th to 90th channels, with few ones in upper ranges. There is one more issue with the TOF-MRA and aneurysm images that there are images with different image dimensions which need to be resized before passing it to the deep learning model, which will be done in the pre-processing step which will be explained in further sections. In some images, there is only one aneurysm bulge present while in others there can be many up to 4 aneurysms locations. We aim to successfully segment these aneurysms for detection and further medical treatment. Other images like pre-processed and struct aligned images are not needed for our problem but can be used in the future if required.

4.3 Data Preprocessing

Data Preprocessing is one of the important steps in this whole pipeline as this process helps to standardize data and better represent it before feeding it to a deep learning model. It is also known as the Data Preparation step. In this step, we perform cleaning and transformations on data to enrich the data. It is a lengthy process where it needs to be revisited several times based on experimentations.

The problem we are dealing with is related to medical images, so we need to perform transformations suitable for such images. As these images are 3-dimensional, performing operations on them requires a significant amount of computing power, so we are trying to minimize the computing power required and maximize the model performance based on these transformations. All of the operations and transformations used will be discussed as follows.

4.3.1 Image Loading

Medical images are in a different format unlike 2D images, which can be loaded using simple, popular computer vision packages. Medical images are 3D and come from different modalities depending on the tools/machines used to capture those images. CT (Computer Tomography), MRI (Magnetic Resonance Imaging), and MRA (Magnetic Resonance Angiogram) are a few of the medical imaging modalities which are non-invasive medical imaging techniques. These images are saved in various formats like Digital Imaging and Communications in Medicine (DICOM) and Neuroimaging Informatics Technology Initiative (NIfTI) are the most commonly used format to store brain imaging obtained after using the previously mentioned modalities, we are given TOF-MRA images in NIfTI format. We are using various Python Software packages for this project, one of them is “SimpleITK” to read 3D medical images. This package allows us to read images of different modalities and different image formats. In machine learning and deep learning, we need to convert raw data from files into a specific data type called arrays. A 2D image is nothing but a 2D array or 2D matrix similarly, a 3D image is a 3D image array, with this package after reading the image file we can convert it into an array/matrix easily. After converting them to a standard representation like arrays, various mathematical operations or transformations can be performed to enhance the image.

4.3.2 Resizing based on Crop or Padding

No raw dataset is perfect, its needs to be standardized first. As discussed in the Data Exploration section, we encountered images of different dimensions, before doing any further processing they need to be brought down to fixed dimensions as it might affect following up transformations in an adverse way. So, in this transformation, we resize the input image based on specific spatial dimensions. We are resizing images into 140x512x512 dimensions, so all the images will be resized to these dimensions. In some cases, the dimensions of the image are less than the specified dimension for our resizing transformation, so this approach applies padding to the image to bring it to specified dimensions and if the dimensions are way greater than the specified dimensions, it crops the images to this dimension.

4.3.3 Resizing

This resizing transformation is a bit different from the previous transformation we discussed. Rather than cropping or padding in this case, this transformation resizes the image completely while preserving its original structure. We are resizing down to 96x96x96 dimensions to reduce the computation time; these dimensions eventually reduced the time taken to train this model by a significant amount of time.

4.3.4 Scaling Intensity

Before passing to any machine learning or deep learning model, it’s necessary to normalize the values. Without normalizing the values, the yielded results are poor so scaling intensities in the image have been a proven approach to yield better results [17]. In the 3D medical image case as well, we will be performing intensity scaling to normalize values.

4.3.5 Label Discretization

The aneurysm files i.e., the target files are not discretized that is the values present are continuous in between 0’s and 1’s. If we pass these values directly to calculate dice score, we will get errors, before evaluating on dice score, these values need to be discretized, so we will be using a discretization transformation on labels i.e., segmentation masks.

4.3.6 Cropping based on Label

The aneurysm mask is very small compared to the whole 3D MRA image, it’s almost insignificant compared to the bigger source image. It will be difficult for the model to find the small spot from this huge image, it’s like finding a needle from a haystack. So, we will be performing a transformation based on labels. As we have labels for each respective TOF-MRA image we will crop the region of the MRA image based on the aneurysm part. The working of this transformation is as follows.

Consider an image with these values

```
[[0.0, 0.3, 0.4, 0.2, 0.0],
 [0.0, 0.1, 0.2, 0.1, 0.4],
 [0.0, 0.3, 0.5, 0.2, 0.0],
 [0.1, 0.2, 0.1, 0.1, 0.0],
 [0.0, 0.1, 0.2, 0.1, 0.0]]
```

And label with these classes

```
[[0, 0, 0, 0, 0],
 [0, 1, 2, 1, 0],
 [0, 1, 3, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0]]
```

So, after applying Crop based on labels we will get cropped image as

```
[[0.1, 0.2, 0.1],
 [0.3, 0.5, 0.2],
 [0.2, 0.1, 0.1]]
```

4.3.7 Output Discretization

After getting some output from the model, we get some positive and negative values, as discussed in label discretization the evaluation metric needs discrete values, so we need to discretize the output to 0's and 1's. This transformation is used to discretize the output similarly but also one-hot encodes the output for easy calculation.

4.4 Building and Training Deep Learning Model(s)

We will be using PyTorch and MONAI python packages to build our deep learning models. PyTorch is an open-source machine learning framework that is widely used for research and production, it also supports GPU support to accelerate the training process by a significant amount of time compared to CPU. MONAI is a PyTorch framework for deep learning in healthcare imaging, it is widely used in medical imaging research to build state-of-the-art models.

We will be using these packages to build and train our deep learning-based segmentation models. As these models are designed for 3D medical images, training these models is a compute intense task, to solve this we will be resizing the images and training on 3D patches of 96x96x96 dimension images. We will be training a variety of models and talk more about the implementation in further sections. Here we will give a brief overview of the models used below sections.

4.4.1 3D-UNet

We discussed this architecture previously in the Background section. This architecture was first proposed to deal with 3D medical imaging tasks [12]. Initially, 2D image segmentation algorithms failed to achieve good accuracy on slices of 3D images, so this architecture was developed in 2016 by the same authors of 2D U-Net. Figure 10 represents the architecture of 3D U-Net, which is similar to 2D U-Net, but the only difference is that instead of 2D convolution layers, 3D convolution layers are used. Even though 3D convolutions operation is a bit compute-intensive it yields better results for 3D medical image segmentation. In earlier days, 3D U-Net has shown great results on brain tumour segmentation tasks and spleen datasets from medical decathlon big datasets. This architecture led the foundation for medical imaging segmentation tasks and more architectures have been using this as a building block in their respective research field.

4.4.2 SegResNet

SegResNet stands for Segmentation Residual Network (ResNet), was designed for segmentation tasks. It uses ResNet [18] like residual connections to improve the learning and segmentation mask prediction capabilities. SegResNet [19] was developed at NVIDIA Labs in 2018 to segment brain tumours from MRI Image and it outperformed other models in BraTS 2018 challenge. Unlike in 3D U-Net which having symmetrical encoder/decoder or contracting / expansive paths, SegResNet follows CNN-based encoder-decoder architecture with an asymmetrical large encoder to extract more image features and a smaller decoder to reconstruct the segmentation mask. An additional branch to the encoder endpoint to reconstruct the original image, similar to auto-encoder architectures [20]. The main motivation behind this branching is to add guidance and regularization to the encoder part as the training data size was smaller. Figure 19 shows the SegResNet architecture from [19].

