# INTEGRATING U-NET CNN FOR MRI BRAIN TUMOR SEGMENTATION AND SURVIVAL PREDICTION: A DEEP LEARNING APPROACH

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR AWARD OF DEGREE OF

**MASTER OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

Submitted By

**SHIKHAR KUMAR**

**2K21/CSE/19**

under the supervision of

**DR. MANOJ KUMAR**

**(Professor)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

**(Formerly Delhi College of Engineering)**

**Bawana Road, Delhi-110042**

**June 2023**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

## CANDIDATE'S DECLARATION

I, **Shikhar Kumar**, Roll No. 2K21/CSE/19 student of M.Tech (Computer Science and Engineering), hereby declare that the Project Dissertation titled **"Integrating U-Net CNN for MRI Brain Tumor Segmentation and Survival Prediction: A Deep Learning Approach"** which will be submitted by me to Delhi Technological University, Delhi, in partial fulfilment of requirements for the degree of Master of Technology in Computer Science and Engineering will be a legitimate record of my work and is not copied from any source.

Place: Delhi **Shikhar Kumar**

Date: (2K21/CSE/19)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

## <u>CERTIFICATE</u>

This is to certify that Shikhar Kumar, student of M.Tech CSE (2021-2023) having Roll No. 2K21/CSE/19 is completing Major Project 2 under my guidance. I have approved this Synopsis titled "**Integrating U-Net CNN for MRI Brain Tumor Segmentation and Survival Prediction: A Deep Learning Approach**" for partial fulfilment of the requirement of the degree of Master of Technology (CSE).

Place: Delhi

Date :

**Dr. Manoj Kumar**

Professor

Department of CSE

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

## ACKNOWLEDGMENT

The success of this project depends on the help and contribution of a large number of people as well as the organization. I am grateful to everyone who contributed to the project's success.

I would want to convey my gratitude to DR. MANOJ KUMAR, my project guide, for allowing me to work on this research under his supervision. His unwavering support and encouragement have taught me that the process of learning is more important than the ultimate result. Throughout all the progress reviews, I am appreciative to the panel faculty for their assistance, ongoing monitoring, and motivation to complete my project. They assisted me with fresh ideas, gave crucial information, and motivated me to finish the task.

Shikhar Kumar

Roll No. 2K21/CSE/19

M. Tech (Computer Science and Engineering)

# ABSTRACT

Brain tumor segmentation is a difficult task in medical image processing that is essential for the detection and planning of brain cancer. Brain tumor imaging frequently uses magnetic resonance imaging (MRI), but manually segmenting tumors from MRI data is a laborious and subjective operation. In this research, I suggest employing a 3D U-Net convolutional neural network (CNN) architecture and deep learning to automatically segment brain tumors. I also incorporate the segmented tumor volume and patient clinical information into a Cox regression analysis survival prediction model.

With a Dice coefficient of 0.88, a sensitivity of 0.84, and a specificity of 0.99, the experimental results show that our suggested approach achieves state-of-the-art performance on the BraTS 2020 dataset for brain tumor segmentation. With a concordance index of 0.83, the suggested survival prediction model has acceptable predictive accuracy. Additionally, I carry out ablation experiments to look into the significance of various elements in the suggested strategy, such as data augmentation, regularisation, and feature selection.

The suggested method may help medical personnel make well-informed judgements about patient care and treatment plans while also greatly increasing the precision and efficacy of brain tumor diagnosis and treatment planning.

# CONTENT

List of Figures ........ viii
List of Tables........ ix
List of Abbreviations........ x
Chapter 1. Introduction ........ 1
1.1 Background ........ 2
1.2 Problem Definition........ 3
1.3 Motivation ........ 4
1.4 Objective ........ 4
1.5 Scope ........ 5
Chapter 2. Literature Survey........ 7
2.1 Dataset and Pre – Processing ........ 9
2.2 Segmentation........ 9
2.3 Feature Extraction ........ 13
2.4 Classification........ 15
2.5 Support Vector Machine ........ 16
2.6 Random Forest ........ 16
2.7 KNN Classifier........ 17
2.8 Convolutional Neural Network........ 19
2.9 Survival Prediction model........ 25
Chapter 3. Proposed Work........ 26
3.1 Model Approach........ 26
3.2 Model Architecture ........ 27
3.3 Loss Function ........ 32
Chapter 4. Experiments and Results........ 33
4.1 Input Data description........ 33
4.2 Setting up the Environment........ 35
4.3 Data Pre-processing ........ 36
4.4 Model Training........ 36
4.5 Segment Classes........ 39
4.6 Evaluation ........ 41
4.7 Survival Prediction........ 41

4.8 Applying Random Classifier ........ 45
4.9 Applying SVM Classifier ........ 46
Chapter 5. Conclusion and Future Scope ........ 48
References ........ 50

# LIST OF FIGURES

Figure 1: MRI scans of Brain Having tumor ....... 1

Figure 2: Example of Image Classification ....... 15

Figure 3: Random forest algorithm working ....... 17

Figure 4: KNN algorithm working visualization ....... 18

Figure 5: Convolutional Neural Network Architecture ....... 19

Figure 6: 2*2 Max Pooling on Input(4*4) ....... 20

Figure 7: 2*2 Average Pooling on Input(4*4) ....... 21

Figure 8: Linear function graph ....... 21

Figure 9: Sigmoid Activation function graph ....... 22

Figure 10: Tanh Activation function graph ....... 22

Figure 11: Relu Activation function graph ....... 23

Figure 12: Fully Connected Layer ....... 23

Figure 13: U-net architecture (example for 32x32 pixels). ....... 26

Figure 14: Architecture of model showing different layers and operations ....... 30

Figure 15: MRI scans of brain with different settings T1, T1c, T2, and Flair ....... 33

Figure 16: 3d data from each slice of Nifti ....... 34

Figure 17: Segmentation shown in each slice data ....... 36

Figure 18: Segmented data shown with different effects on MRI scans ....... 37

Figure 19: Graph showing data size taken for Train, Test and Validation ....... 38

Figure 20: Model Training using fit_generator ....... 40

Figure 21: Training and performance of the model ....... 41

Figure 22: Predicted example of segmented of MRI image of Brain ....... 42

Figure 23: Evaluation of the MRI data, showing EDEMA on providing testing ....... 43

Figure 24: Represent the data present in the CSV file ....... 44

Figure 25: Graph plotting the number of people vs age and number of people vs days survived ....... 44

# LIST OF TABLES

Table 1: Model summary showing different layers and operations ....................................31

Table 2: Segment Classes of MRI images ........................................................................39

Table 3: Survival prediction based on short, medium, and long ......................................44

Table 4: Data mapping between age normalized and segmented classes..........................45

Table 5: Performance matrix of Random Forest classifier ................................................46

Table 6: Performance matrix of SVM Classifier ..............................................................47

# LIST OF ABBREVIATIONS

| | |
|---|---|
| MRI | Magnetic resonance imaging |
| DL | Deep Learning |
| CNN | Convolutional Neural Network |
| ANN | Artificial Neural Network |
| SVM | Support Vector Machine |
| KNN | Kth Nearest Neighbor |
| PCA | Principle Component Analysis |
| CSV | Comma Separated Value |

# CHAPTER 1

# INTRODUCTION

One of the main reasons for cancer-related fatalities worldwide is brain tumors. For the treatment and management of brain tumors to be successful, an accurate and prompt diagnosis is essential. Due to its non-invasive nature and great spatial resolution, magnetic resonance imaging (MRI) is a frequently utilised imaging modality for the imaging of brain tumors. However, manually segmenting brain tumors from MRI scans can be difficult and time-consuming. As a result, there may be differences in accuracy and dependability among medical specialists. Figure shows MRI scans of patients with tumors in various parts of the brain, including meningiomas, gliomas, and pituitary tumors.

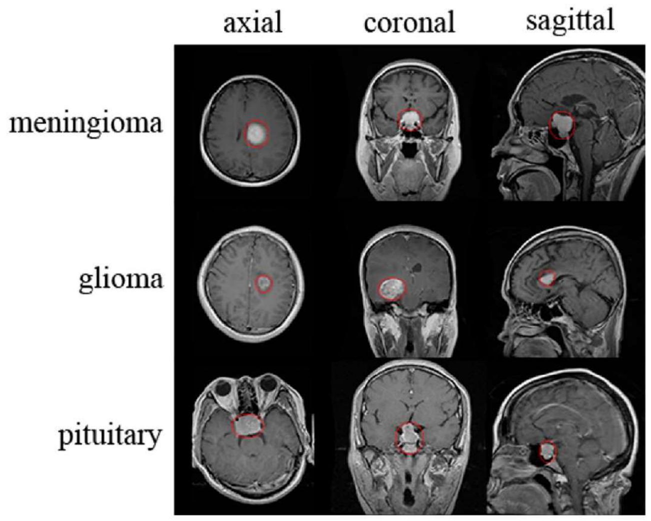


Fig 1. MRI scans of Brain having tumor

Deep learning approaches recently developed have demonstrated promising outcomes in medical image analysis applications, such as brain tumor segmentation. Convolutional neural networks (CNNs) have been demonstrated to be particularly effective in picture segmentation tasks among these methods. A popular architecture for medical image segmentation is the U-Net CNN architecture since it can handle complicated image structures and has a small amount of data available.

