Brain Tumor Detection and Visualization in VR/AR

A PROJECT REPORT

DE-41 (DC&SE)

Submitted by

NS Muhammad Ahmad Masood
(Registration No:297374)

NS Mian Ahmed Raza Mahmood
(Registration No:321757)

NS Fawad Ahmed Qureshi
(Registration No:283119)

Bachelors
in
Computer Engineering
Year
2023

Project Supervisor

Dr. Ali Hassan

DEPARTMENT OF COMPUTER AND SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING
PESHAWAR ROAD, RAWALPINDI

Acknowledgements

First and foremost, we would like to thank ALLAH Almighty the most merciful and the most beneficent for all His blessing upon us, who gave us the strength to accomplish this work. We would also like to thank our families who gave us the opportunity to pursue studies in higher education.

We would like to express our sincere gratitude to our supervisor, Dr.Ali Hassan, for his intellectual support and ideas. He is a truly inspirational instructor, who supported us during this under graduate study. He kept us motivated during this arduous journey while providing both technical and moral support. His incisive feedback propelled us to polish our skills and critical thinking in order to accomplish our project goals. Without his precious support, it would not be possible for us to complete this project. We can't thank him enough for the encouragement he showed in us that kept us on track in this tough and tiring expedition. It has been a great honor for us to accomplish our aims under his supervision.

(Muhammad Ahmad Masood) I am grateful to be able to contribute my research in the biomedical field. I am motivated by the memory of my mother, Naila Aslam, who died after a long and difficult battle with pancreatic cancer. I dedicate my research to her, whose loving care enabled me to achieve so much in life.

We hope that our research and project will contribute to making the world a better place, where no one dies of cancer.

Muhammad Ahmad Masood
Mian Ahmed Raza Mahmoood
Fawad Ahmed Qureshi

Abstract

Globally, cancer is the second leading cause of death. Approximately 70% of deaths from cancer occur in the lower middle-income countries. Head and neck cancers are the sixth most common cancers worldwide, with 630000 new cases diagnosed annually, causing 350000 deaths. [1] Globally, brain tumors are also a significant source of cancer-related morbidity and mortality, with an overall incidence of 4–5/100 000 cases annually, contributing to 2% of all cancer deaths, ranking at 10^{th} place among cancers as the leading cause of death. [2] [3] In Pakistan, brain cancers rank at 11^{th} place, where 150000 new cases of cancer are diagnosed annually, causing 60%–80% of deaths. Most deaths and grievous side effects arise from the lack of tumor detection in time when the optimal time for treatment has already passed. This may be because of a multitude of reasons, like deficiency of money, doctors, and the belief that such a thing cannot occur to oneself. We proposed a solution in the form of Computer-Aided Diagnostic (CAD), where MRI images would be classified and tumors segmented by convolutional neural network (CNN) models, trained on previously given data sets. This would serve as an assisting tool for doctors to get a second opinion of sorts and to minimize any human error that may result in a false-negative result, particularly in a situation where false-positives are highly preferable to false negatives. Our solution would also allow any person who had an MRI of their brain done, for any other reason can process their MRIs to ensure that no tumor has been formed. This easy and cheap solution to detect tumors would decrease the average time between the formation of a tumor and its detection. We further aim to make a 3D model, VR/AR compatible, that would allow you to easily visualize and manipulate the tumor inside the brain model. This would allow for an understanding that a plain 2D image may not convey. The model could be used as training material to train new doctors, as well as allow patients to be better informed due to easy-to-understand visuals. The model can also be used to trace the progression of a patient's tumor, allowing the doctors and medical staff to enhance their understanding of the tumor, and thus propose the optimal treatment plan.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Brain cancer, also known as brain tumors, is a complex and challenging disease that affects the delicate and vital organ, the brain. It involves the abnormal growth of cells within the brain or its surrounding structures, which can disrupt normal brain function and potentially be life-threatening. Brain cancer encompasses a wide range of tumor types, including both benign and malignant tumors.

In the battle against brain cancer, medical imaging plays a crucial role in the diagnosis, treatment planning, and monitoring of the disease. Among the various imaging modalities available, Magnetic Resonance Imaging (MRI) stands out as a powerful tool for providing detailed and comprehensive visualization of brain tumors.

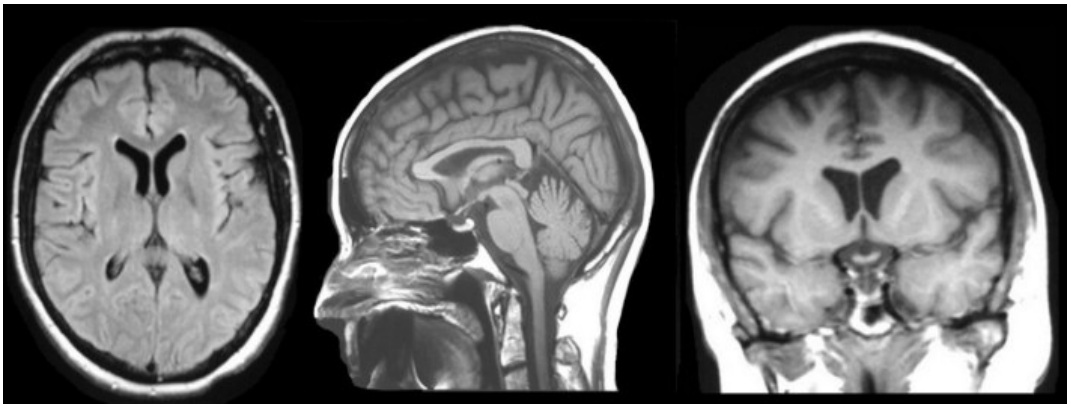

Figure 1.1: MRI Images

MRI utilizes strong magnetic fields and radio waves to generate detailed images of the brain. It offers exquisite anatomical detail, allowing healthcare professionals to accurately identify and characterize brain tumors. MRI scans can reveal important information about the tumor's location, size, shape, and its relationship with surrounding brain structures. This information is vital for treatment decision-making, as it helps determine the optimal approach, such as surgery, radiation therapy, chemotherapy, or a combination thereof. [4]

Moreover, MRI enables the assessment of tumor progression and response to treatment over time. Follow-up MRI scans can provide valuable insights into the effectiveness of therapeutic interventions and guide adjustments to the treatment plan if necessary. Additionally, advanced MRI techniques, such as functional MRI (fMRI) and diffusion tensor imaging (DTI), can provide functional and structural information, respectively, assisting in the mapping of critical brain areas and understanding the impact of the tumor on brain functions.

The integration of MRI findings with clinical data and other diagnostic tests enhances the accuracy of brain cancer diagnosis and aids in the selection of the most appropriate treatment strategy. Furthermore, MRI-guided biopsies allow for precise targeting of tumor tissue, aiding in the determination of tumor type and grade, which in turn influence treatment decisions and prognosis.

In summary, MRI plays an indispensable role in the comprehensive management of brain cancer. Its ability to provide detailed anatomical and functional information about brain tumors enables healthcare professionals to make informed decisions regarding diagnosis, treatment planning, and follow-up care. With ongoing advancements in MRI technology and image analysis techniques, the potential for improved outcomes and enhanced patient care continues to grow, underscoring the significance of MRI in the fight against brain cancer.

1.1 Scope

In this age of information, arguably one of the most useful resources available today is data. From the inception of the internet, as everything is going digital, the amount of data stored has shown a rapid increase, ranging from the most mundane of topics to extremely critical information. This availability of data has ushered in an era where everything can be shared and collaborated on. The healthcare industry is one where this has shown significant effects. This medical information has been utilized for a variety of purposes, including patient follow-up or advice, and innovations in treatment methods. However, advanced data analysis techniques are required such as statistics or machine learning to make the maximum utilization of this information.

Machine learning is the study of how computers learn from data without explicitly being programmed to do so. Deep learning is a machine learning sub-field that uses complex algorithms inspired by human neurons, allowing for the processing of unstructured data such as images, signals, and text. Convolution Neural Network (CNN) and Recurrent Neural Network (RNN) were the first architectures used to process images and text respectively.

By offering an automated, accurate, and reliable detection of brain tumors as well as clearly segmented tumors that may help doctors in rapid diagnosis, along with a 3D model importable in a VR/AR environment that can be used for training and keeping a record of the progress of cancer, this project aims to improve the facilities for diagnosing brain cancer.

1.2 Motivation

Brain cancer is a global health concern, affecting individuals regardless of geographical location or socio-economic background. International Agency for Research on

Cancer (IARC) has reported in Pakistan that the proportion of newly diagnosed cancers is 0.18 million, the number of cancer fatalities is 0.11 million, and the number of prevalent cases (5 years) is 0.32 million [5]. In the case of Pakistan, addressing the challenges posed by brain cancer becomes a matter of utmost importance due to several compelling reasons.

1. High Incidence and Mortality Rates: Brain cancer has been identified as one of the leading causes of cancer-related deaths in Pakistan. The incidence of brain cancer in the country is significant, with a rising trend observed in recent years. Tackling brain cancer becomes crucial to reducing the burden of cancer-related morbidity and mortality in Pakistan.

2. Lack of Awareness and Early Detection: There is a significant lack of awareness among the general public and healthcare professionals in Pakistan regarding brain cancer. As a result, patients often seek medical attention at advanced stages of the disease when treatment options are limited, and outcomes are poor. Raising awareness about brain cancer symptoms, risk factors, and the importance of early detection can lead to improved patient outcomes and survival rates.

3. Limited Access to Specialized Care: Access to specialized healthcare services, including neuro-oncology expertise, advanced imaging facilities, and multidisciplinary treatment teams, is limited in many regions of Pakistan. This disparity in healthcare resources hampers timely diagnosis, accurate staging, and appropriate treatment planning for brain cancer patients. Addressing this issue is essential to ensure equitable access to high-quality care for all patients across the country.

4. Detection of Brain Cancer helps detect other Cancers: The phenomenon of secondary brain cancer where the origination of the tumor is from other parts of the body, indicates that detecting Brain cancer can indicate cancer in other

areas of the body, as the cancerous cells found in the brain would be from originating tumor like lung cells from lung cancer, so can specify exactly where to find the previously undetected tumor.

In conclusion, the motivation for tackling brain cancer in Pakistan stems from the urgent need to reduce the burden of this disease, improve patient outcomes, and bridge the gaps in healthcare resources and awareness. By prioritizing brain cancer research, enhancing access to specialized care, and developing CAD solutions, Pakistan can make significant strides in addressing this challenging disease and ensuring a brighter future for brain cancer patients.

1.3 Objectives

The purpose of this research study is to provide CAD solutions in fields related to brain cancer. This would allow the detection of brain cancer at an early stage, allowing the afflicted individual to be treated in a timely manner, where it is normal for patients to have late diagnoses which have a direct impact on their treatment and the likelihood of survival and recovery. A cost-effective CAD will assist doctors and radiologists in having a second opinion. With a 3D model importable in VR/AR environments, it would help improve the learning experience of prospective doctors and allow easy to visualize track of tumor progress for ease of doctors. The objectives to achieve this goal are as follows:

1. To develop a deep learning-based framework capable of analyzing MRIs to identify different brain tumors and segment them out.
2. To develop a program to generate a 3D model of the brain from MRIs.
3. To detect tumors in 3D view and visualize them.
4. Generating VR/AR of the brain with tumors shown in it.