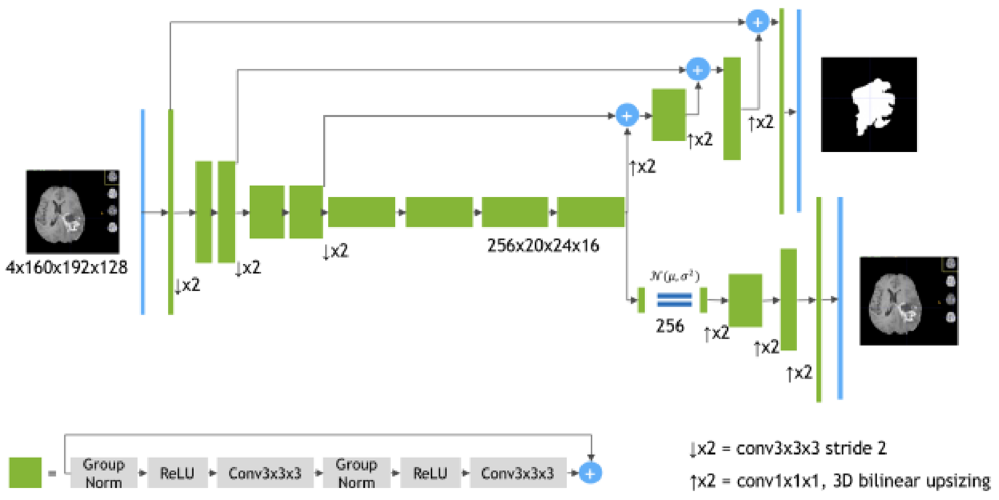

Figure 19. SegResNet Architecture.

The green block is a ResNet-like block with GroupNorm normalization with Convolution layers and ReLU activation. We planned to use this architecture as it was built for limited dataset size and this auto-encoder branch is shown to regularize better than other approaches for smaller datasets.

4.4.3 3D-UNet Transformer

UNet-Transformer or UNETR was developed at NVIDIA Labs in early 2021 [15], inspired by Transformer architecture's success in the field of NLP. The motivation behind this architecture is that models like UNET, SegResNet, and others based on fully convolutional neural networks are limited by learning long-range spatial dependencies. In NLP long-range dependencies issues have been sorted using Transformer's architecture, so this similar technique has been adapted for the volumetric segmentation task. It follows the same encoder-decoder architecture that of UNET but instead of the encoder with CNN (Convolutional Neural Network) an encoder with Transformers is used. An overview of the UNETR approach is shown in figure 20. The benefit of using a transformer as an encoder is to learn sequence representations of the medical image and effectively capture global information, while also following the U-Shaped network designed for the encoder and decoder. All other skip connections from the encoder are directly connected to the decoder in a similar U-Net fashion to generate the final segmentation mask. The detailed architecture of the model is showcased in figure 21.

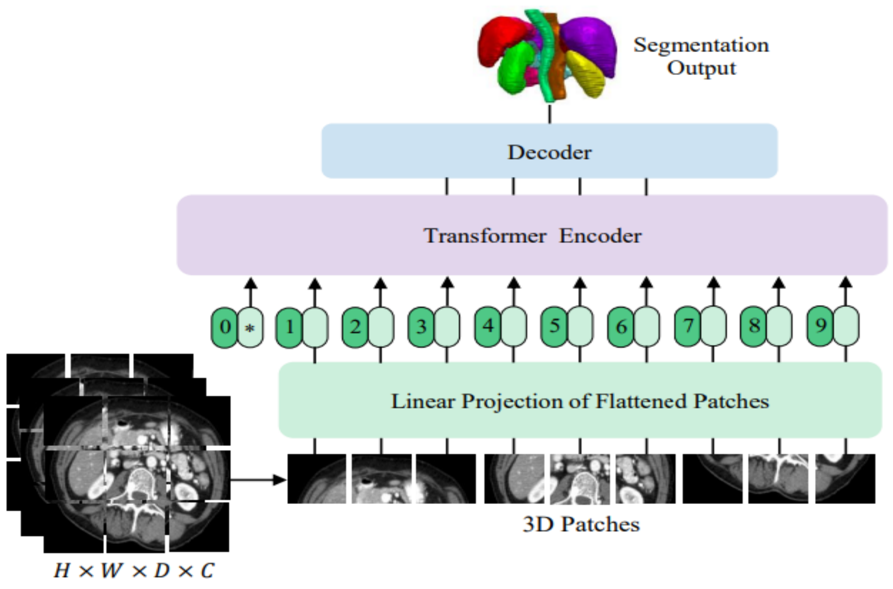
Segmentation Output
Decoder
Transformer Encoder
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
3D Patches
$H \times W \times D \times C$

Figure 20. Overview of UNETR approach for a 3D medical image. The medical image is split into patches and projected linearly with positional encoding and passed through the encoder to learn long-sequence representations and then to the decoder to get the final segmentation mask.

This model has shown state-of-the-art results Multi Atlas Labelling Beyond the Cranial Vault (BTCV) dataset for multiorgan segmentation. We decided to use this model based on sequence learning to capture the global information which could be helpful in our case as these brain vessels have a complex structure.

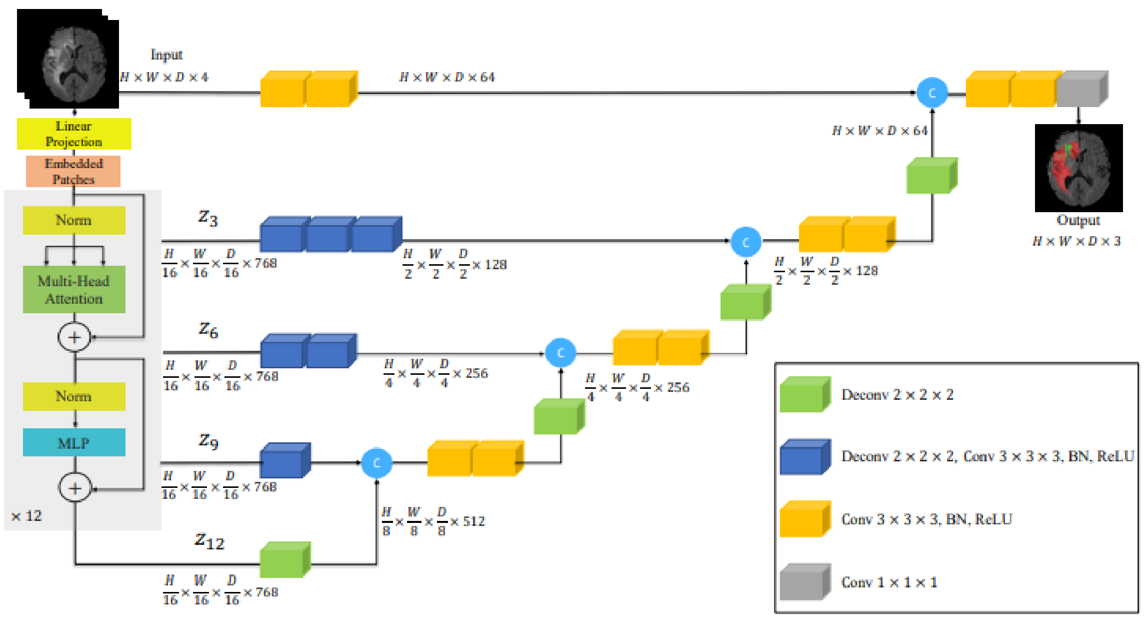
Input
$H \times W \times D \times 4$
$H \times W \times D \times 64$
$H \times W \times D \times 64$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
× 12
z_3
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
z_6
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
z_9
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 512$
z_{12}
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
Output
$H \times W \times D \times 3$
Deconv $2 \times 2 \times 2$
Deconv $2 \times 2 \times 2$, Conv $3 \times 3 \times 3$, BN, ReLU
Conv $3 \times 3 \times 3$, BN, ReLU
Conv $1 \times 1 \times 1$