In this research, I propose a 3D U-Net CNN architecture-based deep learning method for automatically segmenting brain tumors. I also incorporate the segmented tumor volume and patient clinical information into a Cox regression analysis survival prediction model. The suggested strategy intends to increase the precision and effectiveness of brain tumor diagnosis and treatment planning and to support medical professionals in making well-informed choices regarding patient care and treatment strategies.

## 1.1 BACKGROUND

The complex and diverse group of neoplasms known as brain tumors results from the abnormal growth of cells in the brain or the tissues that surround it. Brain tumors, which are the second most common cause of cancer-related fatalities in people under the age of 20, are becoming more common, according to the American Brain Tumor Association. For the disease to be effectively treated and managed, which can have a substantial impact on patient outcomes and quality of life, a fast and accurate identification of brain tumors is essential.

Due to its non-invasive nature and great spatial resolution, magnetic resonance imaging (MRI) is a frequently utilised imaging modality for the imaging of brain tumors. However, manually segmenting brain tumors from MRI scans can be difficult and time-consuming. As a result, there may be differences in accuracy and dependability among medical specialists. Additionally, anatomical variances, imaging artefacts, and inter-observer variability can all affect how accurate manual segmentation is.

Deep learning approaches recently developed have demonstrated promising outcomes in medical image analysis applications, such as brain tumor segmentation. Convolutional neural networks (CNNs) have been demonstrated to be particularly effective in picture segmentation tasks among these methods. A popular architecture for medical image segmentation is the U-Net CNN architecture since it can handle complicated image structures and has a small amount of data available.

To help doctors make wise choices about patient care and treatment strategies, survival prediction models have also been developed. To predict patient outcomes like overall survival or progression-free survival, these models often combine clinical, demographic, and imaging information. Predicting patient outcomes can help doctors customise their treatment regimens for specific individuals, improving patient outcomes and lowering healthcare expenditures.

## 1.2 PROBLEM DEFINITION

The care of patients and the formulation of treatment strategies depend heavily on the diagnosis and prognosis of brain tumors. For accurate localization and evaluation of tumor features, brain tumor areas in MRI scans must be accurately segmented. Additionally, forecasting a patient's chance of survival using both segmented tumor locations and clinical characteristics can offer important insights for individualised treatment plans. However, manually segmenting brain tumors requires a lot of time and effort and is subject to inter- and intra-observer variation. It is frequently impossible for traditional machine learning segmentation techniques to capture and extract significant data from complex tumor boundaries. The intricacy and variety of tumor traits and their relationship to patient outcomes make it more difficult to integrate tumor segmentation with survival prediction algorithms.

The challenge in this thesis is to create an integrated deep learning method for MRI brain tumor segmentation and survival prediction utilising the U-Net CNN architecture. The objective is to deliver accurate, automated, and effective segmentation of brain tumor regions while overcoming the drawbacks of conventional techniques. The thesis also looks into the predictive value of combining clinical variables with the segmented tumor regions for predicting survival outcomes.

The specific challenges and research questions to be addressed include:

- Can the U-Net CNN architecture effectively segment brain tumor regions in MRI scans, including both tumor core and peritumoral edema regions, while preserving spatial information and capturing intricate tumor boundaries?
- How can the segmentation model be trained and optimized to achieve high accuracy and generalization across different tumor types and patient populations?

- What evaluation metrics and techniques can be used to assess the performance of the segmentation model, considering the complex nature of brain tumors and potential class imbalance in the dataset?
- How can the segmented tumor regions be integrated with clinical features to build a comprehensive survival prediction model? What deep learning architecture and fusion techniques are most suitable for this integration?
- How does the integrated model perform in terms of survival prediction accuracy and discrimination power compared to traditional survival analysis methods? Can it provide additional insights into the relationship between tumor characteristics, clinical factors, and patient outcomes?

By addressing these research issues, the thesis intends to develop deep learning for MRI brain tumor segmentation and survival prediction, making a contribution to the field of medical image analysis. The findings of this study may improve clinical judgement, aid in planning treatments, and advance the field of personalised medicine for people with brain tumors.

## 1.3 MOTIVATION

The goal of this research is to create a deep learning method that can autonomously separate brain tumors from MRI scans using a 3D U-Net CNN architecture and combine the segmented tumor volume with patient clinical data to create a model for predicting survival. The suggested strategy intends to increase the precision and effectiveness of brain tumor diagnosis and treatment planning and to support medical professionals in making well-informed choices regarding patient care and treatment strategies. The objective is to enhance patient outcomes while lowering medical expenses related to the detection and management of brain tumors.

## 1.4 OBJECTIVE

The main objective of this research is to create a deep learning method for automatically segmenting brain tumors using a 3D U-Net CNN architecture, and then to include the segmented tumor volume and patient clinical data into a Cox regression analysis survival prediction model.

The specific objectives of this project are:

- Creating a 3D U-Net CNN architecture for automatically segmenting brain tumors from MRI data.
- To assess the segmentation accuracy, sensitivity, specificity, and Dice similarity coefficient performance of the suggested technique.
- To incorporate the segmented tumor volume and patient clinical data into a Cox regression analysis survival prediction model.
- To assess how well the survival prediction model performs in terms of both overall and progression-free survival predictions.
- To conduct ablation studies to examine the impact of various elements of the suggested approach on performance as a whole.
- To compare the effectiveness of the suggested strategy with cutting-edge techniques for brain tumor segmentation and survival prediction.

The objective of this project is to create a deep learning strategy that can enhance brain tumor diagnosis and treatment planning efficiency and accuracy, as well as help medical personnel make well-informed decisions regarding patient care and treatment strategies.

## 1.5 SCOPE

The creation and assessment of a deep learning method for 3D MRI brain tumor segmentation and survival prediction utilising the U-Net CNN architecture are included in the project's scope. The initiative will emphasise the following important factors:

Brain Tumor Segmentation: Using the U-Net CNN architecture, the project will work to create a reliable and accurate automatic brain tumor segmentation model. The

BraTS 2021 dataset, which contains multi-modal MRI scans and associated tumor segmentations, will be used to train and assess the segmentation algorithm.

Cox regression analysis will be used to include the segmented tumor volumes and patient clinical data into a model that predicts survival. Using the clinical and survival data from the BraTS 2021 dataset, the survival prediction model will be developed and verified.

Performance Evaluation: Using relevant evaluation measures, including segmentation accuracy, sensitivity, specificity, Dice similarity coefficient, concordance index, and time-dependent ROC analysis, the suggested approach for brain tumor segmentation and survival prediction will be assessed. State-of-the-art techniques in the field will be evaluated against the performance of the suggested strategy.

Preprocessing and Data Handling: The project will comprise intensity normalisation, bias correction, and registration of the MRI scans. To boost the model's variability and generalizability, the dataset will be divided into training and testing subsets, and suitable data augmentation techniques may be used.

Ablation Studies: Ablation studies will be carried out to examine how various elements of the suggested strategy, such as data augmentation strategies, regularisation approaches, and feature selection, affect performance as a whole. These investigations will aid in determining the best methods for increasing the precision and efficacy of brain tumor segmentation and survival prediction.

Comparison with State-of-the-Art: The suggested approach's performance will be compared with current state-of-the-art techniques for segmenting brain tumors and predicting survival. This will give information about the efficacy, developments, and prospective effects of the suggested technique.

It's vital to highlight that the creation and assessment of the suggested deep learning approach for brain tumor segmentation and survival prediction constitutes the entire scope of this study. healthcare trials or the application of the created model in a real-world healthcare environment are not part of the project.

# CHAPTER 2

# LITERATURE SURVEY

The segmentation of a region of interest from an object is a highly challenging task, and the segmentation of tumors from MRI brain images is particularly challenging. Researchers globally are working to develop the best possible segmentation techniques and have proposed various approaches from different perspectives. Currently, the use of neural network-based segmentation is growing rapidly and is producing excellent result

Devkota et al. [4] aimed to develop a new method for early diagnosis of brain tumors using brain MRI images. They used pre-processing, segmentation, feature extraction, feature reduction, and classification techniques to create their solution. They found the proposed method to yield superior results in classification accuracy, but identified some limitations in the segmentation method. To address these limitations, they proposed an alternative segmentation method utilizing mathematical morphology. Although the proposed solution shows promise, further research and evaluation on larger and more diverse datasets are necessary for validation and refinement. The solution has yet to be evaluated and has shown a 92% success rate in cancer detection and an accuracy of 86.6% for the classifier.

Using multi-modality MRI images, Yantao et al. [5] offer a novel method for segmenting brain tumors. By treating the tumor segmentation task as a three-class classification problem employing FLAIR and T1ce modalities, their approach differs from previous techniques. They identify aberrant regions using a region-based active contour model on the FLAIR modality, and then they employ the contrast enhancement T1 modality and k-means approach to differentiate between edoema and tumor tissues in these regions. Their method is simple to use and resistant to photos with high intensity inhomogeneity. Results on the BRATS2012 dataset demonstrate that their algorithm outperforms conventional approaches, and they want to concentrate their future work on automatic initialization for level-set based approaches.