1.4 Problem Statement

In a country with a total population of more than 233 million people, of which more than 50% don't have access to basic primary healthcare services, and a huge part fail to receive health coverage. Despite the thousands of doctors being produced every year, the country's doctor-patient ratio remains 1 for every 1300 patients. [6] For a specialized field like oncology (study of cancers), the ratio would be even lower. The lack of healthcare facilities, infrastructure, affordability, and accessibility to treatment in Pakistan indicates that Computer Aided Diagnostic (CAD) systems are directly needed.

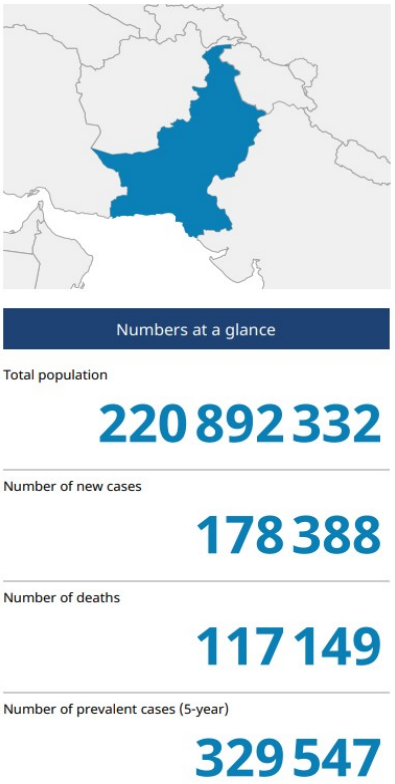

Figure 1.2: Statistics

[3]

Tumor detection using MRIs is a tedious job, one where even a very slight chance of human error would have far-reaching consequences, and a second or perhaps third

opinion would be highly desirable. Using CAD systems would have a significant decrease in how many symptoms are missed, especially where a false-positive which can be correctly diagnosed after scrutiny, would be highly preferred over a false-negative where any tumor missed would have adverse consequences for the patient.

Due to the low number of oncologists, something which can help them save significant time like an easily visualized 3D model that shows tumor and can be used to keep track of the disease's progress would be highly beneficial. Furthermore, as the 3D model would be easily importable in a VR/AR environment, it would also serve as an aid for residents training to be doctors, providing them with virtual experience beforehand.

In this age of information, where science and technology are rapidly growing and evolving, Pakistan's need for adequate health services can only be met by creating innovations that integrate health and technology.

1.5 Structure of Thesis

The structure of thesis is as follows:

Chapter 2 presents a brief overview of cancer and brain cancer, their treatments and the relevant use of AI in their treatment.

Chapter 3 presents the literature analysis based on approaches to tackle issues of brain cancer, the role of MRI scans, and how AI has started to affect the treatments and diagnostics of brain tumors. Furthermore, CNN models are introduced with their working logic and the reason for using the STL format for 3D model generation.

Chapter 4 comprehends the datasets that are utilized in this project along with the specific methodologies that are used to process them to generate the relevant outputs.

Chapter 5 goes in detail over the methodologies mentioned previously with reference to our project.

Chapter 6 concludes the results and delves into future prospects.

Chapter 2

EXPLORING CANCER AND BRAIN CANCER

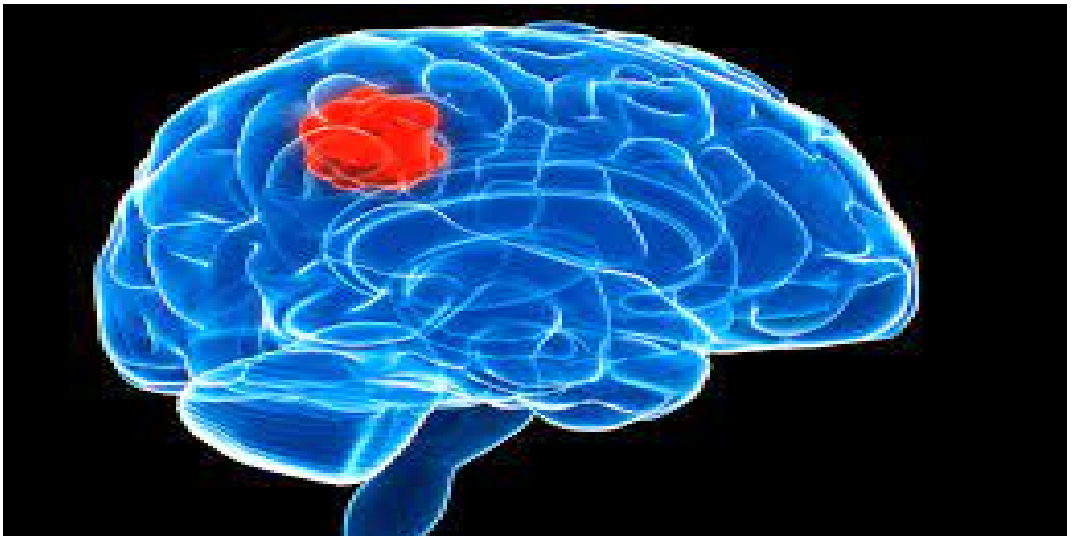

Figure 2.1: Brain Tumor

Cancer is a devastating disease that affects millions of individuals worldwide, posing significant challenges to healthcare systems and impacting the lives of patients and their families. Globally, it is the second leading cause of death with approximately 70% of the deaths from cancer occurring in the lower middle-income countries. It is characterized by uncontrolled cell growth and the ability to invade surrounding

tissues, leading to the formation of malignant tumors. Cancer can arise in various parts of the body, and one specific form, brain cancer, presents unique complexities and considerations.

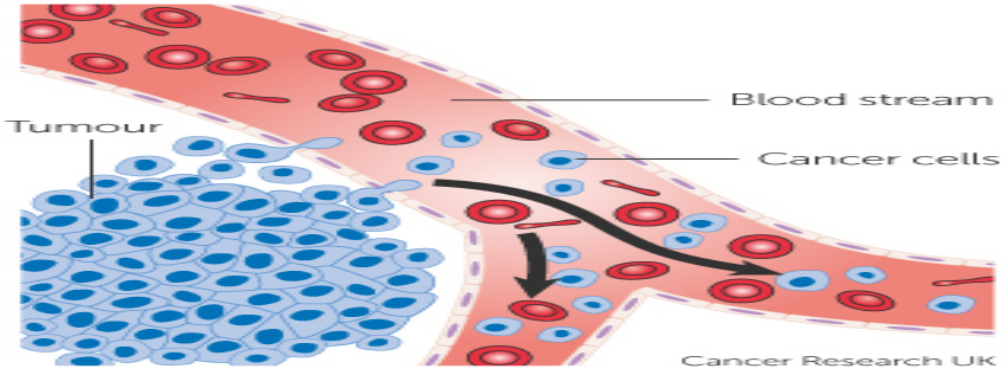

Figure 2.2: Secondary Brain Tumor Working

Brain cancer refers to tumors that originate within the brain or its surrounding structures, such as the meninges, cranial nerves, or pituitary glands. These tumors can be broadly categorized as primary brain tumors, which originate within the brain, or secondary brain tumors, which result from the spread of cancer cells from other parts of the body (metastasis). [7] Primary brain tumors can further be classified based on the types of cells involved, including gliomas (arising from glial cells), meningiomas (arising from the meninges), and medulloblastomas (arising from embryonic cells), with more than 120 different types, differentiated by their occurring location and their cell composition. [8]

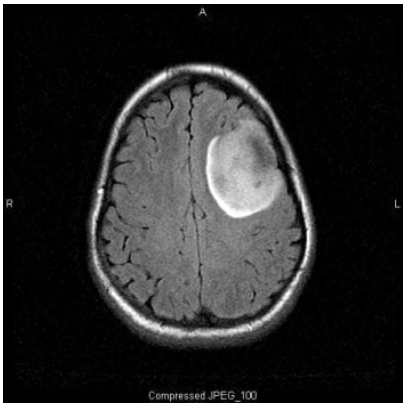

Figure 2.3: Gliomas

[9]

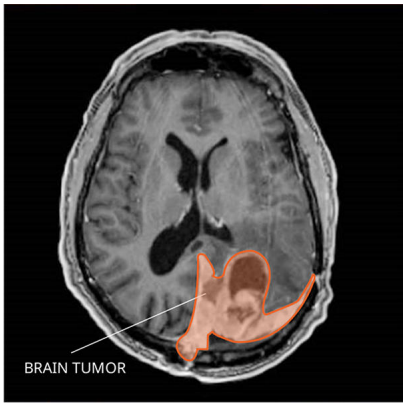

Figure 2.4: Meningiomas

[10]

Figure 2.5: medulloblastomas

[11]

The tumors are also divided into four stages/grades, depending on the tumor growth, likeliness to spread, ease/difficulty of surgical removal, probability of recurring, and various other factors. [12]

Brain cancer poses unique challenges due to the vital functions performed by the brain and its intricate anatomical structures. The presence of a blood-brain barrier, which restricts the entry of many substances into the brain, complicates the delivery of therapeutic agents. Additionally, the brain's complexity and interconnectedness make the surgical removal of brain tumors a delicate and challenging procedure. The

proximity of tumors to critical regions responsible for essential functions, such as motor control, speech, and cognition, requires careful planning and precision during treatment.

Diagnosing brain cancer involves a combination of clinical evaluation, neuroimaging techniques, and tissue sampling. Magnetic resonance imaging (MRI), computed tomography (CT), and positron emission tomography (PET) scans are commonly employed to visualize the tumor's location, size, and potential impact on surrounding brain tissue. Biopsy or surgical resection of the tumor is often necessary to obtain tissue samples for histopathological examination, which helps determine the tumor type and grade.

The treatment of brain cancer depends on various factors, including the tumor type, size, location, and the patient's overall health. The primary treatment modalities for brain cancer include surgery, radiation therapy, and chemotherapy. Surgery aims to remove as much of the tumor as possible while preserving brain functionality. Radiation therapy utilizes high-energy beams to target and kill cancer cells, while chemotherapy uses drugs to destroy cancer cells or inhibit their growth. In recent years, targeted therapy and immunotherapy have emerged as promising approaches for certain types of brain cancer, utilizing drugs that specifically target genetic mutations or enhance the immune system's ability to recognize and attack cancer cells.

Ongoing research efforts in the field of brain cancer focus on improving diagnostic techniques, refining treatment strategies, and developing novel therapies. Advances in molecular profiling, genomics, and biomarker identification contribute to personalized medicine approaches, allowing for tailored treatments based on the specific genetic alterations driving the tumor. Furthermore, the application of artificial intelligence and machine learning algorithms to analyze medical imaging data holds promise for more accurate tumor detection, segmentation, and prognosis.

In conclusion, brain cancer represents a significant challenge in the field of oncology due to its unique complexities and the critical role of the brain. Ongoing research, advancements in diagnostic tools, and the development of innovative treatment modalities offer hope for improved outcomes for patients with brain cancer. A multidisciplinary approach involving collaboration among researchers, clinicians, and industry partners is crucial in advancing our understanding and effectively combating this devastating disease.