Figure 21. Detailed UNETR architecture.

As seen in figure 21, UNETR has a similar encoder-decoder architecture with a transformer as the encoder. As the encoder is working on image patches and performing attention mechanisms, this model is much compute-intensive than previous architectures.

4.5 Evaluation

After building and training models, we need a metric to evaluate the performance of the model. Accuracy and error are not a good metric for segmentation tasks, we need a metric that will compare the segmentation mask and give us a proper number or value which can help to determine the performance of the model. Researchers from [21] show that Dice Score or Dice Similarity Coefficient (DSC) is one of the good metrics to evaluate segmentation model performance. The equation for DSC is as follows.

$$\frac{2 * |X \ \cap Y|}{|X| \cup |Y|}$$

Where X and Y represent segmentation masks. It calculates the overlap between two masks and returns a value, values closer to 1 mean it’s a perfect match while a value close to 0 represents no match at all. So, we aim to maximize the dice score for these models.

4.6 Software and Packages

We will be using Python programming language for this dissertation along with PyTorch which is a deep learning library and MONAI library for fast prototyping of our models. The models we are building are quite complex and compute-intensive, so we will be using Google Collaboratory to run our experiments which provides free GPU usage for 12 hours and we will be using local machines to train these models.

4.7 Training Models

4.7.1 Training 3D-UNet

We are using 3D U-Net as benchmarking for other models as it was the building block of all the latest models in medical image segmentation. As our medical brain TOF-MRA images have only one colour channel i.e., grayscale channel out input channels for our 3D U-Net model will be 1 and the output channels will be 2 as we are trying to predict a segmentation mask so 1st channel would be a background and 2nd one will be the segmentation mask. As U-Net consists of contrastive (encoder) and expansive (decoder) paths, we need to define the number of filters to be used in these 3D convolutional layers. We will be using 5 convolution blocks in the encoder part of U-Net of 16, 32, 64, 128, 256 filters respectively from initial to deeper layers. The main job of these convolutions layers is to extract as much information or features by using these filters, so increasing the number of filters will help extract more and better features as we learn from the input images. We will use strides of 2x2x2x2 to move our filters on images or feature maps by this interval which will result in lower dimensions image from the original input image as this operation reduces the image size by little if no padding is applied. This operation is followed by a 3D Max-Pooling operation which reduces the size of the feature map by half by selecting the highest feature among the group of voxels (a voxel is a unit of graphic information that defines a point in three-dimensional space) in our case of medical image. After extracting prominent features by consecutively applying convolution operations and down sampling the input and feature maps, to build a segmentation mask, we use this selective feature. In expansive path, we use these features and up-sample them to increase the dimensions of the feature maps and we use the feature maps of convolution layers from the contrastive path and concatenate them with the up-sampled output by using skip connections, so we perform the same operation for the same number of times as we used convolution operation on contrastive path to building the segmentation mask. We get an output mask, and we compare it with the ground truth mask and calculate an error function also called a loss function to readjust the weights in the U-Net model to perform better next time, this is an iterative process and take a significant amount of time. We are using Dice Loss to calculate the error between model output mask and original segmentation mask and Adam Optimizer to adjust weights in the neural network based on the loss and learning rate. Learning rate is a hyperparameter that controls how much the weights should change in response to the error calculated by the loss function. We train this model for 100 epochs and stop when this model shows no sign of converging the error.

4.7.2 Training SegResNet

This SegResNet model has also been implemented in PyTorch and MONAI framework with the same pre-processing pipeline as mentioned in the above sections. SegResNet is also an encoder-decoder-based CNN network but with an asymmetrically larger encoder to extract image features more precisely and a small decoder to reconstruct the segmentation mask [19]. As previously discussed in 4.4.2 there is an additional branch connection to the encoder endpoint to reconstruct the original image, similar to auto-encoder architectures, refer to figure 19 for architecture. We will be using the same default parameters used in the original research papers for training our model [19]. The ResNet blocks in the encoder consist of two convolution layers followed by Group Normalizations and with ReLU activations. The same CNN approach is used to progressively downsizing images by 2 with strides convolutions and increasing feature size by 2. Kernel size is 3x3x3 with filters number equal to 32. The decoder structure is similar to the encoder one with a single block instead of two blocks. Decoder level begins by upsizing and doubling spatial dimension using bilinear up-sampling followed by encoder output to equivalent spatial level which makes the same spatial size of input image followed by a sigmoid function to squeeze output in between zero to one.

In the paper they used combinations of loss functions we stuck to Dice Loss as we are dealing with binary segmentation tasks, which do not need other loss functions to optimize the weights. We use Adam optimizer to optimize our models' internal weights.

4.7.3 Training UNETR

UNETR stands for U-Net Transformer, it uses a transformer architecture [7] as its encoder and standard expansive path or decoder to build the segmentation path. Usage of transformers for image data was first carried out in mid-2020 at Facebook AI labs which are called Visual Transformers [22]. After successful applications in NLP [7] transformers soon start to gain their traction in the computer vision space which leads to research in the medical image segmentation field developing UNETR at NVIDIA Labs. It uses the same Attention mechanism used in [7] and [22] called Multi-head Attention. In our case, this attention mechanism helps to capture global features from different image patches to accurately find an aneurysm in the whole image giving a segmentation mask.