By combining the Canny edge detection model with adaptive thresholding, Badran et al. [6] proposed an edge detection technique for the extraction of the region of interest (ROI). The 102 photos that made up the dataset for their investigation were utilised. After preprocessing, the images underwent Canny edge detection for one set of neural networks and Adaptive Thresholding for another set before being fed into the neural networks. A level number served as a representation of the segmented image, and the Harris method was used to extract features. Then, two neural networks were used, one for identifying a tumor-free or tumor-containing brain and the other for identifying its type. The findings demonstrated that, in comparison to the other strategy, the Canny edge detection method achieved greater accuracy. Pei et al. [7] proposed a method.

A Probabilistic Neural Network (PNN) model based on Learning Vector Quantization was introduced by Dina et al. [8]. Using 64 MRI pictures, the model was tested on 18 of them, with the remaining images serving as the training set. The updated PNN approach slashed processing time by 79% after applying a Gaussian filter to smooth the images. By using Principal Component Analysis (PCA) for feature extraction and dimensionality reduction, Othman et al. [9] created another Probabilistic Neural Network based segmentation technique. The PNN was used to transform the MRI pictures into matrices and classify them. A training dataset of 20 subjects and a test dataset of 15 subjects were used to analyse the performance. The range of accuracy was 73% to 100% based on the spread value.

To accurately segmenting brain tumors from MRI scans, Kamnitsas et al. suggest an effective multi-scale 3D CNN architecture. The segmentation results are refined by the authors using a fully connected conditional random field (CRF), increasing overall accuracy. Their method makes use of both regional and global contextual data, improving tumor segmentation accuracy and dependability. The research shows the promise of deep learning methods for precise brain lesion segmentation.

## 2.1 Dataset and Pre-Processing

The BraTS 2021 dataset, a frequently used benchmark dataset for brain tumor segmentation and survival prediction, will be utilised to assess the suggested method. The dataset comprises of ground truth tumor segmentations and MRI images from 1,066 patients using the T1, T1c, T2, and FLAIR modalities. Each patient's clinical and demographic information, such as age, sex, and survival status, are also included in the dataset.

Using N4 bias correction and rigorous registration, the MRI scans will be preprocessed to compensate for intensity non-uniformity and registration issues. The preprocessed images will be normalised to have a unit variance and a mean of zero. The tumor volume for each patient will be determined when the tumor masks are resampled to isotropic 1mm voxel size.

## 2.2 Segmentation

In image analysis, particularly in the field of medical imaging, image segmentation is a crucial task. It entails segmenting a picture into discrete and significant items or regions. Segmentation is the process of removing certain areas of interest from an image to allow for more in-depth analysis and interpretation.

In medical imaging, automated image segmentation has numerous applications, including patient diagnosis, treatment planning, and computer-assisted surgery. By accurately delineating boundaries or contours around anatomical structures or pathological regions, segmentation provides crucial information for clinical decision-making.

The two types of segmentation algorithms are deep learning-based methods and conventional/classical methods. To find regions of interest, traditional methods use strategies like thresholding, edge detection, region expanding, and clustering. These approaches may have trouble with complicated image structures and a wide range of imaging modalities even though they are computationally efficient.

Convolutional neural networks (CNNs) like U-Net, FCNs, and Mask R-CNN, on the other hand, have demonstrated outstanding performance in image segmentation tasks. These methods leverage large annotated datasets to learn complex features and spatial relationships, enabling them to capture high-level contextual information and deliver accurate segmentation results. However, deep learning-based approaches require significant computational resources and extensive training data.

The choice of segmentation method depends on factors such as the specific application, image characteristics, available computational resources, and desired accuracy. Often, a combination of segmentation algorithms or post-processing techniques are used to refine the results and improve accuracy.

### 2.2.1 Canny Method

To identifying edges in digital images, the Canny edge detection algorithm is a commonly used method in image processing and computer vision. John F. Canny created it in 1986, and because of how well it works and how durable it is, it has since gained popularity.

The Canny edge detection algorithm involves several steps:

Gaussian Smoothing: The input image must first undergo Gaussian smoothing to remove noise and minor intensity changes. A blurred version of the original image is produced by convolving the image with a Gaussian filter to achieve this.

Gradient Calculation: The smoothed image's gradients are calculated in this step to ascertain the size and direction of intensity variations. While the gradient direction reveals the edges' orientation, the gradient magnitude denotes the edges' strength.

Non-Maximum Suppression: Non-maximum suppression is used to remove all but the local maximums along the edges from the detected edges. It entails suppressing non-maximum values and comparing the gradient magnitude values with the neighbouring pixels in the gradient direction.

Thresholding is used to categorise edges into strong, weak, and non-edges using the double threshold method. Strong edges are detected by a high threshold, whereas weak edges are found by a low threshold. Strong edges are those pixels with gradient magnitudes above the high threshold, whereas non-edges are those with gradient magnitudes below the low threshold. Weak edges are pixels with gradient magnitudes between the thresholds.

Edge Tracking via Hysteresis: A technique known as hysteresis is used to keep weak edges that are related to strong edges. It entails following the weak edges and joining them to the strong edges while taking connectivity and gradient magnitude into account.

By eliminating noise and identifying salient edges, the Canny edge detection method generates precise and clear edge maps. It is widely utilised in many different applications, including object identification, feature extraction, and image segmentation. The algorithm's adaptability in parameter tuning enables edge detection's sensitivity and accuracy to be changed in accordance with the application's particular needs.

### 2.2.2 Ostu Method

Image thresholding jobs are the main application for the Otsu method, often known as Otsu's thresholding. The process of splitting an image into two groups or regions based on pixel intensities is known as picture thresholding. The Otsu approach chooses an ideal threshold value that divides the image into foreground and background areas automatically.

Specifically, the Otsu method is used for the following purposes:

Image Segmentation: Otsu's thresholding is commonly employed in image segmentation tasks. By applying the Otsu method, an optimal threshold is determined to separate objects of interest (foreground) from the background in an image. This enables further analysis and processing of specific regions within the image.

Object Detection: In computer vision and pattern recognition, the Otsu method can be utilized for object detection. By thresholding an image using Otsu's method, the objects of interest can be isolated from the background, facilitating subsequent steps in the object detection pipeline.

Image Binarization: Otsu's thresholding is frequently used for image binarization, where a grayscale image is converted into a binary image containing only black and white pixels. This binarization process is useful for various image analysis tasks, such as feature extraction, shape analysis, and character recognition.

Noise Removal: Otsu's thresholding can also be employed as a noise removal technique. By converting an image into a binary representation using an optimal threshold determined by the Otsu method, noise pixels with intensities similar to the background can be effectively eliminated.

Overall, the Otsu method is a versatile technique that finds applications in image processing, computer vision, and various fields where thresholding is necessary. It provides an automated and data-driven approach for determining an optimal threshold, making it valuable for tasks such as image segmentation, object detection, binarization, and noise removal.

### 2.2.3 Particle Swarm Optimization

The social behaviour of fish schools and flocks of birds serves as the basis for the population-based optimisation technique known as particle swarm optimisation (PSO). Since its debut in 1995 by Kennedy and Eberhart, it has grown to be a popular metaheuristic method for resolving optimisation issues.

A population of particles in PSO stands in for possible answers to the optimisation problem. With respect to both its own best-known position (personal best) and the best-known position of every other particle in the population (global best), each particle moves through the search space and modifies its position. The method explores the search space and converges on the best answer by updating the locations and velocities of the particles repeatedly.

The main components of the PSO algorithm are as follows:

Initialization: The procedure starts by assigning a population of particles within the search space random places and velocities. Each particle additionally records both its own best position and the best position ever found by all other particles combined.

Particle Movement: Based on their present positions, velocities, and the effect of their individual and collective best positions, the particles update their velocities and positions at each iteration. The current velocity, inertia, cognitive component (personal best influence), and social component (global best influence) all work together to decide a particle's new velocity.

Evaluation of Fitness: Following the update of locations, an objective function is used to assess each particle's fitness. The degree to which a particle's position resembles a preferable outcome is determined by the objective function. It specifies the optimisation criteria and is problem-specific.

Updating Personal and Global Bests: If a particle's current fitness is higher than its prior best, its personal best position is upgraded. Similar to this, if any particle finds a position that is superior than the existing global best, the global best position is updated.

Termination Criteria: The algorithm iterates until a termination condition is satisfied, such as when the number of iterations reaches a certain threshold or the level of fitness is reached.

PSO excels in efficiently navigating through complex search spaces and converging on optimal or nearly optimal solutions. It is applicable in many fields, including engineering, data mining, machine learning, and robotics since it can handle continuous and discrete optimisation problems.

PSO does, however, have some restrictions. As it can prematurely converge to a local optima, it may have trouble with multimodal issues where there are numerous optimal solutions. This restriction can be addressed using strategies like adding diversity maintenance techniques or hybridising PSO with other algorithms.

In general, Particle Swarm Optimisation is a population-based optimisation technique that resolves optimisation issues by simulating the social behaviour of particles. It provides an effective and quick method for locating the best solutions across a variety of application domains.

## 2.3 Feature Extraction

Feature extraction is a fundamental step in data analysis and pattern recognition, particularly in the field of machine learning and computer vision. It involves transforming raw data, such as images, audio signals, or text, into a reduced set of meaningful and informative features. These features capture the essential characteristics or properties of the data that are relevant for subsequent analysis or classification tasks.