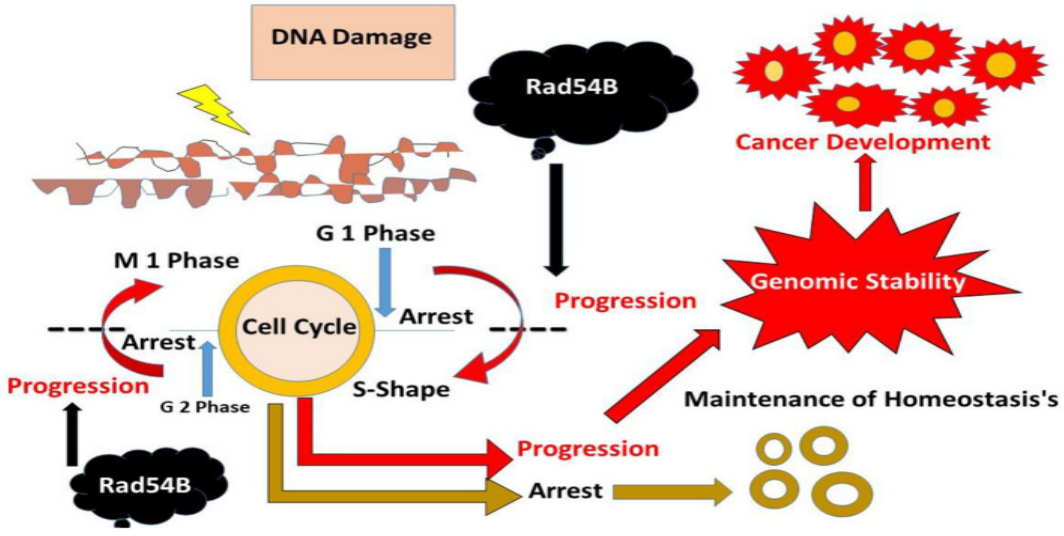

Figure 2.6: Brain Tumor Working

Chapter 3

LITERATURE REVIEW

Magnetic resonance imaging (MRI) is a commonly used test in neurology and neurosurgery, providing comprehensive detail of the brain, spinal cords, and vascular anatomy, allowing views in 3 planes, namely sagittal, axial, and coronal. Compared to a CT scan, an MRI scan can also detect the flow of blood and any malformations that may result in its suboptimal flow. In the past, the processing of MRI scans was a tedious and time-consuming task, but after the introduction of AI in the medical field, the process became easier. AI was first introduced in the 1970s when MYCIN was developed for the diagnosis of blood clotting diseases. [13] [14]

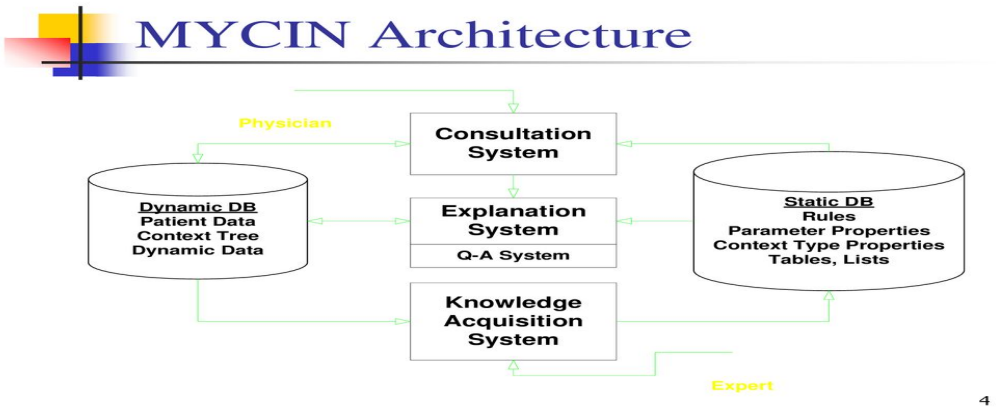

Figure 3.1: MYCIN

[15]

Later, as perfect data was not available, the direction of AI was shifted to be an assisting tool and to build on the expertise of physicians, leading to the development of Artificial Neural Networks, a subfield of which is the Convolutional Neural Network (CNN) [16] architecture, used for processing of data in form of images like MRIs.

These deep learning models enhanced the processing of data, with faster processing, greater accuracy, and better reliability. Every fresh model introduced was either an optimization of the previously released model, or its base model was modified from the previous model. This gradually resulted in multiple models with different working logic, suited to their focus areas.

There are various CNN models with various types of uses, some common types being classifiers and segmenters. The classifier models process the image to detect various features to calculate and reconfigure themselves to reach a desired classification. The following steps are involved in the process.

1. Data Preparation: The first step is to prepare the data for training the classification model. This includes gathering and cleaning the data, handling missing values, and converting categorical variables into numerical representations if necessary. The dataset is typically divided into two subsets: a training set and a test set.

2. Feature Selection/Extraction: In this step, relevant features or attributes are selected or extracted from the dataset. Feature selection helps to reduce dimensionality and focus on the most informative aspects of the data. Feature extraction techniques, such as Principal Component Analysis (PCA), can be used to transform the original features into a lower-dimensional representation.

3. Training the Model: The classification model is trained using the labeled train-

ing dataset. The model learns patterns and relationships between the input features and their corresponding class labels. Popular classification algorithms include logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks.

4. Model Evaluation: After training, the model's performance is evaluated using the test dataset. Evaluation metrics such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve are commonly used to assess the model's predictive performance. Cross-validation techniques, such as k-fold cross-validation, can be employed to obtain more robust performance estimates.

5. Model Tuning: If the initial performance of the model is not satisfactory, hyperparameter tuning can be performed to optimize the model's performance. Hyperparameters are configuration settings that are not learned during training and need to be set beforehand. Techniques like grid search or random search can be employed to find the optimal combination of hyperparameters that yield the best performance. [17]

6. Prediction: Once the model has been trained and evaluated, it can be used to make predictions on new, unseen data. The trained model takes the input features as input and outputs the predicted class label or a probability distribution over multiple classes, depending on the specific classification algorithm used.

The segmenter models process the image to detect various features to calculate and reconfigure themselves to segment out the desired area. The following steps are involved in the process.

1. Data Preparation: The first step is to prepare the training data for the segmenter model. This involves collecting a dataset of labeled images where each

pixel or region is assigned a corresponding class or label. The dataset typically consists of input images and their corresponding segmentation masks, which indicate the ground truth segmentation.

2. Architecture Selection: Choose an appropriate architecture for the segmenter model. Popular architectures include Fully Convolutional Networks (FCNs), U-Net, Mask R-CNN, and DeepLab, among others. These architectures are designed to process the entire input image and produce pixel-wise predictions or region proposals.

3. Model Training: The segmenter model is trained using the labeled training dataset. During training, the model learns to identify patterns and features within the input images that are indicative of the desired segmentation. This is done by optimizing a loss function that measures the discrepancy between the predicted segmentation and the ground truth segmentation. Common loss functions for segmentation include pixel-wise cross-entropy loss, dice coefficient loss, and Jaccard loss.

4. Model Evaluation: After training, the performance of the segmenter model is evaluated using an evaluation dataset or through cross-validation techniques. Evaluation metrics such as Intersection over Union (IoU), Dice coefficient, and pixel accuracy are commonly used to assess the quality of the segmentations produced by the model.

5. Post-processing: Depending on the application and specific requirements, post-processing steps may be applied to refine the segmentation results. This can involve techniques such as thresholding, morphological operations, or connected component analysis to remove noise, smooth the boundaries, or extract specific regions of interest.

6. Inference: Once the segmenter model has been trained and evaluated, it can

be used to segment new, unseen images. The model takes the input image as input and produces a segmentation map or mask, indicating the regions or objects of interest within the image.

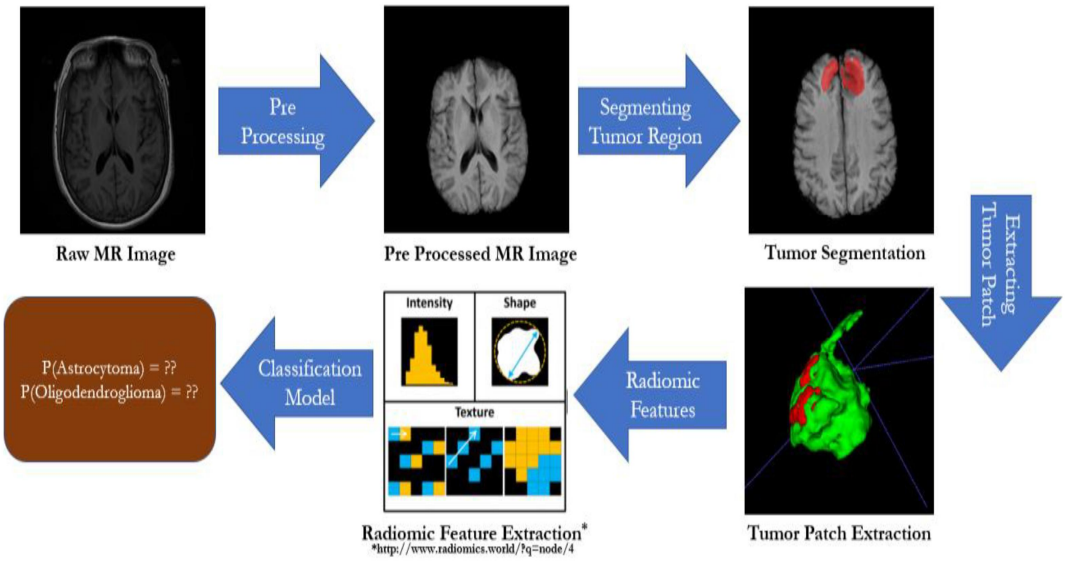

Figure 3.2: Classification and Segmentation Process

Both types of models make use of supervised learning [18] where data with expected output is fed to the model to train it and later while testing uses the refined model and generates output to be matched with actual expected output to calculate accuracy values for the model. Sometimes extra processing needs to be done to avoid cases of overfitting.

For 3D model formation, the STL format [19] is preferred as it uses a series of linked triangles for the description of the 3D model surface geometry. This format is optimal for modeling as it automatically maps the number of triangles required to the complexity of the given shape, increasing its resolution to capture the essence of the model without unnecessary redundant data being stored. STL files are smaller in size yet faster in processing giving them an edge over other formats.

Chapter 4

MATERIALS AND METHODS

In this chapter, we present an overview of the proposed system along with a detailed discussion of the datasets which we have used in this research. The chapter also explains in detail the tools used, as well as the specific AI models that made up our classifier and segmenter architecture.

4.1 Datasets

4.1.1 Kaggle

Some of the MRIs taken from the dataset taken from Kaggle for machine learning.