We will first describe the encoder as one of the most important blocks in this whole network. Before passing our 3D medical image to our transformer, transformers need these images in patches to form a sequence-like input. We create patches of our 3D image using a 3D convolution network first and flatten them with a linear projection of these patches. To identify these patches order in this long sequence and later by the network we perform positional embeddings on these patches. After doing all these steps we send these patches with their position embeddings to the encoder and then to the decoder to get the final segmentation mask this overview can be seen in figure 20. Similar to U-Net here also we use skips connections to the decoder but here the skip connections are passed through by a set of layers to build a feature map before concatenating on the decoder side. These layers perform deconvolution layer followed by convolution layer then by batch normalization and ReLU activation. In this architecture, all convolution layers are followed by batch normalization to regularize the training process and avoid gradient explosion making the whole training process stable. As this was a compute intense training process, we are using different hyperparameters to make this process bit faster. We are using an image size of 64x64x64 and a patch size of 16 i.e., patches of 16x16x16 would be extracted from the 3D image and we used normal perceptron to position embed these patches of size 1024. We set 8 attention heads inside the transformer with a hidden layer of size 256. We also set dropout to 0.5 to avoid overfitting of the network. We are keeping the number of filters the same as of the original model i.e., 64,128,256,512, and 768 as shown in figure 21. We train this model for 100 epochs with Dice Loss as loss function and ADAM optimizer.

More details regarding the results of the experiment will be discussed in further sections comparing with other approaches.

4.8 What is Data Pipeline?

Data Pipeline is a series of steps for data processing before doing any further operation, this further operation could be storing the data, finding insights, streaming data, etc. In our case, at the start of the pipeline the data is ingested through the pipeline and further processed i.e., few transformations are applied before sending this data to train our deep learning-based segmentation models. The simple flowchart can be seen in figure 22.

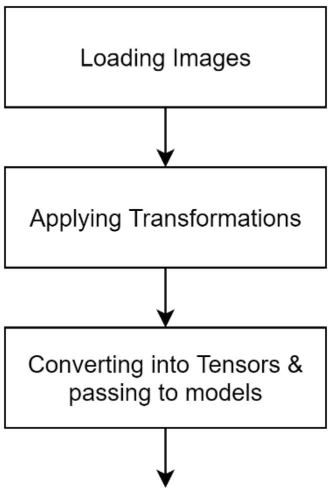

Figure 22. Simple Data Pipeline.

We load the images using the SimpleITK python library which handles medical images. After that we apply all the transformations, we described in section 4.3 and after that, we convert the transformed image to a tensor as deep learning models deal with multi-dimensional data which are called tensors. We apply all these transformations using the MONAI package which provides numerous transformations in the field of medical imaging.

4.9 Training Workflow

We build this whole approach in an agile way, where we can go back to the previous section of workflow to amend changes if required. The whole workflow of our approach can be seen in figure 23 which represents all the steps we carried to train these models. The “Build Dataset” process block in figure 22 is an abstraction of the Data Pipeline we explained in the previous section.

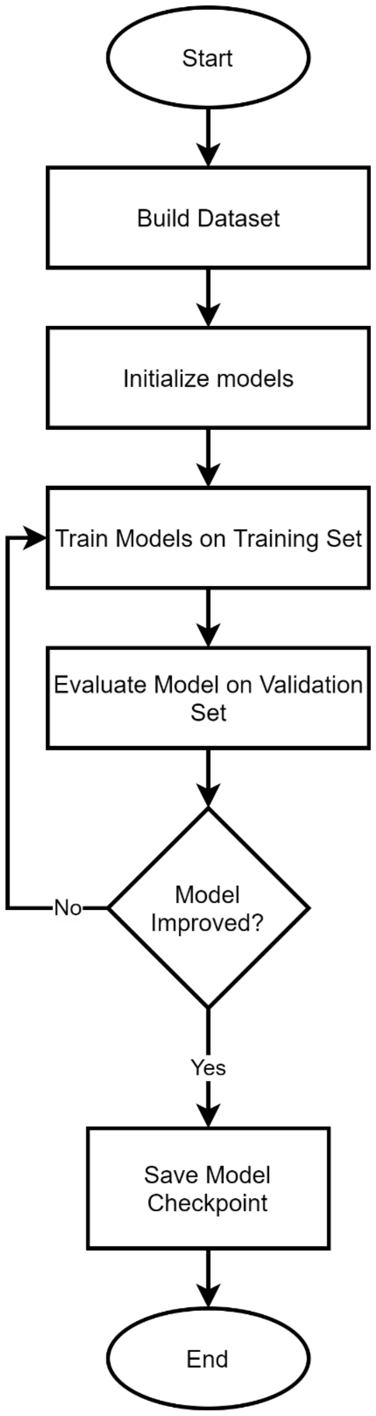

Figure 23. Training Workflow

In workflow in the “Initialize Model” block, we built the model architecture by using the hyperparameters described in section 4.7 and load them on GPU as processing and training are faster compared to CPU. We create a training and validation set from the dataset, a training set will be used to train the model and we will evaluate the model on the validation set. We set a certain number of iterations to train the model that means the training set will be shuffled and used to train the model multiple times which is equal to the number of iterations we set. We train the model by giving batches of the data to help the model learn efficiently. After training on all batches of the training set, we set the model to evaluation model and test our model’s performance on the validation set, if the performance is better than the previous iteration, we save the model, we repeat the same process for the given number of iterations or until the model converges, i.e., model does not improve over time. This process is the same for all the discussed models with minor modifications which will be clarified in the results section if any. The whole training process is time-consuming, and we are not training for thousands of iterations as mentioned in models research papers because of hardware limitations and time constraints.

Chapter 5
Results

5.1 Overview

We used the same pre-processing data pipeline for all the above-discussed models. Due to limitations of computing power, we resized the medical images to 96x96x96 dimensions. We trained each model for only 100 epochs as training these models was very time-consuming, it takes more than 36 hours to train for 100 epochs for a single model. In research papers, the authors have trained for 25000 epochs to get accurate segmentation masks for their focused task, which was infeasible at this stage for aneurysm segmentation task. All of the models have been evaluated on Dice Score / Dice Similarity Coefficient for both training and test data.

5.2 3D U-Net

We trained 3D U-Net as our first model to check the performance of the legacy model architecture. This model performed quite well on this task than expected being an old model. The models' performance can be seen in figure 24 which showcases the training loss, training dice score, and validation dice score. The training loss is going down over time showing convergence and dice score on training set going up showing the model is learning to detect aneurysm though the dice score is not that great. As the number of epochs progresses the training loss after 30 epochs start to shake around which might show that this model is struggling to learn more features and getting saturated till the end.