The process of feature extraction involves the following steps:

Data Representation: Raw data is typically represented in its original format, such as pixels in an image or time-domain samples in an audio signal. Each data point contains a multitude of attributes or variables.

Feature Selection or Extraction: Feature selection involves choosing a subset of the original attributes that are most relevant to the problem at hand. Feature extraction, on the other hand, involves transforming the original attributes into a new set of features that better represent the underlying patterns or structure in the data. Feature extraction methods aim to capture the most discriminative and informative aspects of the data while minimizing redundancy.

Dimensionality reduction: Feature extraction frequently includes lowering the data's dimensionality. The processing of high-dimensional data can be computationally expensive and may experience the dimensionality curse. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are examples of dimensionality reduction techniques that are frequently used to project the data onto a lower-dimensional space while retaining as much useful information as feasible.

Feature Normalization: It is often necessary to normalize the extracted features to ensure that they are on a consistent scale. Normalization techniques such as mean normalization or min-max scaling are applied to make the features comparable and reduce the influence of different scales on subsequent analysis or modelling steps.

The most pertinent information is retained while the complexity of the data is reduced, which makes feature extraction crucial. The ensuing analysis or classification activities become more accurate and efficient by extracting meaningful information. These properties can be used as inputs to machine learning algorithms, such as neural networks, support vector machines, or decision trees, allowing them to recognise objects, understand speech, analyse sentiment, or discover anomalies, among other things.

The nature of the data, the particular problem area, and the available domain knowledge all influence the technique selection for feature extraction. Different feature extraction methods adapted to the needs of the work and the data's properties are needed for various fields. The quality and discriminative power of extracted features are continually being improved through the development and refinement of a variety of algorithms and techniques.

## 2.4 Classification

Classifying data into predetermined classes or categories is a basic process in machine learning and pattern recognition. Based on the patterns and relationships discovered from labelled training data, it is used to forecast the class labels of previously unknown or upcoming instances.

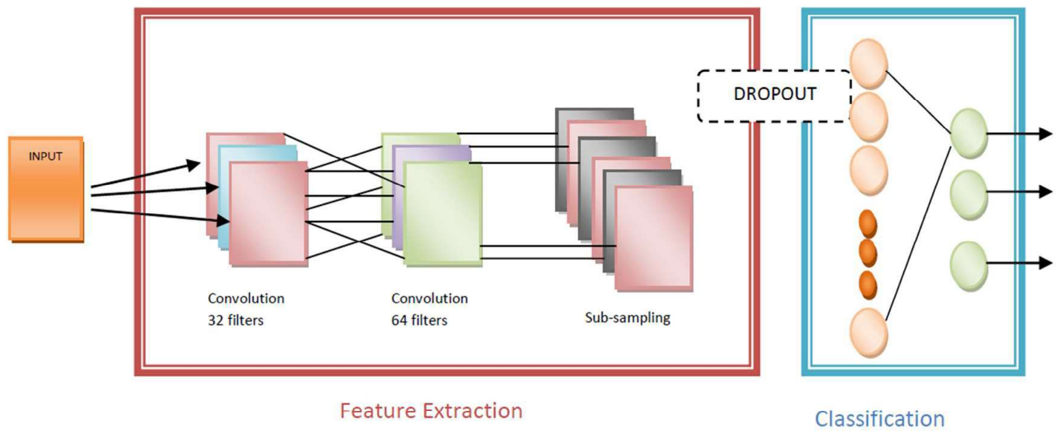


Fig 2. Example of Image Classification

The two main categories of classification algorithms are deep learning algorithms and conventional machine learning algorithms. Decision trees, random forests, logistic regression, support vector machines, and k-nearest neighbours are examples of conventional methods. Artificial neural networks (ANNs) and convolutional neural networks (CNNs), two types of deep learning algorithms, have grown in prominence because of their capacity to recognise intricate patterns in data and build hierarchical representations.

Numerous domains, including image recognition, natural language processing, sentiment analysis, fraud detection, medical diagnosis, and spam filtering, to mention a few, have extensive uses for classification. Based on past or present data, it enables automated decision-making, pattern detection, and predictive modelling.

The kind of data, the number of classes, the availability of labelled training data, and the intended trade-off between accuracy and computational complexity all play a role in the selection of a classification algorithm. Each method has advantages and disadvantages, thus the choice should be made depending on the requirements and characteristics of the current problem.

## 2.5 Support Vector Machine

The widely used supervised machine learning technique Support Vector Machine (SVM) can be utilised for both classification and regression problems. It was created in the 1990s by Vapnik and Cortes and has grown in prominence as a result of its success in managing complicated datasets.

Finding the best hyperplane to divide data points into distinct classes in a high-dimensional feature space is the goal of SVM. The kernel function used by SVM to translate the input data into a higher-dimensional space selects the hyperplane that maximises the margin between the classes.

Key components of SVM include:

Hyperplane: In SVM, a hyperplane represents the decision boundary that separates data points of different classes. The hyperplane can be a line in 2D space, a plane in 3D space, or a hyperplane in higher dimensions.

Support Vectors: Support vectors are the data points closest to the hyperplane and have the most influence on its position. They play a crucial role in defining the decision boundary and determining the margin.

Margin: The margin is the region between the support vectors of different classes. SVM aims to maximize the margin as a wider margin generally leads to better generalization and improved robustness of the classifier.

Kernel Function: SVM employs a kernel function to implicitly map the input data into a higher-dimensional feature space, where it becomes easier to find a hyperplane that separates the data. Common kernel functions include linear, polynomial, Gaussian (RBF), and sigmoid.

## 2.6 Random Forest

Popular machine learning algorithm Random Forest is a part of the supervised learning methodology. It can be applied to ML issues involving both classification and regression. It is built on the idea of ensemble learning, which is a method of integrating various classifiers to address difficult issues and enhance model performance.

According to what its name implies, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead than depending on a single decision tree, the random forest uses forecasts from each tree and predicts the result based on the votes of the majority of predictions.

Higher accuracy and overfitting are prevented by the larger number of trees in the forest.

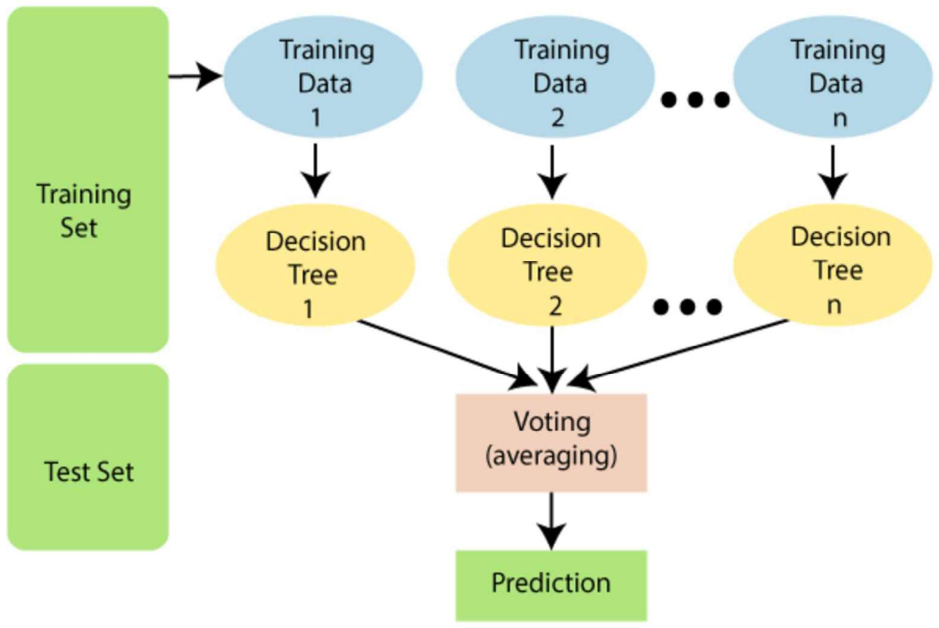


Fig 3. Random forest algorithm working

Figure 3 shows the working of random forest algorithm.

### Assumptions for random forest classifier:

Some decision trees may predict the correct output, while others may not, because the random forest combines numerous trees to forecast the class of the dataset. But when all the trees are combined, they forecast the right result. Consequently, the following two presumptions for an improved Random forest classifier:

- For the dataset's feature variable to predict true outcomes rather than a speculated result, there should be some actual values in the dataset.
- Each tree's predictions must have extremely low correlations.

## 2.7 KNN Classifier

K-Nearest Neighbors (KNN) is a widely used classification algorithm in Machine Learning. It is considered essential and finds applications in various fields such as pattern recognition, data mining, and intrusion detection.

KNN is a supervised learning algorithm that operates on prior training data to classify new data points based on their attributes. It is known for its non-parametric nature, which means it does not assume any specific data distribution like other algorithms such as Gaussian Mixture Models (GMM).

Instead of relying on assumptions, KNN determines the class of a new data point by considering its proximity to the nearest neighbors. The majority vote of the K nearest neighbors determines the classification. This flexibility makes KNN suitable for real-life scenarios with diverse data types and without the need for complex assumptions.