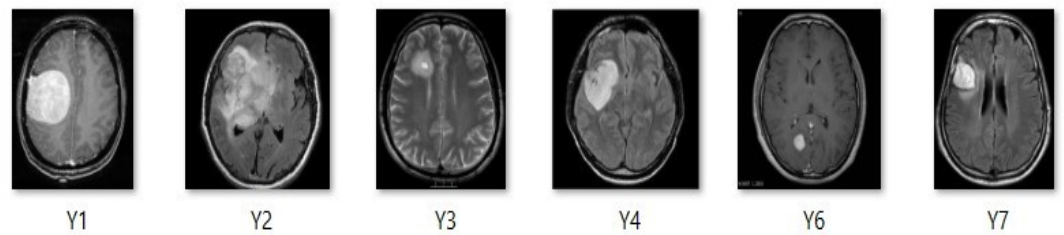

Figure 4.1: Partial dataset from Kaggle

[20, Source of Kaggle Dataset]

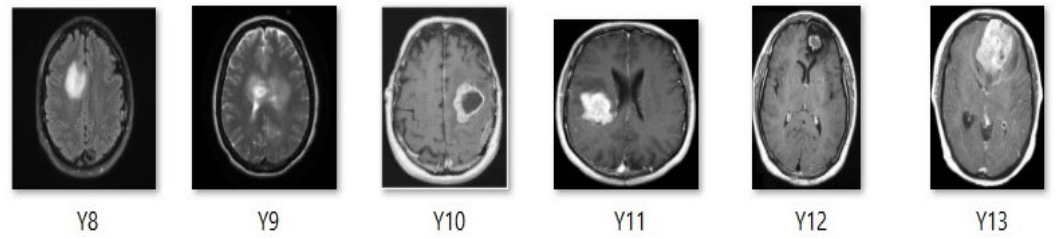

Figure 4.2: Partial dataset from Kaggle

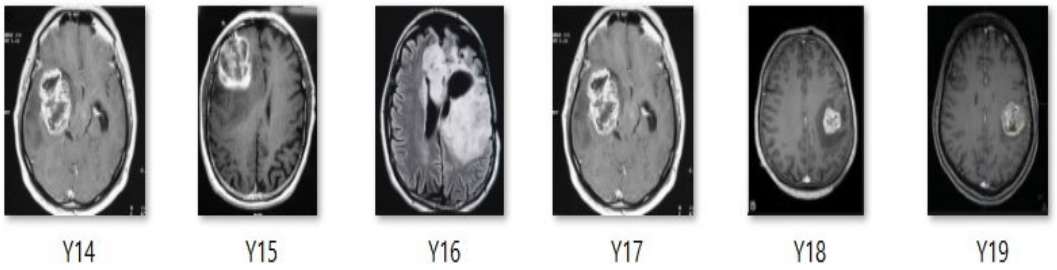

Figure 4.3: Partial dataset from Kaggle

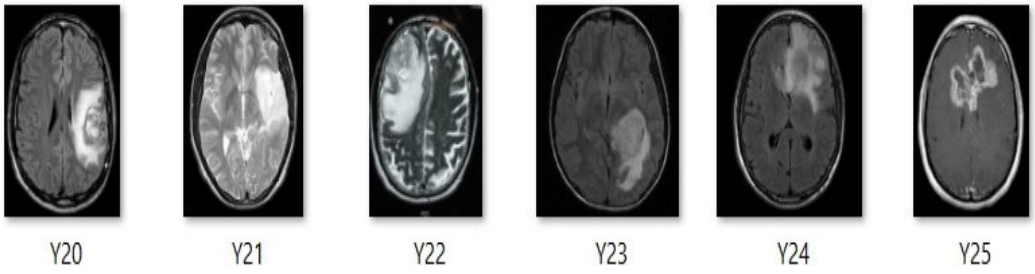

Figure 4.4: Partial dataset from Kaggle

4.1.2 DICOM MRIs

Brain DICOM MRIs, also known as Brain Magnetic Resonance Imaging (MRI) scans in DICOM format, are medical imaging studies used to visualize the internal structures of the brain. DICOM (Digital Imaging and Communications in Medicine) is a standard format for storing, transmitting, and sharing medical images.

Here's a brief explanation of brain DICOM MRIs:

1. Purpose: Brain DICOM MRIs are performed to assess the brain's anatomy, detect abnormalities, and aid in diagnosing various neurological conditions such as tumors, strokes, multiple sclerosis, and other brain disorders.

2. Imaging Technique: Brain MRIs use a powerful magnetic field and radio waves to generate detailed images of the brain. The patient lies inside a large, cylindrical machine called an MRI scanner, which captures the images by measuring the response of hydrogen atoms in the body to the magnetic field.

3. Image Characteristics: Brain DICOM MRIs produce high-resolution, cross-sectional images of the brain. These images provide detailed information about the brain's structures, including the cortex, white matter, ventricles, blood vessels, and any abnormalities or lesions present.

4. DICOM Format: DICOM is a standard format for storing medical images, including brain MRIs. DICOM files contain not only the image data but also important metadata such as patient information, acquisition parameters, and imaging protocols. This format ensures compatibility and interoperability between different imaging systems and software.

5. Visualization and Analysis: Brain DICOM MRIs can be viewed and analyzed using specialized medical imaging software. Radiologists and neurologists interpret the images to identify any abnormalities, measure sizes, analyze tissue characteristics, and plan appropriate treatment strategies.

6. Clinical Applications: Brain DICOM MRIs play a crucial role in diagnosing and monitoring a wide range of neurological conditions. They assist in detecting brain tumors, assessing the extent of trauma or injury, evaluating blood flow and vascular abnormalities, detecting signs of neurodegenerative diseases, and guiding surgical interventions or radiation therapy planning.

7. Multi-Sequence Imaging: Brain MRIs often utilize multiple imaging sequences to provide complementary information. These sequences, such as T1-weighted, T2-weighted, FLAIR, and diffusion-weighted imaging, help visualize different aspects of brain anatomy, tissue characteristics, and pathological changes.

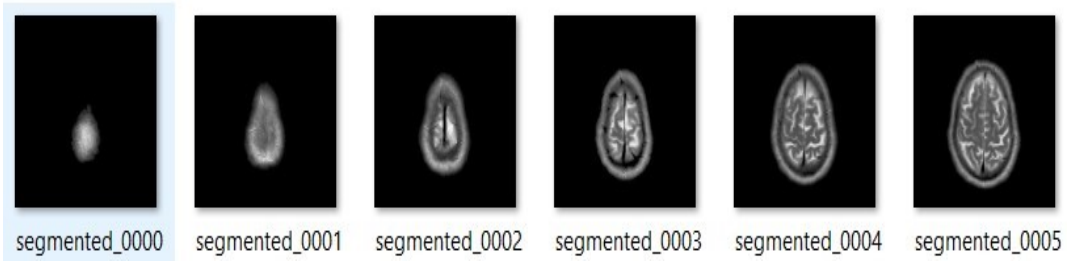

Figure 4.5: DICOM Dataset

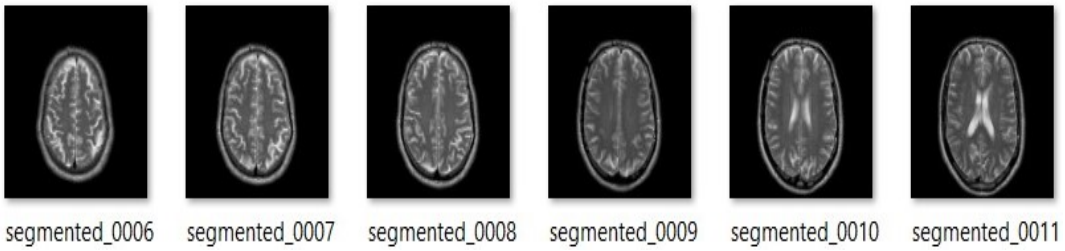

Figure 4.6: DICOM Dataset

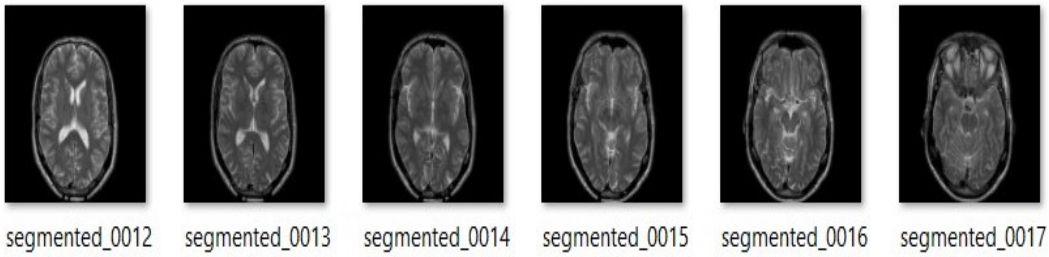

Figure 4.7: DICOM Dataset

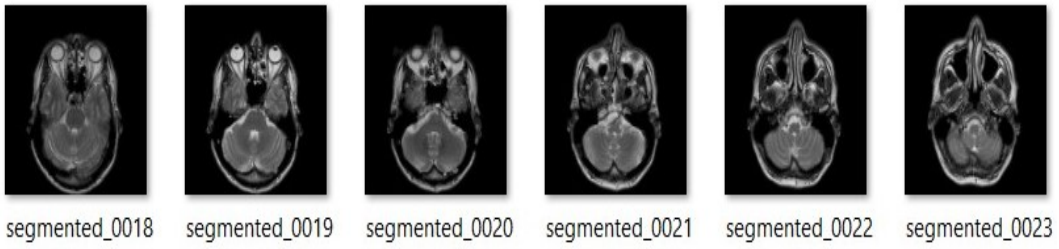

Figure 4.8: DICOM Dataset

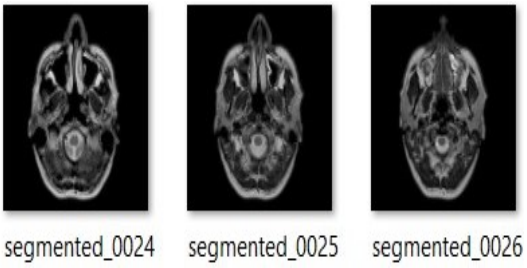

Figure 4.9: DICOM Dataset

4.2 Tools and Methodology

4.2.1 Pycharm

PyCharm, a popular Python IDE, provides a comprehensive development environment for classification and segmentation tasks. With its intelligent code editor, debugging tools, and seamless integration with popular libraries, PyCharm simplifies the development and experimentation process, allowing for efficient implementation and refinement of classification and segmentation algorithms.

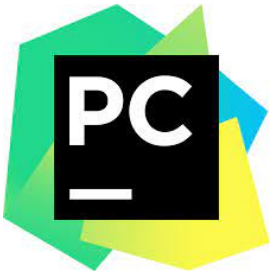

Figure 4.10: Pycharm

4.2.2 Google Colab

Google Colab, a cloud-based Jupyter Notebook platform, is widely utilized for classification and segmentation tasks. It offers free access to powerful GPUs and TPUs, enabling faster model training and inference. Additionally, Colab's integration with Google Drive allows for seamless data storage and collaboration, making it a convenient choice for developing and sharing classification and segmentation projects.

Figure 4.11: Colab

4.2.3 Python

Python is extensively used in classification and segmentation tasks in computer vision, leveraging its rich ecosystem of libraries and frameworks like TensorFlow, Keras, and PyTorch. Its simplicity and versatility make Python a popular choice for efficiently developing and deploying models for accurate object recognition and precise image segmentation.

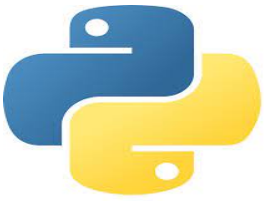

Figure 4.12: Python

4.2.4 Tensor Flow

Tensor flow is a widely used machine learning framework, which offers powerful tools for classification and segmentation, designed for building and training deep learning models, leveraging popular architectures like CNN.