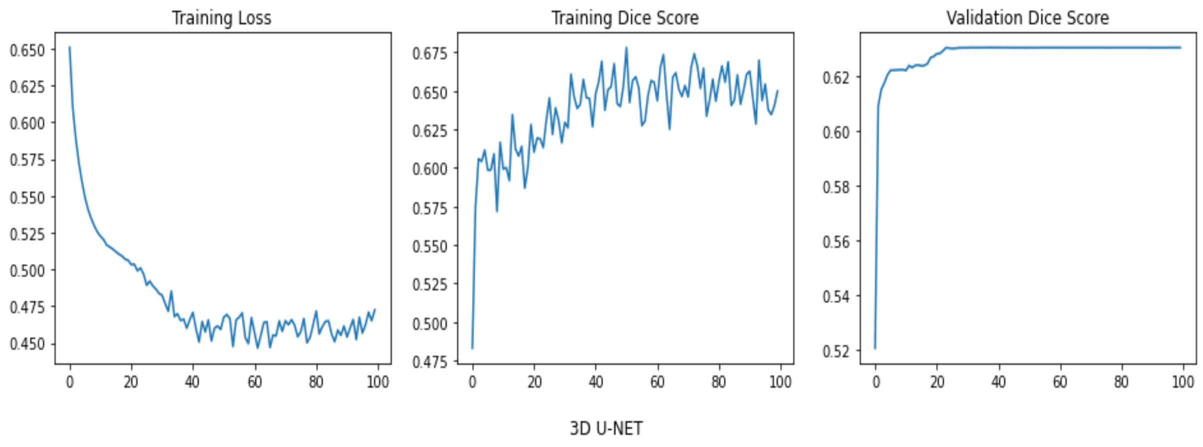

Figure 24. Model Performances Metric on 3D U-Net

The validation dice score also increases but after 20 epochs it fails to learn new features which help to detect aneurysm making it reach a saturation point further making it difficult to learn further.

5.3 SegResNet

As a quick overview, SegResNet Model follows the same encoder-decoder architecture as of 3D U-Net but with an asymmetrical structure with a bigger encoder for feature extraction and a smaller decoder to create a segmentation mask. This model took 38 hours to train for 100 epochs showing a little bit better performance than 3D U-Net.

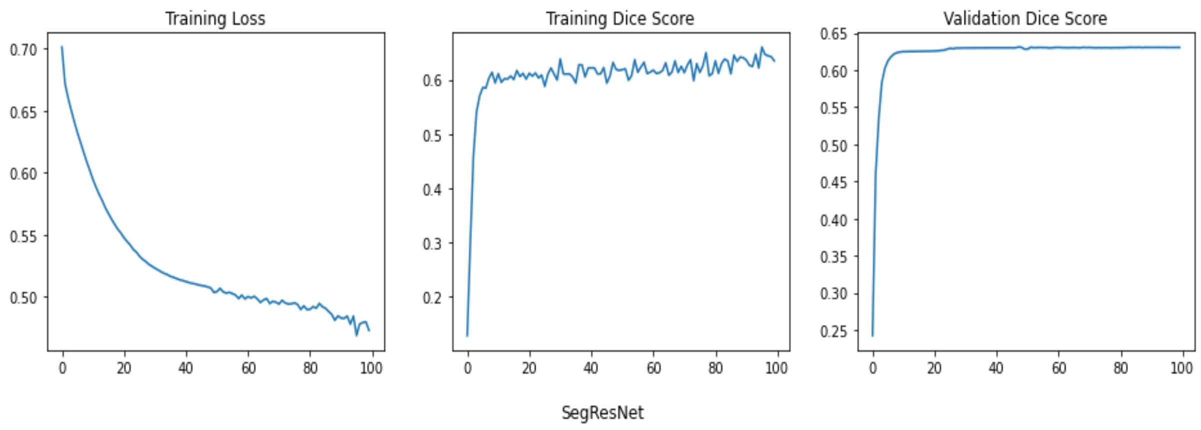

Figure 25. Model Performance Metrics on SegResNet

As seen in figure 25 which shows the model performance over the number of epochs, The training dice loss is gradually decreasing smoothly which is showing sign of convergence after 100 epochs, the dice score on training and validation set the shoot up very quickly but is improving with we very small upward slope. The reason behind this could be the auto-encoding mechanism used internally in this model architecture which creates slight variations in the input features which takes a skip connection from input to end of the encoder to performing variational autoencoding of the inputs image.

5.4 3D UNETR

The most recently released architecture UNet Transformer aka UNETR has shown great results in a variety of tasks. This is one of the most compute-intensive architectures which took around 43 hours to train for 100 epochs and shown some surprising results. This follows the same architecture of U-Net with transformer network at the encoder and regular convolution layers at the decoder for building segmentation mask. As shown in figure 26 the training dice loss is showing signs of convergence with some shaking behaviour around the

80th to 100th epoch. On another hand, the dice score on the training set is following a staircase kind-off pattern showing improvements in steps. The strange behaviour of dice score on validation set represents a saturation point, to tackle this we need more data to generalize our model.

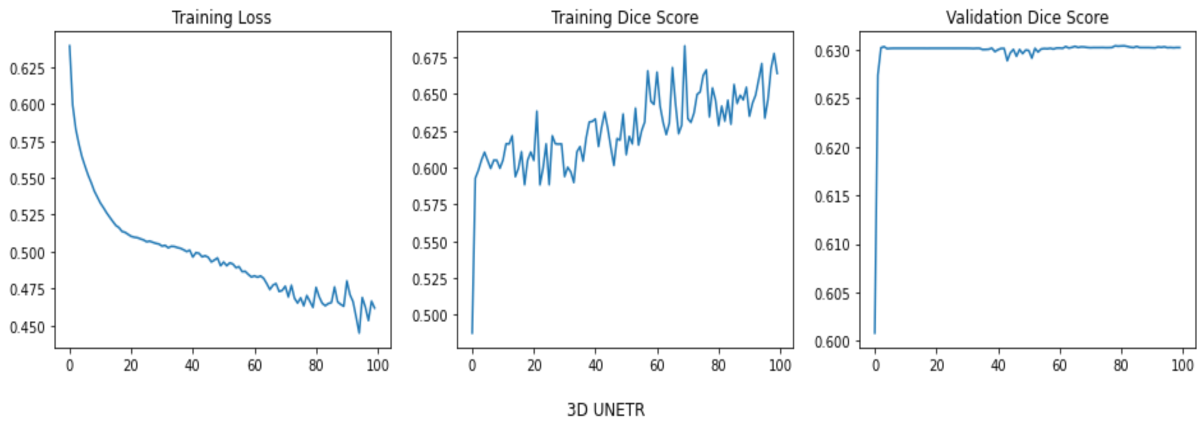

Figure 26. Model Performance Metrics on 3D-UNETR

5.5 Insights after Training

After looking at all the model’s performance on this aneurysm segmentation task, we observe that all the models are reaching a saturation point on the validation set but having different training behaviours on the training set. This could be happening because of having a limited amount of data with high variations in the validation set or it could be because it’s not trained enough to perform better on the validation set or could be both. This can be solved with a bigger dataset and training these models for thousands of epochs.

5.6 Results for different Inputs

After completed training for 100 epochs for each model, we loaded the best weights for the model and tested it on new MRA images. Although the results are not perfect but satisfactory which can be seen in figure 27 (a) where we are showing a slice of 3D MRA image with all the model's segmentation masks. SegResNet is the closest one and UNETR stands second based on the output mask, there might be great performance gain if UNETR and SegResNet could be trained for thousands of more iterations rather than only 100 iterations. In this example, 3D UNet failed to produce any segmentation mask for the given input image, which explains the early saturation point in training loss in figure 24. 3D U-Net being a basic architecture might be failing to capture the small aneurysm mask, in some cases, this mask is even smaller and hardly visible. This could have been possibly avoided by keeping the original image dimensions and not resizing them, but we had to reduce the dimensions due to time and computing power constraints.