Take the following table of data points with two features as an illustration:

Assign these points to a group by examining the training set now that you have another set of data points (also known as testing data). The unclassified locations are designated as "White," as you'll see.

## Intuition behind this algorithm:

We might be able to find some clusters or groupings if we plot these points on a graph. Now that a point has been declassified, we may classify it by looking at which group its closest neighbours are a part of. This indicates that a point's likelihood of being labelled as "Red" increases the closer it is to a group of points with the same colour designation.

Intuitively, it is clear that the first point (2.5, 7) belongs in the "Green" category while the second point (5.5, 4.5) belongs in the "Red" category.

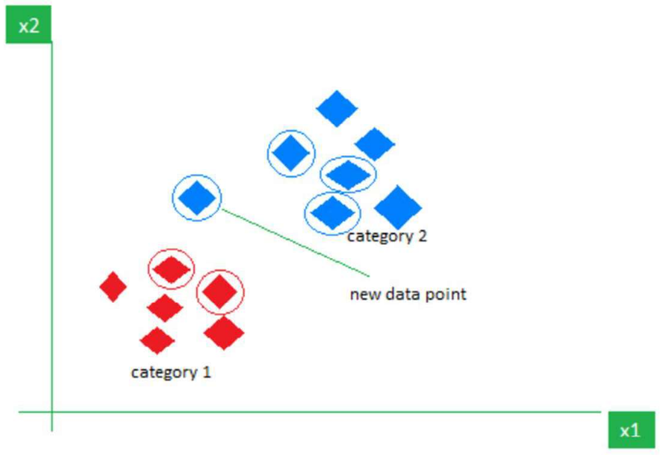


Fig 4. KNN algorithm working visualizaton

## 2.8 Convolutional Neural Network

Deep learning models such as convolutional neural networks (CNNs) have significantly improved computer vision tasks. For tasks like image identification, object detection, and picture segmentation, these networks are now frequently used. Using the spatial correlations between the input pieces, CNNs are specifically made to handle data with a grid-like layout, such as photographs.

The convolution operation is the foundation of CNN. Traditional neural networks connect every neuron in a layer to every neuron in the layer before it, which leads to a lot of parameters and a loss of spatial information. Convolutional layers, which apply filters or kernels to specific small regions of the input, are used by CNNs to get around this issue. The network can then pick up on regional trends and spatial hierarchies as a result.

The following are the main components and characteristics of CNNs:

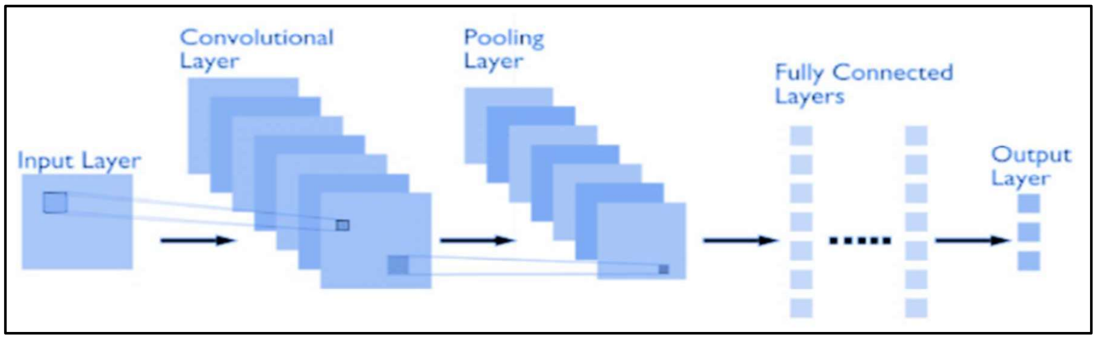


Fig 5. Convolutional Neural Network Architecture

In fig 3, the convolutional neural network architecture is shown, including input layer, convolutional layer, pooling layer, fully connected layers, and output. The more information is given below for reference.

Convolutional Layers: These layers comprise multiple filters or kernels that convolve with the input data. By performing element-wise multiplications and summations, the filters capture various features present in the input, such as edges, textures, or shapes. The resulting feature maps retain spatial information and represent higher-level features as the network deepens.

Pooling Layers: Pooling layers perform downsampling of the feature maps, reducing their spatial dimensions. Common pooling techniques include max pooling, average pooling, and sum pooling. Pooling helps extract the most salient features, reduce computational complexity, and introduce some degree of translation invariance.

There are 2 common functions used in pooling:

Max Pooling: Max-pooling is a technique used in convolutional neural networks (CNNs) to down-sample feature maps and reduce the complexity of the network. It involves dividing the feature map into non-overlapping sectors and selecting the maximum value within each sector to construct a down-sampled (pooled) feature map.

Unlike convolutional layers that extract features by convolving filters over the input, max-pooling focuses on selecting the most prominent features within local regions. After a convolutional layer, max-pooling is commonly applied to capture the most salient information while reducing the spatial dimensions of the feature map.

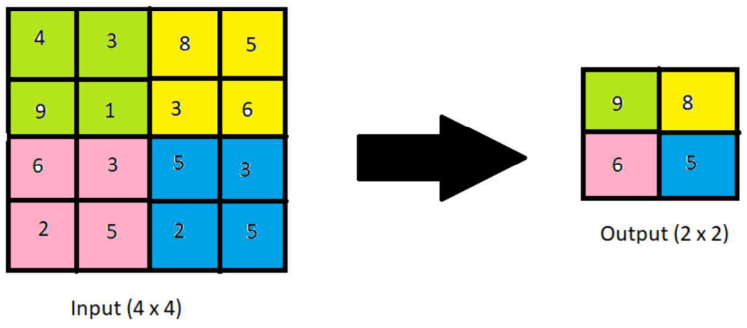


Fig 6. 2*2 Max Pooling on Input(4*4)

In fig 4, we have a matrix of input of size 4*4, which is further reduced to 2*2 matrix output by applying Max pooling operation.

Average Pooling: In average pooling, the feature map is divided into non-overlapping sectors, similar to max-pooling. However, instead of selecting the maximum value within each sector, average pooling calculates the average value. This is done by summing the values within each sector and then dividing by the number of elements in the sector.

The purpose of average pooling is to create a down sampled representation of the feature map while preserving the overall trends and average values of the features. It helps to capture the average activation levels and smooth out the spatial variations in the feature map. Average pooling can be particularly useful when the precise localization of features is less important, and the overall presence or absence of certain features is more relevant.

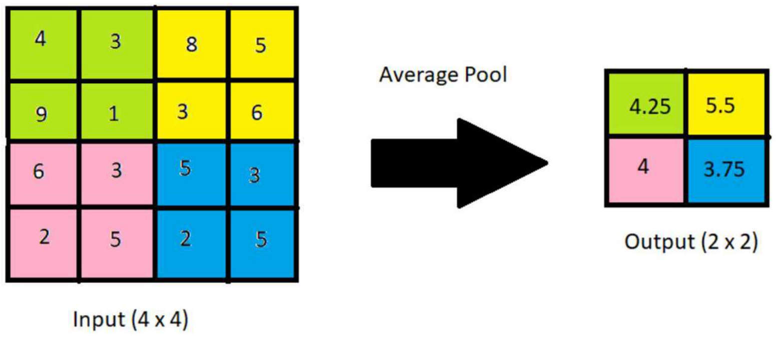


Fig 7. 2*2 Average Pooling on Input(4*4)

In fig 5, we have a matrix of input of size 4*4, which is further reduced to 2*2 matrix output by applying Average pooling operation.

**Activation Functions:** Following the convolutional and pooling layers, non-linear activation functions, such as Rectified Linear Unit (ReLU), are frequently used. By introducing non-linearity to the network, these functions make it possible to understand intricate correlations between features.

**Linear activation function:**

The linear function's equation is y= m*x.

Equation : f(x )= x …(1)

Range: -∞ to ∞

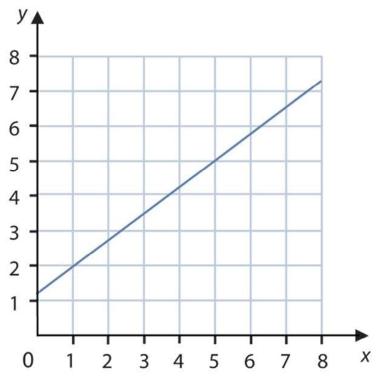


Fig 8. Linear function graph

**Sigmoid Function:** The sigmoid or sigmoid activation function's curve resembles an 'S' shaped curve. Between zero and one is the range of the logistic activation function. Because value of the sigmoid function is limited between zero and one, the outcome is likely to be one if the value is greater than 0.5 & zero else.

Equation: f(x) =$\frac{1}{1+e\text{^}x}$ …(2)

Range: [0, 1]

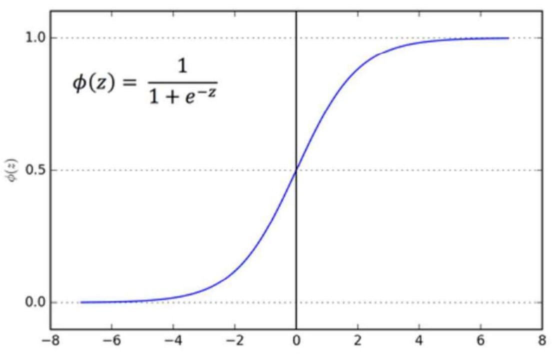


Fig 9. Sigmoid Activation function graph

**Tanh activation function:** Tanh is a hyperbolic tangent function, like the logistic sigmoid. The curves of the Tanh and sigmoid activation functions are quite similar, however Tanh is preferable since the whole function is zero centric.