Figure 4.13: TensorFlow

4.2.5 Keras

Keras is a high-level neural network API, which can rapidly build and train deep learning models with a simple and intuitive interface. Its user-friendly design, extensive library of pre-trained models, and seamless integration with TensorFlow make this a core component in using CNN models.

Figure 4.14: Keras

4.2.6 Pandas

Pandas is a powerful open-source library for data manipulation and analysis in Python. It provides data structures like Series and DataFrame, allowing easy handling of structured data. With Pandas, you can perform tasks such as data cleaning, filtering, grouping, and merging, making it ideal for data preprocessing. It also offers comprehensive support for time series data analysis and integrates well with visualization libraries like Matplotlib. Pandas simplifies input/output operations, enabling seamless reading and writing of data in various formats. With its integration with NumPy and Scikit-Learn, Pandas is widely used in data science and machine learning workflows.

Figure 4.15: Pandas

4.2.7 OpenCV

OpenCV (Open Source Computer Vision Library) is a versatile library widely used for image processing, with various common tools to enhance how you can interact with and manipulate image data.

Figure 4.16: OpenCV

4.2.8 Numpy

NumPy provides powerful multi-dimensional array operations and mathematical functions that are essential for handling and manipulating image data, and with efficient array processing capabilities, enable faster data preprocessing, feature extraction, and model evaluation in classification tasks. In segmentation, NumPy facilitates the manipulation and analysis of pixel-wise segmentation masks, allowing for tasks such as calculating metrics, applying post-processing operations, and visualizing results. Its efficient numerical computing capabilities make NumPy an indispensable tool for classification and segmentation workflows.

Figure 4.17: Numpy

4.2.9 MAtplotlib

Matplotlib is a popular Python library for creating visualizations and plots. It provides a comprehensive set of functions and methods to generate a wide range of high-quality 2D and limited 3D plots. Matplotlib is highly customizable, allowing users to create visually appealing and informative graphs.

Figure 4.18: Matplotlib

4.2.10 Converting NPY files to JPG format

We mainly converted NPY (NumPy) files to JPG (JPEG) format for Visualization. NPY files typically store numerical data in a format suitable for scientific computation and analysis. Converting NPY files to JPG allows the visual representation of the data.

4.2.11 Blender

Blender, a versatile 3D modeling software, is commonly used for generating 3D models from arrays and exporting them to STL files. With its array modifiers and powerful modeling tools, Blender allows users to create complex and detailed 3D models, which can then be exported as STL files for use in various applications such as 3D printing or visualization in CAD software like Unity.

Figure 4.19: Blender

4.2.12 Unity

Unity, a popular game development engine, is widely utilized for creating immersive VR experiences with 3D models. With its powerful rendering capabilities, physics simulation, and VR support, Unity enables developers to build interactive and realistic virtual environments that bring 3D models to life, offering users an engaging and interactive VR experience.

Figure 4.20: Unity

4.2.13 Oculus

Oculus, a leading VR platform, seamlessly integrates with Unity to provide users with an immersive experience of seeing and interacting with 3D objects. By utilizing the power of Oculus and Unity, users can explore virtual environments, manipulate 3D objects, and gain a sense of depth and presence, enhancing the realism and interactivity of their VR experiences.

Figure 4.21: Oculus

4.2.14 PIL

PIL (Python Imaging Library) is a popular open-source library for image processing in Python. It provides a wide range of functions and modules for manipulating and analyzing images. The 'PIL.Image' library provides a set of functions and classes for creating, opening, and manipulating images.

Figure 4.22: PIL

4.2.15 OS

The OS libraries in Python provide a range of functions and modules for interacting with the operating system. These libraries allow you to perform various tasks related to file handling, directory operations, process management, and more by providing a powerful set of tools.

4.2.16 STL

STL (Standard Template Library) libraries provide a collection of template classes and functions that offer commonly used data structures and algorithms in Python.

The algorithm library provides a set of functions for performing common algorithms and operations on sequences of elements. The vector library provides a dynamic array implementation, allowing efficient insertion, deletion, and access of elements.

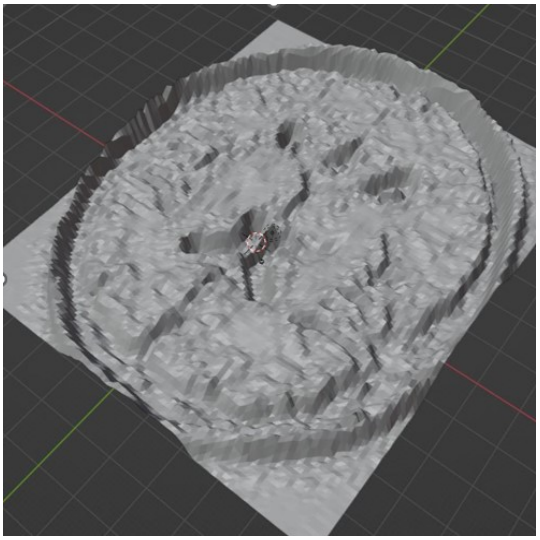

Figure 4.23: 1 layer STL object

4.2.17 OpenMesh

OpenMesh is a versatile and feature-rich library for handling polygonal meshes. It provides efficient data structures and algorithms for representing and processing 3D meshes.

4.2.18 Z Dimension Scaling based on Pixel Intensity

The Z Dimension Scaling based on Pixel Intensity is a technique used to scale the Z (depth) dimension of an image or a 3D volume based on the pixel intensity values. This process allows emphasizing certain features or enhancing the visual representation of the data. Here's an explanation of how this technique works:

1. Purpose: The purpose of Z dimension scaling based on pixel intensity is to modify the depth information of an image or a 3D volume based on the intensity values of the pixels. By scaling the Z dimension, you can enhance or emphasize

specific structures or regions of interest in the data.

2. Input Data: The input data for Z dimension scaling is typically a grayscale image or a 3D volume, where each pixel or voxel has an associated intensity value. The intensity values can range from 0 (black) to the maximum intensity value (white) in the image.

3. Intensity-based Scaling: The scaling process involves mapping the intensity values of the pixels to new Z-coordinate values. Higher intensity values will correspond to higher Z-coordinate values, resulting in the scaling effect. The exact mapping function can be customized based on the desired visual representation or analysis requirements.

4. Scaling Techniques: There are several techniques that can be used for Z dimension scaling based on pixel intensity. Here are a few commonly used methods:

a. Linear Scaling: A linear mapping is applied to scale the Z dimension based on the intensity values. For example, the Z-coordinate of each pixel can be multiplied by a scaling factor that is proportional to the intensity value.

b. Non-linear Scaling: Non-linear functions can be used to map intensity values to Z-coordinate values. This allows for more complex mappings, such as logarithmic scaling, exponential scaling, or custom transfer functions.

c. Threshold-based Scaling: Specific intensity ranges can be defined as thresholds, and the Z dimension scaling can be applied selectively based on these thresholds. For example, pixels with intensity values above a certain threshold can be scaled differently from those below the threshold.

d. Adaptive Scaling: Adaptive scaling techniques can be employed to dynamically adjust the scaling based on local intensity variations. This approach can help highlight subtle features or details in the data.

5. Visualization and Analysis: Z dimension scaling based on pixel intensity can

be useful for visualizing and analyzing volumetric data in various fields, including medical imaging, scientific visualization, and computer graphics. By manipulating the depth dimension, specific features can be brought into focus or exaggerated for better visualization or analysis purposes.

4.2.19 Stacking of STL Files

The purpose of stacking STL files for 3D modeling is to combine multiple individual models or components into a single cohesive model. Stacking allows you to assemble complex 3D structures by merging or aligning individual STL files. Here are some key reasons for stacking STL files:

1. Assembly of Multi-Part Objects: When a 3D object consists of multiple parts or components, each component can be designed and saved as a separate STL file. Stacking allows you to bring these individual components together and assemble them into a complete object.

2. Composite Objects: By stacking STL files, you can create composite objects by merging or combining different models. This is where various components are combined to create a single, integrated structure.

3. Customization and Variation: Stacking STL files enables customization and variation in 3D models. By stacking different variations or options of individual parts, you can create models with interchangeable components or modular designs.

4. Complex Geometries: Stacking STL files can be used to create intricate or complex geometries that are challenging to design as a single model.

Overall, stacking STL files provides flexibility, modularity, and the ability to create complex 3D structures by combining individual components.

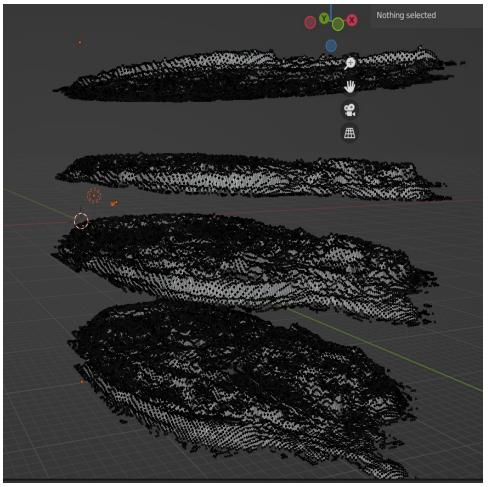

Figure 4.24: Stacked STL layers

4.2.20 OBJ formation from STL Files

The purpose of representing an OBJ file as a collection of STL files is to decompose the complex 3D model in OBJ format into individual STL files. Each STL file represents a separate component or object within the overall structure. This approach offers several benefits, such as ease of management, modularity, and the ability to handle large or complex models efficiently.

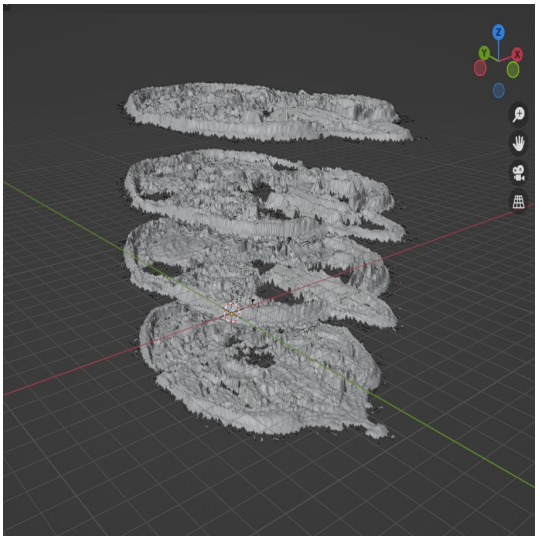

Figure 4.25: Stacked STL layers as object

4.2.21 BPY

The bpy libraries are a collection of Python modules that provide a powerful API (Application Programming Interface) for interacting with and controlling Blender, a popular open-source 3D creation suite. The bpy libraries allow you to automate tasks, create custom tools, and extend the functionality of Blender using Python scripting.

The bpy libraries offer extensive functionality for interacting with Blender and automating tasks through Python scripting. By utilizing these libraries, you can create custom tools, automate repetitive tasks, extend Blender's capabilities, and integrate Blender with other software and pipelines. The Blender Python API documentation provides more detailed information about the bpy libraries and their usage. The bpy libraries in Blender provide a comprehensive set of tools and functionalities for editing, vertex and face manipulation, texturing, coloring, and scaling OBJ files through Python scripting. The bpy library also allows you to import objects and organize them within a parent-child hierarchy, essential for objects with various levels of sub-objects.