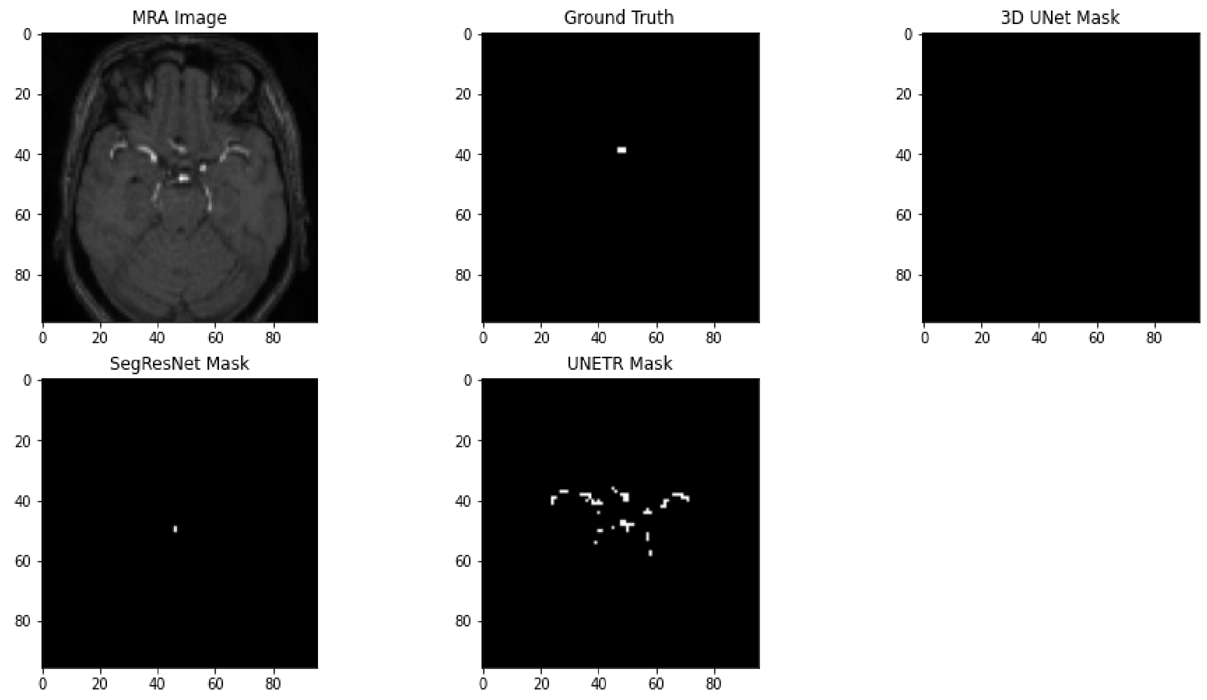

Figure 27 (a). Example output mask of the image with different models

After testing our model on regular images, for experimentation purposes, we tampered with the images to check if our model is robust enough to accommodate the tampering and produce an accurate segmentation mask. In real life, in many cases in medical imaging, several artifacts can be introduced in the images like noisy images, images with different intensities, and much more. Models should be robust against these artefacts to tackle real-life problems and to produce better results in such scenarios. These artefacts can also be considered as a type of adversarial attack on neural networks. An adversarial attack is a kind of attack where a minor modification in input data can cause the neural network to fail [23]. To test the robustness of our models against such attacks or artefacts we tampered with the input image, by changing the intensity of the input medical image. The results can be seen in figure 27 (b) where we just changed the intensity by a minor value and all the models failed to output a segmentation mask.

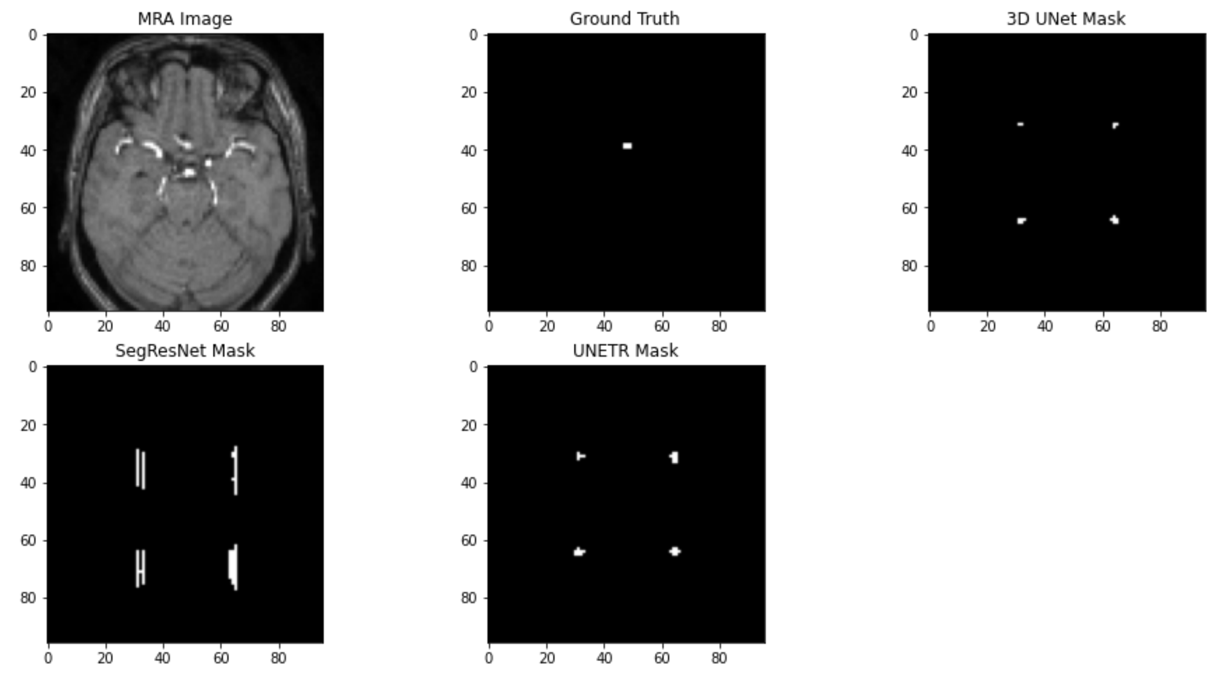

Figure 27 (b). Example output mask of models if the image has tampered

This could make an adverse impact if we used such models that are susceptible to such artefacts or attacks in production and in such delicate fields where human life is at stake. To avoid this and to make our model more robust, we need more data along with such data which represents such artefacts which will help our model learn to handle it. Another way could be using a pre-processing step that handles all such attacks or artefacts, in the pre-processing step it could be anything like another deep learning model or handcrafted rule-based approaches.