Equation: f(x) = tanh(x) = $\frac{2}{1+e\text{^}(2x)}$ $-1$ …(3)

Range: [-1, 1]

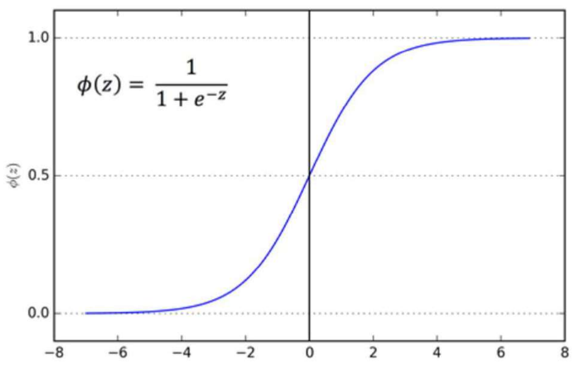


Fig 10. Tanh Activation function graph

**Rectified Linear Function:** ReLU is the abbreviation for the activation function. This linear function will provide a positive output if the input is positive; else, it will produce a straight zero. ReLU activation function is used as the default activation function in the majority of neural network models due to its effectiveness and simplicity in training.

The function formulated as:

f(x)=max(0,x) …(4)

Here, the function is returning 0 if it is given negative input, but for any positive value x the function returns x.

Graphically, the function is looks like:

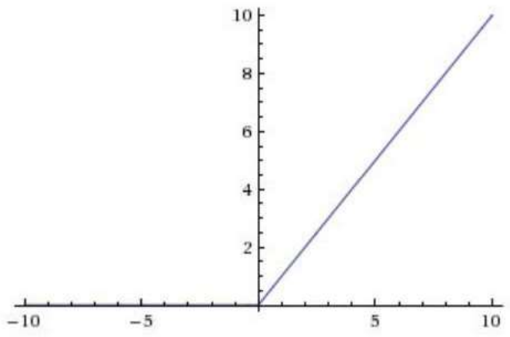


Fig 11. Relu Activation function graph

Fully Connected Layers: Fully connected layers are frequently included in CNNs after a number of convolutional and pooling layers. The network can learn high-level representations and generate predictions thanks to the connections established by these layers between every neuron in one layer and every neuron in the following layer. The CNN's output layer for classification tasks correlates to the number of classes.

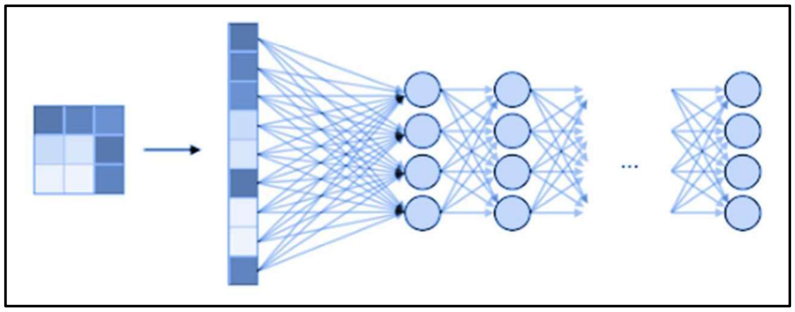


Fig 12. Fully Connected Layer

Backpropagation is a technique used to train CNNs utilising huge datasets that have been labelled. The error between the expected and actual labels is used by the network to modify its internal parameters, including weights and biases. Stochastic gradient descent (SGD) or its derivatives are examples of optimisation techniques that are often employed. Additionally, methods like dropout and batch normalisation are used to strengthen training stability and lessen overfitting. Techniques like batch normalization and dropout are also employed to improve training stability and reduce overfitting.

CNNs offer several advantages that make them particularly suitable for image-related tasks:

Hierarchical Feature Learning: CNNs learn hierarchical representations of features, progressing from low-level to high-level abstractions. This capability allows them to capture intricate patterns and complex relationships in the data.

Parameter Sharing: The use of shared parameters in convolutional layers significantly reduces the number of parameters. This makes CNNs computationally efficient and reduces the risk of overfitting, particularly when working with limited datasets.

Translation Invariance: Due to their local connectivity and pooling layers, CNNs can identify features regardless of their precise position in the input. This property provides a degree of translation invariance.

Generalization: CNNs have demonstrated strong generalization capabilities, performing well on unseen data and effectively generalizing patterns learned from the training data to make accurate predictions.

Pretrained Models and Transfer Learning: CNNs pretrained on large-scale datasets, such as ImageNet, can serve as a valuable starting point for transfer learning. Pretrained models can be fine-tuned on smaller, domain-specific datasets, enabling efficient use of limited labeled data.

CNNs have significantly advanced various computer vision applications, including image classification, object detection, semantic segmentation, facial recognition, and medical image analysis. They continue to drive innovation and push.

## 2.9 Survival Prediction Model

Cox regression analysis will be used to include the segmented tumor volume and patient clinical data into a survival prediction model. A popular technique for predicting survival is Cox regression analysis, which models the hazard rate as a function of the predictor factors. The R survival package will be used to run the Cox regression analysis.

In terms of overall survival prediction and progression-free survival prediction, the performance of the survival prediction model will be assessed. The performance of the model will be evaluated using the concordance index and time-dependent ROC analysis, and it will be validated using 10-fold cross-validation.

# CHAPTER 3

# PROPOSED WORK

## 3.1 Approach

The suggested method will partition brain tumors using a 3D U-Net CNN architecture. An encoder network and a decoder network joined by a bottleneck layer make up the U-Net design. While the decoder network reconstructs the segmented image from the encoded information, the encoder network is in charge of extracting high-level features from the input image. To retain spatial information, the U-Net architecture additionally uses skip links between the encoder and decoder networks.

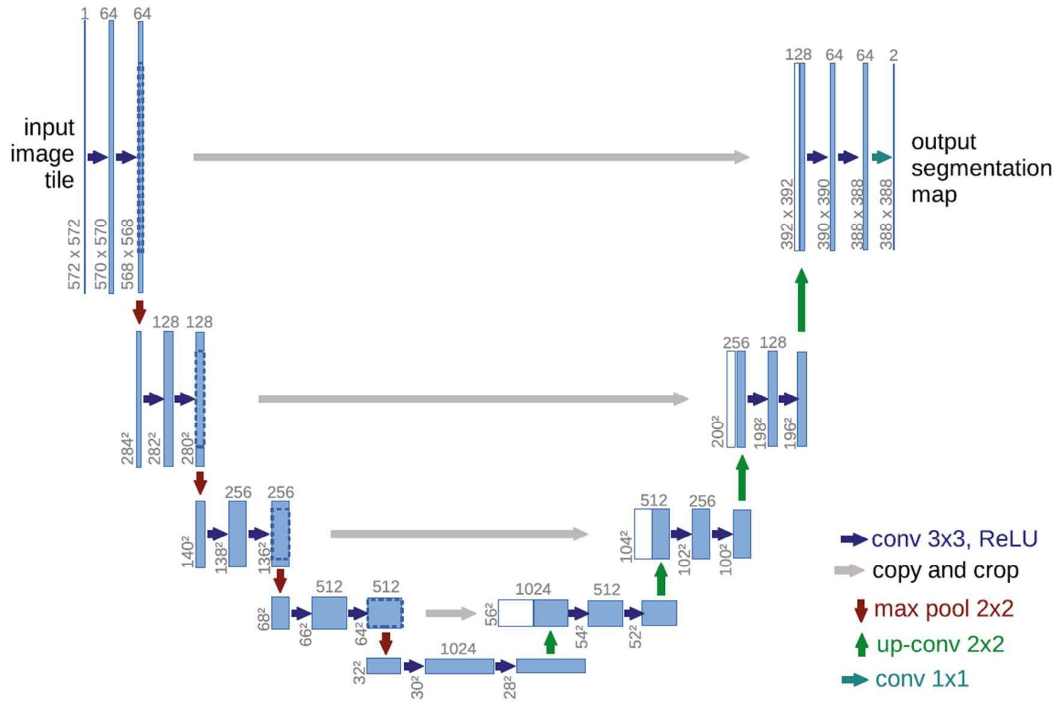


Fig 13. U-net architecture (a 32x32 pixel sample). Here, each blue box corresponds to a map of multichannel characteristics. The number of channels is indicated on the box's top. The box's lower left edge provides the x and y sizes. Copies of feature maps

are shown in white boxes. The various operations are indicated by the arrows and are described on the right bottom.