The bpy library can also be used to change the transparency of an OBJ file in Blender's Python API, allowing inner visualization of a solid object.

Figure 4.26: BPY

4.2.22 PYWavefront

PyWavefront is a Python library that allows you to work with Wavefront OBJ files. It provides a convenient and easy-to-use interface for loading, accessing, and manipulating 3D models stored in the OBJ format.

4.2.23 VGG16

VGG16 is a convolutional neural network architecture that was developed by the Visual Geometry Group (VGG) at the University of Oxford. It is widely used for image classification tasks in computer vision. VGG16 consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. It is known for its simplicity and uniform structure, with small 3x3 filters throughout the network. VGG16 has achieved excellent performance on various benchmark datasets and has become a popular choice as a pre-trained model for transfer learning in deep learning applications.

We added custom layers on top of the VGG16 base model to create the final classification model. The GlobalAveragePooling2D layer is applied to convert the 2D feature maps into a 1D vector. Then, two Dense layers with ReLU and sigmoid activations are added for classification.

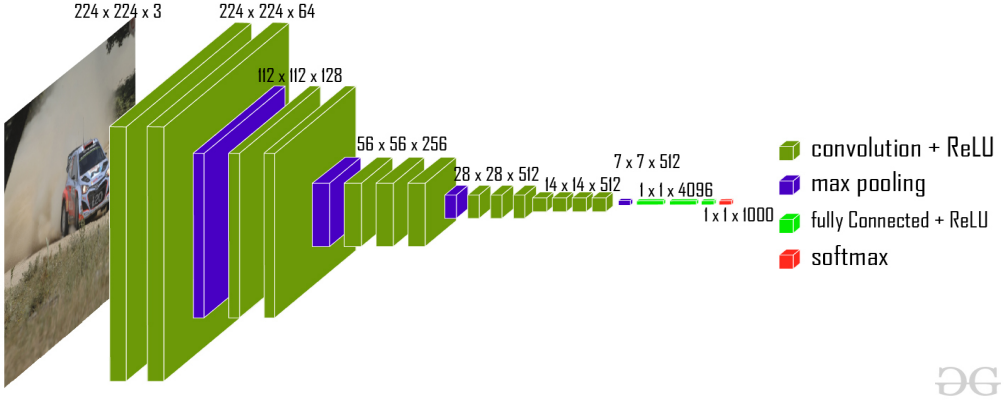

Figure 4.27: VGG16 general working

[21]

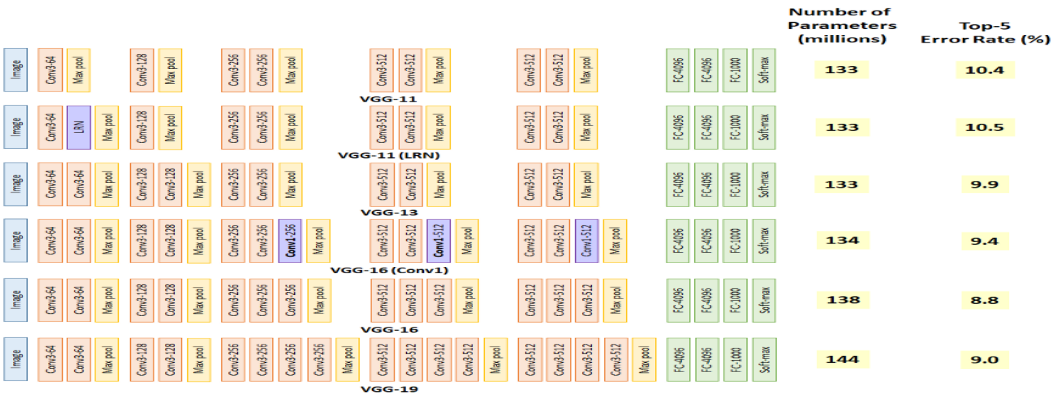

Figure 4.28: VGG general working

4.2.24 U-Net

The model consists of a contracting path (encoder) and an expanding path (decoder) with skip connections. Each level in the contracting path consists of two convolutional layers followed by max pooling and dropout. Each level in the expanding path consists of up-sampling, concatenation with the corresponding skip connection, and two convolutional layers. The final layer uses a 1x1 convolution with sigmoid activation to output the predicted segmentation mask. The model is compiled with the Adam optimizer and binary cross-entropy loss.

In our modified U-Net model, in the encoder part, there are 5 blocks of convolutional layers and max pooling. Each block starts with two 3x3 convolutional layers with ReLU activation, followed by a 2x2 max pooling operation to reduce the spatial dimensions. Additionally, a dropout layer with a rate of 0.5 is applied after each max pooling operation to prevent overfitting.

In our modified U-Net model, in the decoder part, there are 4 blocks of upsampling and convolutional layers. Each block starts with an upsampling layer (using UpSampling2D) to increase the spatial dimensions by a factor of 2, followed by a 2x2 convolutional layer with ReLU activation. The output from the previous decoder block is concatenated with the corresponding encoder block's output using skip connections. After concatenation, two 3x3 convolutional layers with ReLU activation

are applied to refine the features.

The final convolutional layer has 1 filter with a 1x1 kernel and a sigmoid activation function. It produces a segmentation mask that represents the probability of each pixel belonging to the target class (binary segmentation task). The output has the same spatial dimensions as the input image.

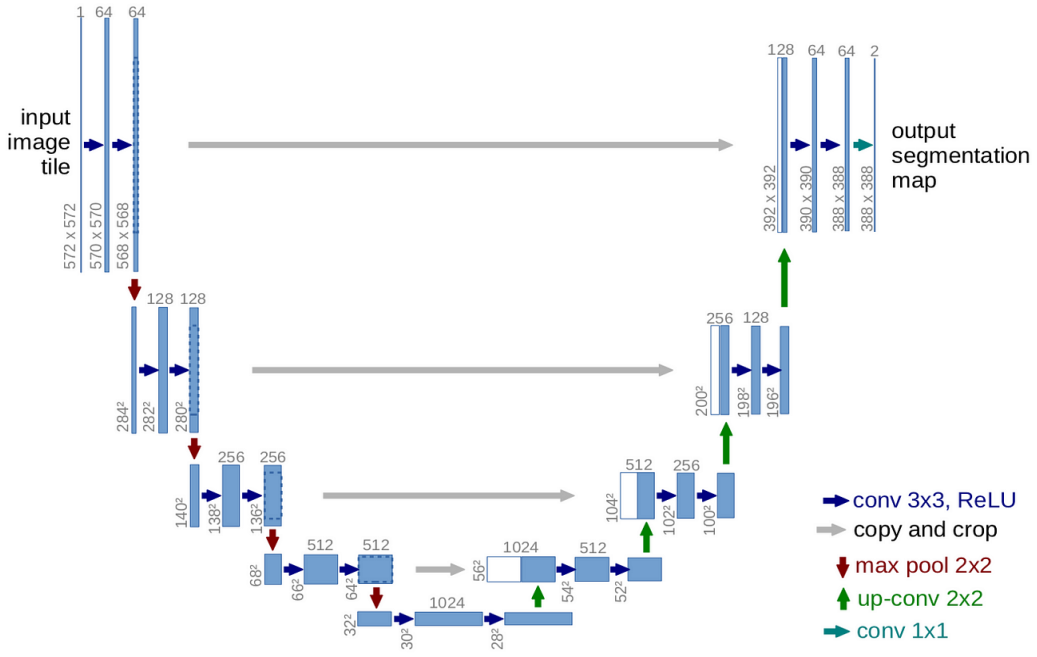

Figure 4.29: U-Net genereal working

Chapter 5

CLASSIFICATION, SEGMENTATION AND 3D MODEL GENERATION FOR USE IN VR/AR

The following diagram shows the general relationship between the dataset, the machine learning section, the 3D model generation section, and finally the output Model that could be displayed in VR/AR environment.

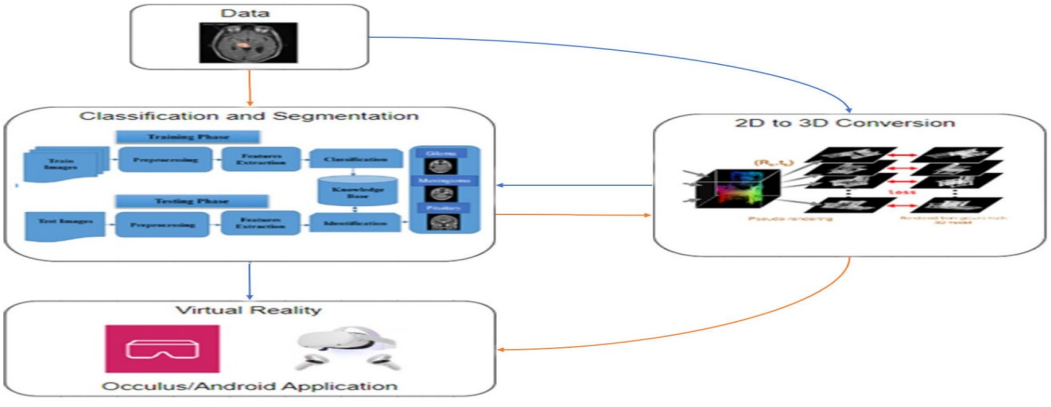

Figure 5.1: Flow Diagram of working of the project

5.1 Classification and Segmentation

5.1.1 Classification using modified VGG16

We used Google Colab for classification and segmentation processing, as machine learning algorithms require vast processing power to be trained in a timely manner. Google Colab also provided a platform where we can modularly test out our code when implementing, allowing us to quickly sort out errors and see outputs alongside the code.

We first gathered a dataset of brain tumor images, in our case DICOM files, and categorized them as either containing brain tumors or not. We later divided them into two sets of training data and testing data. For effective training of the model, our dataset contained several diverse images, allowing us to avoid overfitting cases. The express purpose of using DICOM files was how it allowed us to get the exact number of tumor-containing layers in the testing images from the DICOM file of the patients.

We then set up Google Colab using the free resources available, giving us a good quantity of RAM and GPU resources. We imported TensorFlow, Keras, and other additional libraries as mentioned in the previous chapter and installed them. We then connected the notebook to a GPU runtime when we executed the code to allow faster processing, while minimizing the wastage of the GPU remaining idle.

Before training our modified VGG16 model, we preprocessed the brain tumor images, resizing the images to a uniform size, normalizing pixel values, and potentially applying data augmentation techniques such as rotation, scaling, or flipping to increase the diversity of the training data and improve the model's generalization capability.

Our VGG16 had some custom layers over the usual 16 layers, which included 13 con-

volutional layers and 3 fully connected layers, with small 3x3 filters in use throughout the network. Our custom layers include the GlobalAveragePooling2D layer which is applied to convert the 2D feature maps into a 1D vector, then two Dense layers with ReLU and sigmoid activations added for classification.

Our preprocessed dataset had been divided into testing and validation groups beforehand. We trained the VGG16 model on the training set using batch training and ran it for a specified number of epochs using the fit function and displayed training and validation metrics after each epoch. We monitored the model's performance on the validation set and adjusted hyperparameters, such as learning rate and batch size, to improve training accuracy and prevent overfitting.