Chapter 6
Discussion, Future Work & Conclusion

6.1 Comparison with ADAM Challenge Leader boards

A research paper was published in mid-2021 listing all the approaches used by the top-ranking teams in ADAM Challenge [24] and the model's metrics. The challenge ended in December 2020 and the test set was taken down from their website. Many teams used 3D U-Net-like architectures in this competition with an ensembled approach by training multiple models and making them vote to predict the segmentation mask. These models were being trained for thousands of iterations to get an optimal result whereas we only trained for 100 epochs which are significantly less compared to the top teams' models. Some of the teams used no-new-UNet (nn-UNet) which is a robust and self-adapting network built on top of U-Net [25]. Some teams used external data sources to train the model, the top-ranking team used a combination of provided and external data to gain better dice scores. These models achieved a 0.42 dice score on the test set, unfortunately, the test set was taken down, so we used a sample of the training set as the test set to evaluate the model. We got a dice score of 0.63 from SegResNet and UNETR model with considering the background of the image, if we do not consider the background, we get dice to score around 0.13 which is still better than some of the leader boards teams as shown in Table 1. [24] shows the score of top-performing teams but it's still not clear whether they include background for calculating dice score or not, and the important point here is all the teams are using original image dimensions while we are resizing the images to lower dimensions because of fewer resources at our end. The teams participated in this challenge were from academic research background having a plethora of resources and years of experience.

Table 1: Dice Similarity Coefficient (DSC) or Dice Score of the top teams from ADAM Challenge. This table is taken from Table F1b of [24].

Team Name	***DSC Train Set***	***DSC Test Set***
Junma	0.83	0.41
Joker	0.79	0.40
Kublac	0.78	0.28
Inteneural	0.73	0.17
Zelosmediacorp	0.40	0.09

IBBM	0.29	0.01
Stronger	0.21	0.07
Xlim	0.27	0.21
TUM_IBBM	0.11	0.07

6.2 Our Results

As we discussed the performance of the top teams in the ADAM challenge in the previous section, our results are not that great compared to the top teams, but we should also consider the fact that we used resized images rather than a whole complete image. Resizing an image can severely affect the model's performance in both good and bad ways, resizing reduces the information in the image sometimes even wiping out the important features i.e., tampering with the pixel values which can eventually affect how our model learns. The other factor is the teams trained their models for thousands of iterations while we only trained for 100 iterations. Table 2 shows the Dice Score / Dice Similarity Coefficient for our trained models

Table 2: Dice Score of our models trained over 100 epochs

Model Name	***DSC Train Set***	***DSC Test Set***
3D UNet	0.62	0.62
SegResNet	0.64	0.63
UNETR	0.675	0.62

6.3 Future Work

In this section, we will discuss all the approaches we would have loved to experiment on, if we had more time, more compute resources, and better domain knowledge.

6.3.1 Better Data Preprocessing

Deep learning generally has the power to extract features automatically given raw data and enough compute power, but sophisticated data pre-processing pipelines can help the model learn faster with already pre-processed data. Few approaches can enhance medical image features, but they require quality domain knowledge and its time consuming to build such pipelines. For our problem, we would have like to strip the skull from the medical image to better focus on the cerebral part. More decent approaches can be included like brain vessel segmentation as an aneurysm is formed in these blood vessels which in itself is a complicated task and needs a deep learning-based segmentation model as presented in [26] for doing vessel segmentation.

6.3.2 Data Augmentation

As discussed in 5.6 medical images are subject to artefacts that can be caused due to the environmental conditions in which these images are captured. Our model failed to perform well on the same image when we change the intensity as seen in figure 24(b). To tackle such scenarios, we need to make our model more robust by training on such examples. To solve this, we can augment our training data by introducing such artefacts on TOF-MRA images without tampering with the segmentation mask. Artefacts like noise, rotation in images, changing intensities, and flipping images on different axes can help to counter most of the artefacts making them more robust.

6.3.3 Bigger Dataset and Access to Test data

Deep learning methods generalize better and better as we feed more data with variations. For our problem, we had a limited amount of data with only 113 samples out of which we need to split to create a train and test/validation set. If more data was provided or obtained from other sources then the model performance would have been much better, as it would have let the model generalize on different variety of data with more variations to learn from. If a test dataset was provided, then we would be able to evaluate our model based on the competition standards.

6.3.4 More Compute Power

Training deep learning model is a quite challenging task and to train faster we need high-end GPUs, training on CPU is possible but it's 50 times slower on CPU compared to GPU. In the case of medical images which are 3-Dimensional, more computation power is required. Along with high compute power these model architectures need to be trained for the higher number of epochs to reach a decent dice score. If we had time and compute power, we would have trained on original dimensions of images for longer iterations to reach the point of convergence. In deep learning, there is a trend that bigger models produce better results, high computation would have allowed us to build bigger and deeper models with more layers. There is also a phenomenon in deep learning known as Grokking [27] where the model starts performing well after thousands of iterations after being overfitted on the training set and suddenly reaches higher accuracy on the validation set this research was carried out at OpenAI explained in [27]. This phenomenon might be also applicable to our problem as we reach a saturation point on the validation set as discussed in the Results chapter, so letting the model train for thousand epochs might result in good accuracy at some point.

6.3.5 Specialized Network for Aneurysm Task

An aneurysm is very small, and the used network architectures are designed to segment large objects in an image. This might be one of the reasons why these models are performing poorly on this task. A specialized architecture solely for this task needs to be used or built which can solve this problem, but most of the approaches use 3D U-Net-based architectures with mediocre accuracy on segmentation masks.

6.3.6 Building an Ensemble of Models

These models could be trained with different parameters and different settings resulting in a new model for each setting. All of these models could be combined in an ensembled way i.e., all the outputs will be combined to get an output segmentation mask. This approach could make the ensemble model more robust and might lead to better segmentation masks.

6.3.7 Building a Complete System

A complete solution can be built if we would have a highly accurate model which was capable of segment aneurysms in various conditions, our model is not robust yet. The solution might be a web application where we host our trained model on a web server or a standalone application on the local machine, where the doctor/staff would upload the MRA file of the patient to detect aneurysm as explained in section 3.2. Currently, we have the whole pipeline is in place, but we do not have robust accurate models.

6.4 Conclusion

In this project / exploratory software, we worked on building medical image segmentation models for aneurysm segmentation tasks using the latest deep learning architectures. We studied previously used conventional segmentation methods in depth in literature review and background section in depth while gaining valuable domain knowledge in this field. We learned that medical images cannot be treated as normal 2D images, they contain much information with an additional dimension to deal with. To tackle this task, we studied this vast domain along with the latest deep learning methods to solve this segmentation task. We adopted a software development life cycle to divide this problem into multiple segments and work on them in the allotted time, these methods are proven for better software development and to deliver results in a more ordered and timely fashion.

We gained access and explored the dataset in detail, provided for the task, we pre-processed the data while applying necessary transformations before giving them to the learning algorithms. We build and train three different deep learning algorithms for aneurysm segmentation tasks namely 3D-UNet, SegResNet, and UNETR. Due to limitations of computing resources, we resize the images to 96x96x96 dimensions and train each model for only 100 epochs as training them was hardware intensive and time-consuming, collectively it took more than 100 GPU hours to train for 100 epochs. After training, we got satisfactory results and metrics curves showing improvements over time and loss curves showing signs of convergence over the iterations. If more time and high-end GPUs had been allotted for training these models then the models would have converged pretty well, resulting in better prediction of segmentation masks.