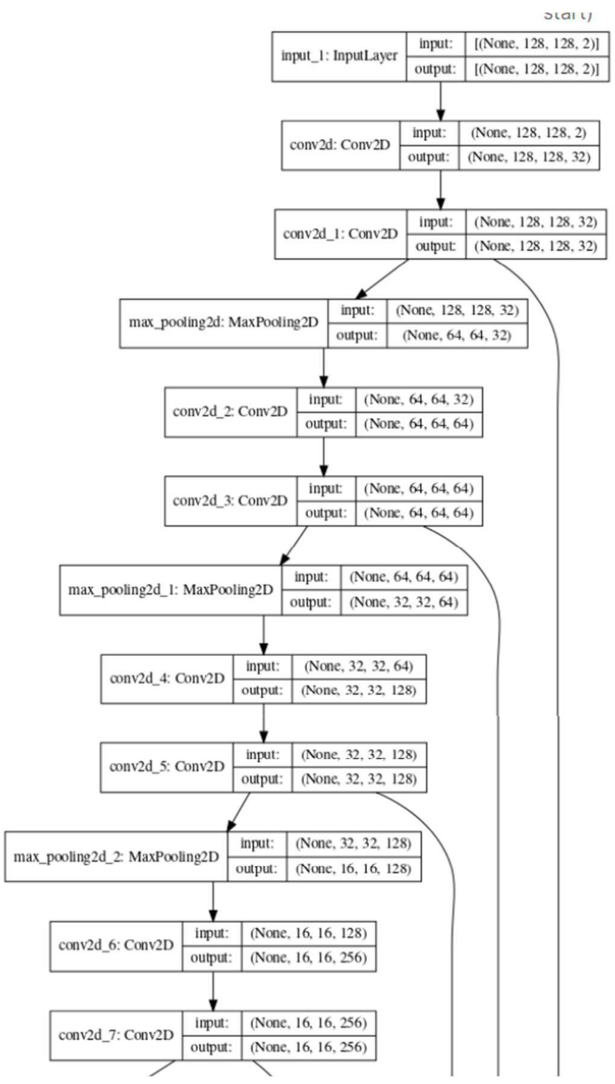
## 3.2 Model Architecture

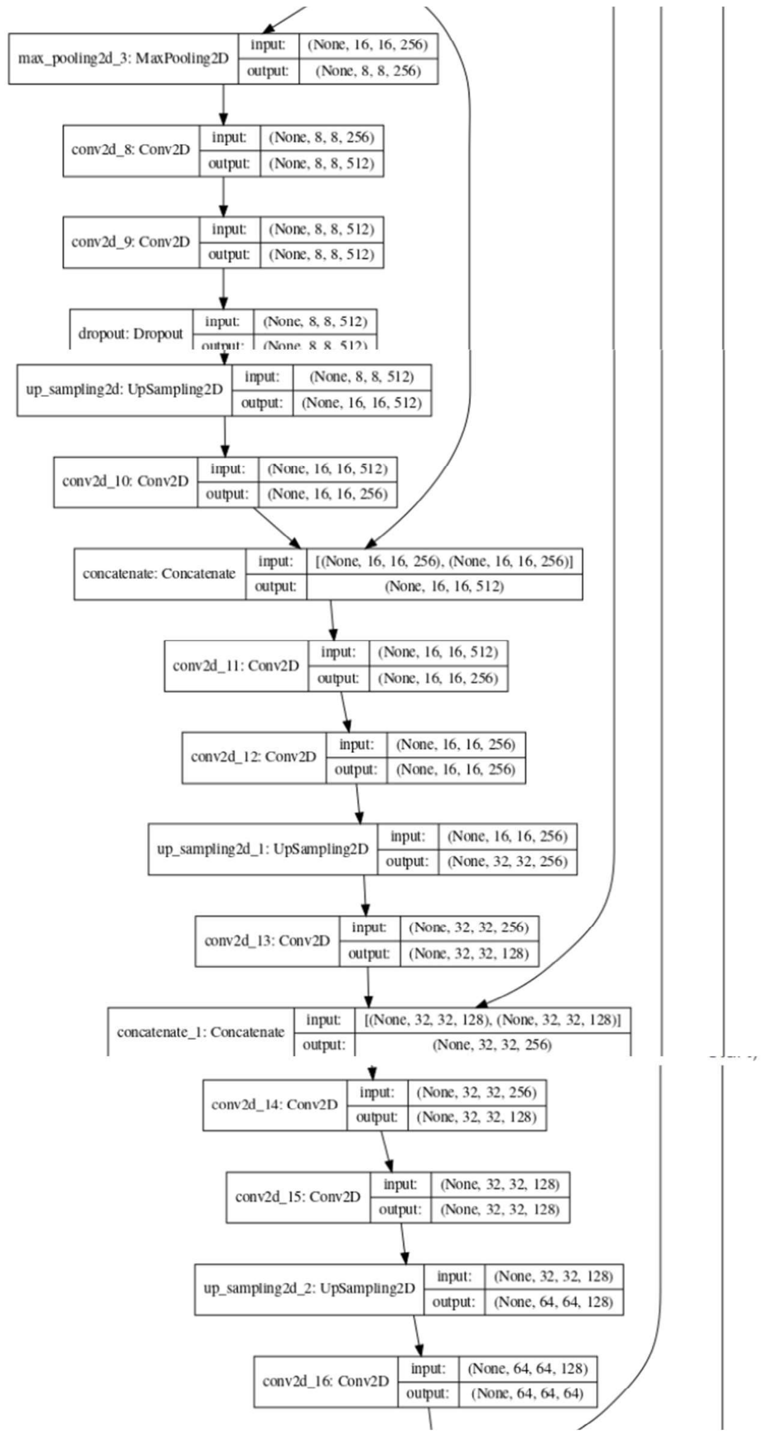
The contracting path (left side) and the expansive path (right side) are the two primary parts of the network architecture shown in Figure 1. The convolutional network architecture used in the contracting path is standard. It consists of a series of two 3x3 convolutions that are immediately followed by rectified linear unit (ReLU) activation functions. For down sampling, a 2x2 max pooling process with a 2 stride is also used. There are two times as many feature channels at each downsampling stage. The contracting path is used to reduce the spatial dimensions while extracting high-level data.

The expansive path seeks to reconstitute the segmentation map as a complement to the contracting path. The feature map is upsampled in each step of the expanding route, and then a 2x2 convolution (also known as "up-convolution") that cuts the number of feature channels in half is applied. The suitably cropped feature map from the contracting path is then concatenated with the upsampled feature map. To incorporate both low-level and high-level features from various scales, concatenation is required. The feature representation is then enhanced using two 3x3 convolutions, each followed by a ReLU activation. The loss of border pixels that occurs during each convolution operation is made up for by the cropping process. In order to transfer the 64-component feature vector to the appropriate number of classes, a 1x1 convolution is used as the final layer. There are a total of 23 convolutional layers in the network.

It is essential to select an appropriate input tile size to guarantee continuous tiling of the output segmentation map. In order to apply all 2x2 max pooling operations to a layer with an even x- and y-size, the input tile size should be chosen in this manner. This factor guarantees consistency in both the upsampling and downsampling processes, producing a consistent segmentation map over the whole image.

By combining the contracting and expansive paths, the network architecture enables effective feature extraction and accurate reconstruction of the segmentation map. The repeated application of convolutions, ReLU activations, and pooling operations aids in capturing hierarchical features at different scales. The architecture's ability to seamlessly tile the output segmentation map ensures consistency and facilitates the integration of high-level and low-level features, ultimately contributing to improved segmentation performance.


input_1: InputLayer
input: [(None, 128, 128, 2)]
output: [(None, 128, 128, 2)]
conv2d: Conv2D
input: (None, 128, 128, 2)
output: (None, 128, 128, 32)
conv2d_1: Conv2D
input: (None, 128, 128, 32)
output: (None, 128, 128, 32)
max_pooling2d: MaxPooling2D
input: (None, 128, 128, 32)
output: (None, 64, 64, 32)
conv2d_2: Conv2D
input: (None, 64, 64, 32)
output: (None, 64, 64, 64)
conv2d_3: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 64, 64, 64)
output: (None, 32, 32, 64)
conv2d_4: Conv2D
input: (None, 32, 32, 64)
output: (None, 32, 32, 128)
conv2d_5: Conv2D
input: (None, 32, 32, 128)
output: (None, 32, 32, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 32, 32, 128)
output: (None, 16, 16, 128)
conv2d_6: Conv2D
input: (None, 16, 16, 128)
output: (None, 16, 16, 256)
conv2d_7: Conv2D
input: (None, 16, 16, 256)
output: (None, 16, 16, 256)


max_pooling2d_3: MaxPooling2D | input: (None, 16, 16, 256) | output: (None, 8, 8, 256)
conv2d_8: Conv2D | input: (None, 8, 8, 256) | output: (None, 8, 8, 512)
conv2d_9: Conv2D | input: (None, 8, 8, 512) | output: (None, 8, 8, 512)
dropout: Dropout | input: (None, 8, 8, 512) | output: (None, 8, 8, 512)
up_sampling2d: UpSampling2D | input: (None, 8, 8, 512) | output: (None, 16, 16, 512)
conv2d_10: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 256)
concatenate: Concatenate | input: [(None, 16, 16, 256), (None, 16, 16, 256)] | output: (None, 16, 16, 512)
conv2d_11: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 256)
conv2d_12: Conv2D | input: (None, 16, 16, 256) | output: (None, 16, 16, 256)
up_sampling2d_1: UpSampling2D | input: (None, 16, 16, 256) | output: (None, 32, 32, 256)
conv2d_13: Conv2D | input: (None, 32, 32, 256) | output: (None, 32, 32, 128)
concatenate_1: Concatenate | input: [(None, 32, 32, 128), (None, 32, 32, 128)] | output: (None, 32, 32, 256)
conv2d_14: Conv2D | input: (None, 32, 32, 256) | output: (None, 32, 32, 128)
conv2d_15: Conv2D | input: (None, 32, 32, 128) | output: (None, 32, 32, 128)
up_sampling2d_2: UpSampling2D | input: (None, 32, 32, 128) | output: (None, 64, 64, 128)
conv2d_16: Conv2D | input: (None, 64, 64, 128) | output: (None, 64, 64, 64)