After training, we evaluated the model's performance using the test set. We calculated classification metrics such as accuracy and precision scores to assess the model's ability to correctly classify brain tumor images. We visualized the results using confusion matrices or ROC curves to gain insights into the model's performance across different classes. We also set the model to output the predicted class and actual classes of some random images.

Once the VGG16 model was trained and evaluated, we saved all the weights and parameters for future use. By leveraging the power of the VGG16 model and the convenience of Google Colab, the classification of brain tumors became more accessible and efficient. This approach empowered us to develop accurate and reliable models for automated brain tumor diagnosis, potentially improving the speed and accuracy of medical decision-making.

5.1.2 Segmentation using modified U-Net

For segmentation, many of the steps are quite similar to the classification parts mentioned above.

We gathered the same dataset of brain tumor images, in our case DICOM files, and

categorized them as original images and the tumor-segmented masks. These masks indicated the exact boundaries of tumor regions in the corresponding MRI images. We later divided them into two sets of training data and testing data. For effective training of the model, our dataset contained several diverse images, allowing us to avoid overfitting cases. We not only used the segmentor for individual images, but also in the images for the 3D model generation for accurate segmentation of tumor in each layer of the 3D model.

We then set up Google Colab using the free resources available, giving us a good quantity of RAM and GPU resources. We imported TensorFlow, Keras, and other additional libraries as mentioned in the previous chapter and installed them. We then connected the notebook to a GPU runtime when we executed the code to allow faster processing, while minimizing the wastage of the GPU remaining idle.

Before training our modified U-Net model, we preprocessed the brain tumor images and corresponding segmentation masks, resizing them to a uniform size, normalizing pixel values, and potentially applying data augmentation techniques such as rotation, scaling, or flipping to increase the diversity of the training data and improve the model's generalization capability.

Our U-Net consisted of a contracting path (encoder) and an expanding path (decoder) with skip connections. Each level in the contracting path consists of two convolutional layers followed by max pooling and dropout. Each level in the expanding path consists of up-sampling, concatenation with the corresponding skip connection, and two convolutional layers. The final layer uses a 1x1 convolution with sigmoid activation to output the predicted segmentation mask. The model is compiled with the Adam optimizer and binary cross-entropy loss.

In the encoder part, we increased to 5 blocks of convolutional layers and max pooling. Each block starts with two 3x3 convolutional layers with ReLU activation, followed by a 2x2 max pooling operation to reduce the spatial dimensions. Additionally,

a dropout layer with a rate of 0.5 is applied after each max pooling operation to prevent overfitting.

In the decoder part, we increased to 4 blocks of upsampling and convolutional layers. Each block starts with an upsampling layer (using UpSampling2D) to increase the spatial dimensions by a factor of 2, followed by a 2x2 convolutional layer with ReLU activation. The output from the previous decoder block is concatenated with the corresponding encoder block's output using skip connections. After concatenation, two 3x3 convolutional layers with ReLU activation are applied to refine the features.

The final convolutional layer has 1 filter with a 1x1 kernel and a sigmoid activation function. It produces a segmentation mask that represents the probability of each pixel belonging to the target class (binary segmentation task). The output has the same spatial dimensions as the input image.

Our preprocessed dataset had been divided into testing and validation groups beforehand. We trained the U-Net model on the training set using batch training and appropriate loss functions such as dice coefficient or cross-entropy loss and ran it for a specified number of epochs using the fit function and displayed training and validation metrics after each epoch. We monitored the model's performance on the validation set and fine-tune hyperparameters, such as learning rate and batch size, to improve training accuracy and prevent overfitting. The tumor regions in the original image and output image are highlighted by changing their colors to red and displayed side by side.

After training, we evaluated the model's performance using the test set. We calculated segmentation metrics such as Dice Coefficient to assess the model's accuracy to correctly segment brain tumor images. We visualized the results against the original masks to gain insights into the model's performance. Post-processing techniques such as thresholding, morphological operations, or connected component analysis may be applied to refine the segmentation results if deemed necessary.

Once the U-Net model was trained and evaluated, we saved all the weights and parameters for future use. By leveraging the power of the U-Net model and the capabilities of Google Colab, the segmentation of brain tumors became more accessible and efficient. This approach empowered us to develop accurate and reliable models for automated brain tumor segmentation, potentially improving the speed and accuracy of medical decision-making.

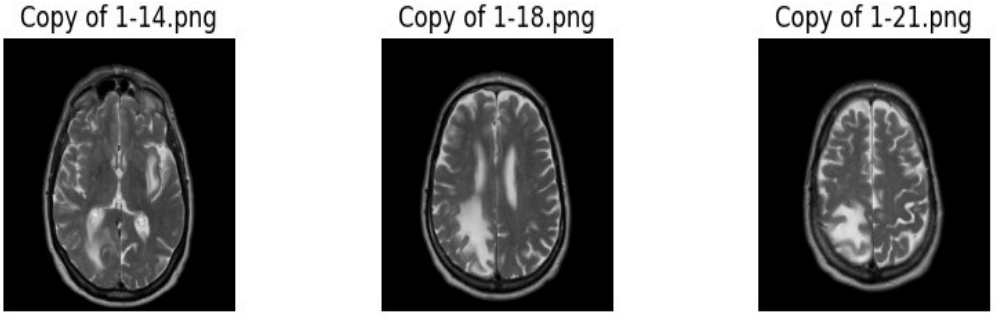

Figure 5.2: Segmented Images

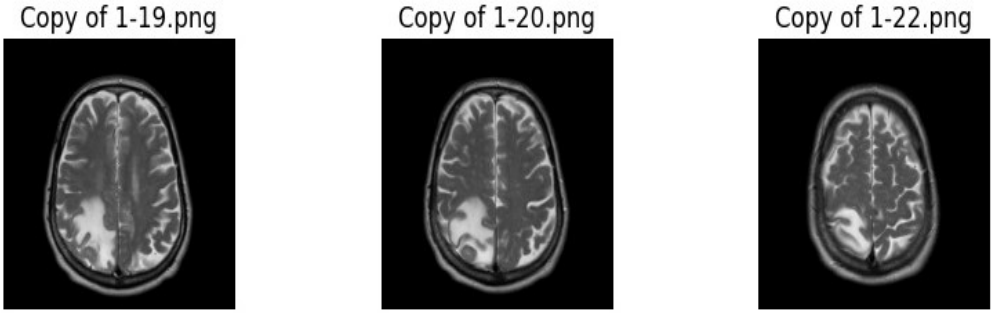

Figure 5.3: Segmented Images

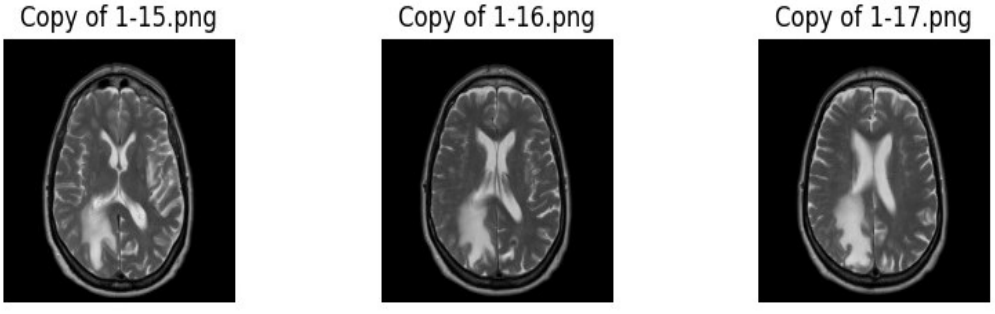

Figure 5.4: Segmented Images

```
Epoch 1/10
22/22 [==============================] - 590s 26s/step - loss: 0.5536 - accuracy: 0.7145 - val_loss: 0.3461 - val_accuracy: 0.8070
Epoch 2/10
22/22 [==============================] - 623s 28s/step - loss: 0.3307 - accuracy: 0.8463 - val_loss: 0.2792 - val_accuracy: 0.8421
Epoch 3/10
22/22 [==============================] - 608s 28s/step - loss: 0.2430 - accuracy: 0.8990 - val_loss: 0.3036 - val_accuracy: 0.8655
Epoch 4/10
22/22 [==============================] - 576s 26s/step - loss: 0.1929 - accuracy: 0.9209 - val_loss: 0.2309 - val_accuracy: 0.8830
Epoch 5/10
22/22 [==============================] - 606s 28s/step - loss: 0.1700 - accuracy: 0.9414 - val_loss: 0.1688 - val_accuracy: 0.9532
Epoch 6/10
22/22 [==============================] - 572s 26s/step - loss: 0.1345 - accuracy: 0.9575 - val_loss: 0.1741 - val_accuracy: 0.9006
Epoch 7/10
22/22 [==============================] - 605s 28s/step - loss: 0.1135 - accuracy: 0.9561 - val_loss: 0.1271 - val_accuracy: 0.9591
Epoch 8/10
22/22 [==============================] - 605s 28s/step - loss: 0.0851 - accuracy: 0.9693 - val_loss: 0.1316 - val_accuracy: 0.9591
Epoch 9/10
22/22 [==============================] - 606s 28s/step - loss: 0.0623 - accuracy: 0.9795 - val_loss: 0.1314 - val_accuracy: 0.9591
Epoch 10/10
22/22 [==============================] - 604s 28s/step - loss: 0.0876 - accuracy: 0.9693 - val_loss: 0.1949 - val_accuracy: 0.9357
<keras.callbacks.History at 0x7f96f7705240>
```

Figure 5.5: Classifying Accuracy

Type of Accuracy	Percentage (%)
Training Accuracy	97%
Validation Accuracy	93%

Table 5.1: Accuracy of Classification Model VGG16

Type of Accuracy	Percentage (%)
Training Accuracy	93%
Validation Accuracy	91%

Table 5.2: Accuracy of Segmentation Model U-Net

5.2 3D Model Generation

[22] [23]

For the 3D model generation, we again made use of Google Colab for processing. We created 3D models of the brain layer by layer from MRI images and stacked them using Google Colab, which involved several steps and the utilization of various libraries such as numpy, PIL (Python Imaging Library), os, stl, bpy (Blender Python API), and pywavefront. Here is a detailed explanation of the process:

1. We started by gathering a complete set of segmented MRI images of the brain from our previous classification and segmentation models. These images in DICOM format, covered the entire brain region with consistent orientation and resolution, and as DICOM format images, were easily processed in Python.

2. We setup our colab sheet by importing all the libraries to be used like numpy for efficient numerical operations, PIL for image processing tasks, os for file and directory operations, stl for creating and exporting STL files, bpy for interacting with Blender, and pywavefront for importing 3D object files.

3. We processed the MRI images to enhance their quality and make them suitable for 3D modeling. This included steps such as rescaling pixel intensities, applying contrast enhancement techniques, and performing image registration or normalization to align the images.

4. After image processing is completed, we stacked them using user input distance between each adjacent layer. We used numpy to stack them to turn them into a 3D volume. This stacking process created a 3D representation of the brain, where each pixel corresponds to a voxel in 3D space. We ensured that the voxel size and dimensions were accurately preserved during the stacking process.