We learned that such methods could take a significant amount of time to get an accurate segmentation mask, to solve this feature-engineering i.e., feature extraction approaches like skull-stripping, blood vessel segmentation models could be used, but they are complex deep learning-based models needed to train on specific tasks. Even without doing such sophisticated feature extraction steps our model is learning pretty well with a decent dice score. This problem is unique even in medical segmentation tasks as an aneurysm is really small in size and form in between blood vessels which can be challenging to detect, in addition to this if there are any artifacts while capturing the image, this can lead to inaccurate predictions.

Even though our simple data modelling and training approach could adapt to be used with more data and computer power to train for thousands of iterations to get a more robust and accurate model.

List of References

1. N. Etminan and G. J. Rinkel, “Unruptured intracranial aneurysms: development, rupture and preventive management,” Nature Reviews Neurology, vol. 12, no. 12, p. 699, 2016

2. Brain Aneurysm [Online]. Available at https://www.nhs.uk/conditions/brain-aneurysm/ [Accessed 16th July 2021].

3. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition. IEEE, 86(11), pp. 2278-2324.

4. S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

5. Albawi, Saad et al. “Social Touch Gesture Recognition Using Convolutional Neural Network.” Computational intelligence and neuroscience vol. 2018 6973103. 8 Oct. 2018, doi:10.1155/2018/6973103.

6. Ronneberger, O.; Fischer, P. & Brox, T. (2015), 'U-Net: Convolutional Networks for Biomedical Image Segmentation', Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.

7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

8. Jetley, Saumya & Lord, Nicholas & Lee, Namhoon & Torr, Philip. (2018). Learn To Pay Attention.

9. X. Wang, R. Girshick, A. Gupta and K. He, “Non-local Neural Networks," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 7794-7803.

10. Alireza Norouzi, Mohd Shafry Mohd Rahim, Ayman Altameem, Tanzila Saba, Abdolvahab Ehsani Rad, Amjad Rehman & Mueen Uddin (2014) Medical Image Segmentation Methods, Algorithms, and Applications, IETE Technical Review, 31:3, 199-213, DOI: 10.1080/02564602.2014.906861.

11. Isensee F, Petersen J, Klein A, Zimmerer D, Jaeger PF, Kohl S, Wasserthal J, Koehler G, Norajitra T, Wirkert S, et al. (2019) nnU-Net: self-adapting framework for U-Net-based medical image segmentation. In: Bildverarbeitung für die Medizin 2019. Springer, pp 22–22.

12. Çiçek, Özgün & Abdulkadir, Ahmed & Lienkamp, Soeren & Brox, Thomas & Ronneberger, Olaf. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.

13. S. Banerjee, A. K. Dhara, J. Wikström and R. Strand, "Segmentation of Intracranial Aneurysm Remnant in MRA using Dual-Attention Atrous Net," 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 9265-9272, doi: 10.1109/ICPR48806.2021.9413175.

14. Islam, Mobarakol & Vs, Vibashan & Valanarasu, Jeya Maria Jose & Wijetilake, Navodini & Utkarsh, Uppal. (2020). Brain Tumor Segmentation and Survival Prediction Using 3D Attention UNet. 10.1007/978-3-030-46640-4_25.

15. Hatamizadeh, A., Yang, D., Roth, H. and Xu, D., 2021. Unetr: Transformers for 3d medical image segmentation. arXiv preprint arXiv:2103.10504.

16. Kimberley Timmins, Edwin Bennink, Irene van der Schaaf, Birgitta Velthuis, Ynte Ruigrok, & Hugo Kuijf. (2020). Intracranial Aneurysm Detection and Segmentation Challenge. 23rd International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2020), Lima, Peru. Zenodo. https://doi.org/10.5281/zenodo.3715848

17. Zhuge, Ying, and Jayaram K Udupa. "Intensity Standardization Simplifies Brain MR Image Segmentation." Computer vision and image understanding : CVIU vol. 113,10 (2009): 1095-1103. doi:10.1016/j.cviu.2009.06.003

18. He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

19. Myronenko, Andriy. "3D MRI brain tumor segmentation using autoencoder regularization." BrainLes@MICCAI (2018).

20. Diederik P. Kingma; Max Welling, An Introduction to Variational Autoencoders , now, 2019.

21. Taha, A. A., & Hanbury, A. (2015). Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC medical imaging, 15, 29. https://doi.org/10.1186/s12880-015-0068-x

22. Wu, Bichen & Xu, Chenfeng & Dai, Xiaoliang & Wan, Alvin & Zhang, Peizhao & Tomizuka, Masayoshi & Keutzer, Kurt & Vajda, Peter. (2020). Visual Transformers: Token-based Image Representation and Processing for Computer Vision.

23. Huang, Sandy & Papernot, Nicolas & Goodfellow, Ian & Duan, Yan & Abbeel, Pieter. (2017). Adversarial Attacks on Neural Network Policies.

24. Timmins KM, van der Schaaf IC, Bennink E, Ruigrok YM, An X, Baumgartner M, Bourdon P, De Feo R, Noto TD, Dubost F, Fava-Sanches A, Feng X, Giroud C, Group I, Hu M, Jaeger PF, Kaiponen J, Klimont M, Li Y, Li H, Lin Y, Loehr T, Ma J, Maier-Hein KH, Marie G, Menze B, Richiardi J, Rjiba S, Shah D, Shit S, Tohka J, Urruty T, Walińska U, Yang X, Yang Y, Yin Y, Velthuis BK, Kuijf HJ. Comparing methods of detecting and segmenting unruptured intracranial aneurysms on TOF-MRAS: The ADAM challenge. Neuroimage. 2021 Sep;238:118216. doi: 10.1016/j.neuroimage.2021.118216. Epub 2021 May 27. PMID: 34052465.

25. Isensee, Fabian & Petersen, Jens & Klein, André & Zimmerer, David & Jaeger, Paul & Kohl, Simon & Wasserthal, Jakob & Koehler, Gregor & Norajitra, Tobias & Wirkert, S. & Maier-Hein, Klaus. (2018). nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation.

26. Cui, C., Liu, H., Englot, D. J., & Dawant, B. M. (2021, February). Brain vessel segmentation in contrast-enhanced T1-weighted MR Images for deep brain stimulation of the anterior thalamus using a deep convolutional neural network. In Medical Imaging 2021: Image-Guided Procedures, Robotic Interventions, and Modeling (Vol. 11598, p. 115980K). International Society for Optics and Photonics.

27. Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2021). GROKKING: GENERALIZATION BEYOND OVERFIT-TING ON SMALL ALGORITHMIC DATASETS. In ICLR MATH-AI Workshop.

Appendix A
External Materials

A.1 Project Code

https://github.com/AdivarekarBhumit/Dissertation-Aneursym-Segmentation

Appendix B
Ethical Issues Addressed

B.1 No Ethical Issues

In this project there are no ethical issues involved, all of the software and libraries used for development are open source. The data is provided by competition upon participation.