5. We then converted the stacked 3D volume into a mesh representation using libraries like stl or bpy. Using STL, we iterated over the voxels, determining the presence of brain tissue, and generating corresponding vertices and triangles to create a mesh representation.

6. We then exported the generated mesh as an STL file, a common format for 3D printing and rendering by using libraries like stl or bpy, which provided functions to save the mesh data in the STL format.

7. After importing the generated STL file into Blender using bpy or pywavefront. We used the powerful 3D modeling environment that allowed us to visualize, refine, and further manipulate the brain model to our desired outcome.

8. We saved the final 3D model in the desired format, OBJ [24] using bpy or other relevant libraries. The OBJ file allowed us to have the whole model with the induvial slices in STL format as sub-objects within the whole object, allowing easy access to

each individual slice as a separate object. This also facilitated in creation of a parent and child hierarchy, useful to keep the brain and its tumor as separate objects while keeping them aligned.

9. We imported the final 3D model in OBJ format to Unity, where we tested the model in the virtual environment, using Unity-provided support. This enabled us to test for both future VR and AR use of our generated model. Unity also allowed us to code for the object to be able to be scaled, rotated and moved for use in both the VR environments like Oculus and in AR environment applications on our mobile devices. The AR would facilitate low-cost usage of our 3D model and allow it to be visualized in the real world. The VR would provide a more focused and detailed visualization of the 3D model and may be allowed to familiarize and manipulate it in more depth.

The following 3D models show how the stacked STL images have been made into one object that can be imported into the VR/AR environment.

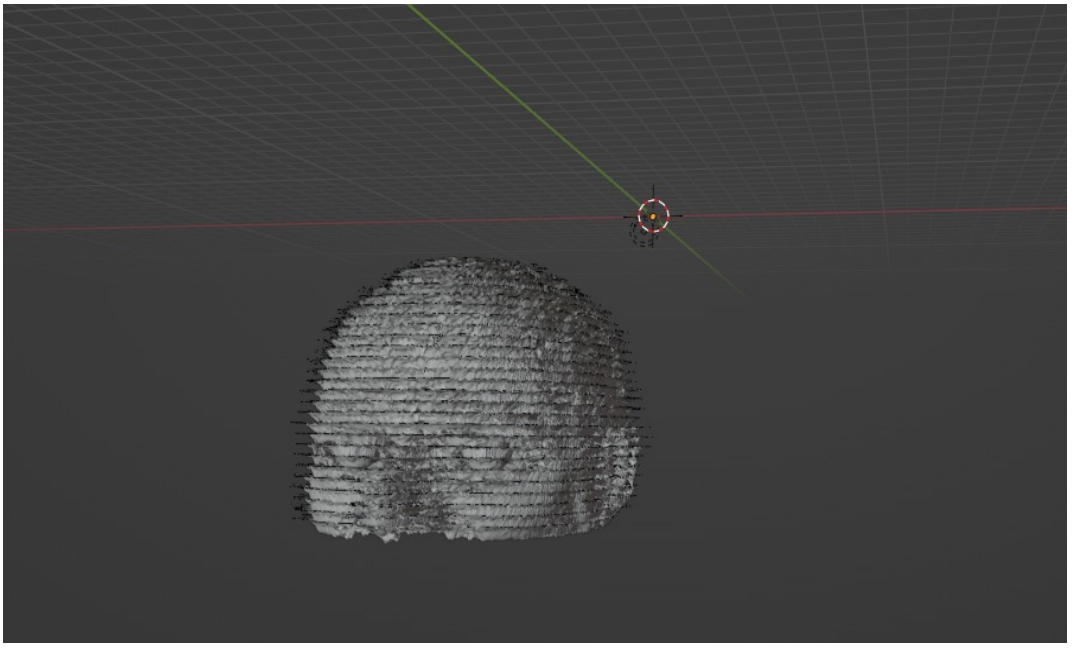

Figure 5.6: 3D Model

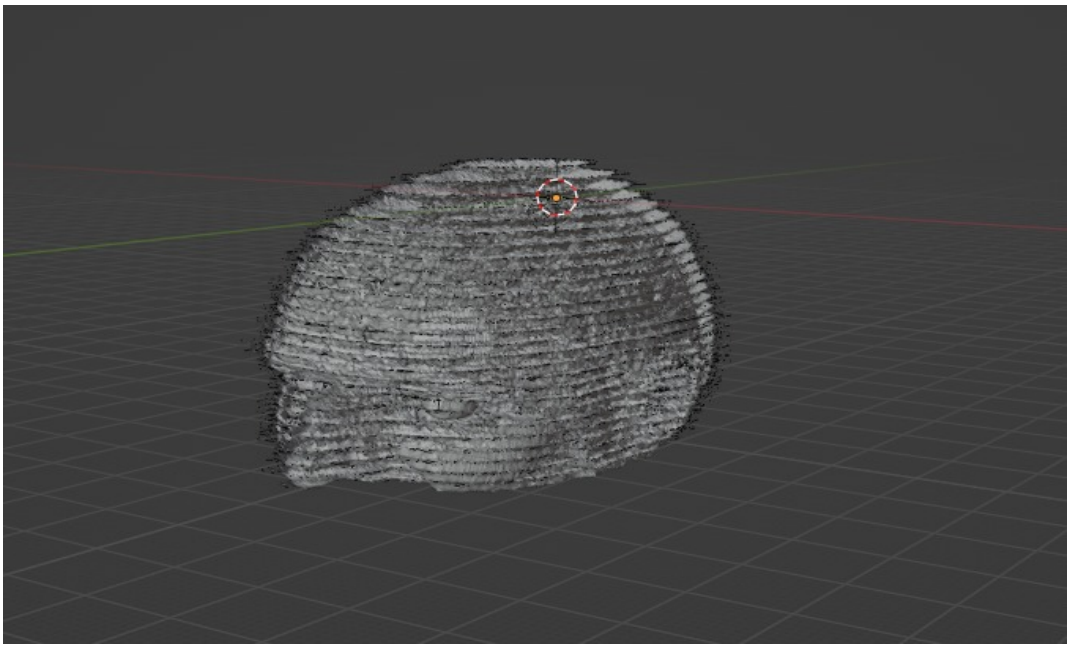

Figure 5.7: 3D Model

Chapter 6

CONCLUSION AND FUTURE PROSPECTS

6.1 Conclusion

The Project, Brain Tumor Detection and Visualization in VR/AR, developed a solution where a helping tool allows doctor and medical staff a cost-effective and quick solution to meet the needs of the population without compromising on quality, as well as providing a learning model that can be used to train future generation of doctors in a manner where valuable experience could be gained without requiring real patients. The solution can also be used by any layman to test their MRIs at home in a very cost-effective manner, saving him both time and money.

6.2 Future Prospects

The project has a great chance of worldwide use as it aims to assist radiologists and doctors in their job. Since there is minimal hardware involved, an AR environment can be easily assessed by existing hardware if VR hardware is not cost-effective to procure, the costs for commercializing it would mainly consist of maintaining

servers to meet calculation and processing requirements, introducing it on a global scale would be quite viable. Also, the product being used in the medical field, which is perpetual, would keep the demand high for this kind of product. As machine learning technologies are in a mode of constant improvement and innovation, there would always be better and more efficient models and solutions arising from the previously developed models, this project would also be improved upon. This would raise the patient's chances of survival and complete recovery, where perhaps one day, eradicating cancer would not be a far-fetched dream. As this solution is mostly concerned with brain cancer, perhaps there would be similar solutions being developed for other cancers and diseases, which in turn could be visualized together in a 3D human model to improve our knowledge about how each part affects and is being affected by others. This may improve treatments as doctors would be able to see previously seemingly unrelated factors, in a new light.

Bibliography

[1] N. Zahid, W. Khalid, K. Ahmad, S. S. Bhamani, I. Azam, N. Asad, A. A. Jabbar, M. Khan, and A. Enam, "Resilience and quality of life (qol) of head and neck cancer and brain tumour survivors in pakistan: an analytical cross-sectional study protocol," 2019. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6756343/.

[2] Cancer.Net, "Brain tumor: Statistics." https://www.cancer.net/cancer-types/brain-tumor/statistics, 2023.

[3] Globocan, "586-pakistan-fact-sheets," 2020. https://gco.iarc.fr/today/data/factsheets/populations/586-pakistan-fact-sheets.pdf.

[4] D. C. Preston, "Magnetic resonance imaging (mri) of the brain and spine: Basics." https://case.edu/med/neurology/NR/MRI, 2016.

[5] A. Ali, M. F. Manzoor, N. Ahmad, R. M. Aadil, H. Qin, R. Siddique, S. Riaz, A. Ahmad, S. A. Korma, W. Khalid, and L. Aizhong, "The burden of cancer, government strategic policies, and challenges in pakistan: A comprehensive review," 2022.

[6] S. A. Khan, "Situation analysis of health care system of pakistan: Post 18 amendments," vol. 7, 2019.

[7] C. R. UK, "What is secondary brain cancer?." https://www.cancerresearchuk.org/about-cancer/secondary-cancer/

secondary-brain-cancer/about, 2020.

[8] J. H. Medicine, "Brain tumor types." https://www.hopkinsmedicine.org/health/conditions-and-diseases/brain-tumor/brain-tumor-types.

[9] M. H. Bilsky, "Gliomas." https://www.msdmanuals.com/en-sg/professional/neurologic-disorders/intracranial-and-spinal-tumors/gliomas.

[10] NIH, "Meningioma diagnosis and treatment." https://www.cancer.gov/rare-brain-spine-tumor/tumors/meningioma.

[11] S. J. Cloud, "Medulloblastoma." https://pbtp.stjude.cloud/diseases/medulloblastoma.

[12] M. Markman and CityofHope, "Brain cancer grades." https://www.cancercenter.com/cancer-types/brain-cancer/grades, 2022.

[13] Wikipedia, "Artificial intelligence in healthcare." https://en.wikipedia.org/wiki/Artificial_intelligence_in_healthcare.

[14] Wikipedia, "Mycin." https://en.wikipedia.org/wiki/Mycin.

[15] L. Johnston, "Mycin." https://slideplayer.com/slide/14936744/.

[16] A. H. Y. Abyaneh, A. H. Gharari, and V. Pourahmadi, "Deep neural networks meet csi-based authentication," 2018.

[17] DominoDataLab, "Model tuning." https://www.dominodatalab.com/data-science-dictionary/model-tuning.

[18] Wikipedia, "U-net." https://en.wikipedia.org/wiki/U-Net.

[19] Adobe, "Stl files." https://www.adobe.com/creativecloud/file-types/image/vector/stl-file.html.

[20] kaggle, "Brain mri images for brain tumor detection." https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection?resource=download, 2020.

[21] B. QoChuk, "Different types of cnn models." https://iq.opengenus.org/different-types-of-cnn-models/.

[22] H. Hogberg, J. Bressler, and K. Christian, "Toward a 3d model of human brain development for studying gene/environment interactions.." https://stemcellres.biomedcentral.com/articles/10.1186/scrt365, 2013.

[23] Google, "3d style transfer." https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/differentiable-parameterizations/style_transfer_3d.ipynb, 2018.

[24] S. Schechter, "Obj file." https://www.marxentlabs.com/obj-files/, 2020.