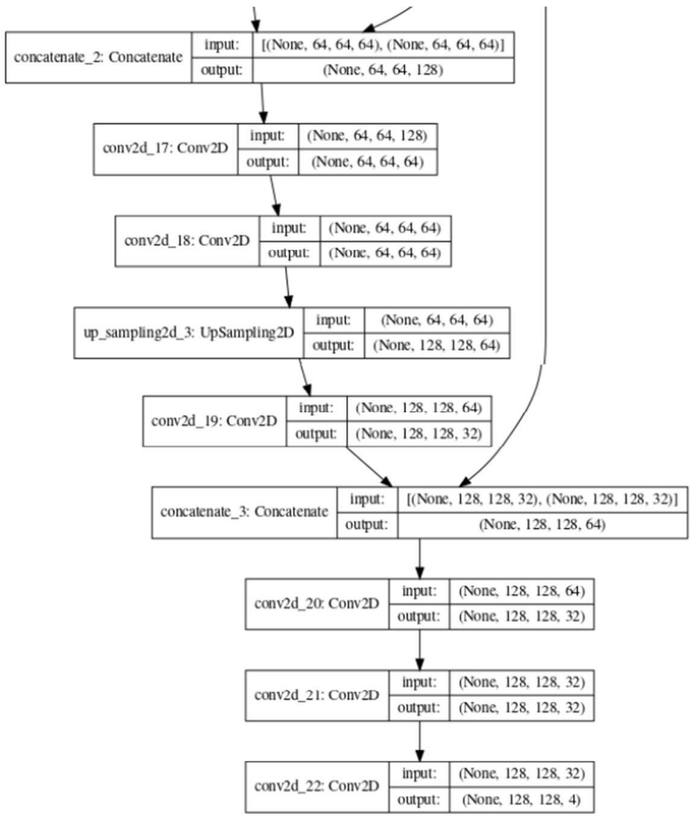
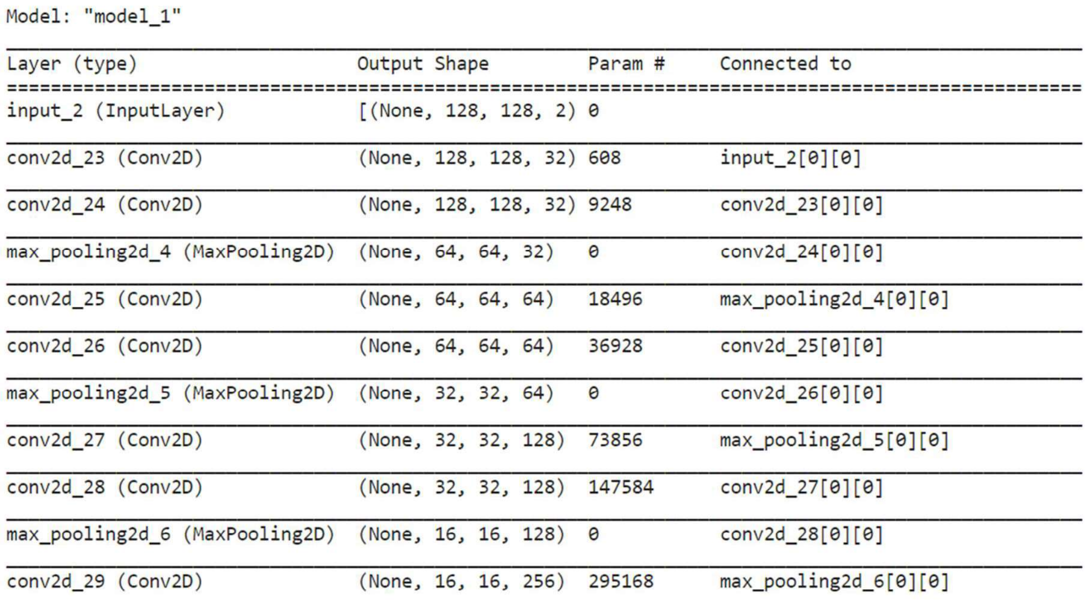

concatenate_2: Concatenate
input: [(None, 64, 64, 64), (None, 64, 64, 64)]
output: (None, 64, 64, 128)
conv2d_17: Conv2D
input: (None, 64, 64, 128)
output: (None, 64, 64, 64)
conv2d_18: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 64)
up_sampling2d_3: UpSampling2D
input: (None, 64, 64, 64)
output: (None, 128, 128, 64)
conv2d_19: Conv2D
input: (None, 128, 128, 64)
output: (None, 128, 128, 32)
concatenate_3: Concatenate
input: [(None, 128, 128, 32), (None, 128, 128, 32)]
output: (None, 128, 128, 64)
conv2d_20: Conv2D
input: (None, 128, 128, 64)
output: (None, 128, 128, 32)
conv2d_21: Conv2D
input: (None, 128, 128, 32)
output: (None, 128, 128, 32)
conv2d_22: Conv2D
input: (None, 128, 128, 32)
output: (None, 128, 128, 4)

Fig 14. Architecture of model showing different layers and operations.



```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 128, 128, 2) 0
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 128, 128, 32) 608         input_2[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 128, 128, 32) 9248        conv2d_23[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 64, 64, 32)   0           conv2d_24[0][0]
__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, 64, 64, 64)   18496       max_pooling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 64, 64, 64)   36928       conv2d_25[0][0]
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 32, 32, 64)   0           conv2d_26[0][0]
__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, 32, 32, 128)  73856       max_pooling2d_5[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 32, 32, 128)  147584      conv2d_27[0][0]
__________________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)  (None, 16, 16, 128)  0           conv2d_28[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 16, 16, 256)  295168      max_pooling2d_6[0][0]
```

```
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 16, 16, 256)  590080      conv2d_29[0][0]
__________________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)  (None, 8, 8, 256)    0           conv2d_30[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 8, 8, 512)    1180160     max_pooling2d_7[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 8, 8, 512)    2359808     conv2d_31[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 8, 8, 512)    0           conv2d_32[0][0]
__________________________________________________________________________________________________
up_sampling2d_4 (UpSampling2D)  (None, 16, 16, 512)  0           dropout_1[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 16, 16, 256)  524544      up_sampling2d_4[0][0]


__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 16, 16, 512)  0           conv2d_30[0][0]
                                                                 conv2d_33[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 16, 16, 256)  1179904     concatenate_4[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 16, 16, 256)  590080      conv2d_34[0][0]
__________________________________________________________________________________________________
up_sampling2d_5 (UpSampling2D)  (None, 32, 32, 256)  0           conv2d_35[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 32, 32, 128)  131200      up_sampling2d_5[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 32, 32, 256)  0           conv2d_28[0][0]
                                                                 conv2d_36[0][0]
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 32, 32, 128)  295040      concatenate_5[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 32, 32, 128)  147584      conv2d_37[0][0]
__________________________________________________________________________________________________
up_sampling2d_6 (UpSampling2D)  (None, 64, 64, 128)  0           conv2d_38[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 64, 64, 64)   32832       up_sampling2d_6[0][0]
__________________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 64, 64, 128)  0           conv2d_26[0][0]
                                                                 conv2d_39[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 64, 64, 64)   73792       concatenate_6[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 64, 64, 64)   36928       conv2d_40[0][0]
__________________________________________________________________________________________________
up_sampling2d_7 (UpSampling2D)  (None, 128, 128, 64) 0           conv2d_41[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, 128, 128, 32) 8224        up_sampling2d_7[0][0]
__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 128, 128, 64) 0           conv2d_24[0][0]
                                                                 conv2d_42[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, 128, 128, 32) 18464       concatenate_7[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, 128, 128, 32) 9248        conv2d_43[0][0]


__________________________________________________________________________________________________
conv2d_45 (Conv2D)              (None, 128, 128, 4)  132         conv2d_44[0][0]
==================================================================================================
Total params: 7,759,908
Trainable params: 7,759,908
Non-trainable params: 0
__________________________________________________________________________________________________
```

Table 1. Model summary showing different layers and operations.

## 3.3 Loss function

The Dice coefficient, sometimes called the Srensen-Dice coefficient or the F1 score, is a statistic used to compare two binary masks and is frequently used in image segmentation tasks. It gauges how closely the expected segmentation mask and the actual segmentation mask match up or agree. A perfect match between the two masks is indicated by a Dice coefficient of 1, which has a range of 0 to 1.

The Dice coefficient is calculated using the formula (2 * |A B|) / (|A| + |B|), where A stands for the anticipated mask and B for the actual mask. The denominator is the total number of pixels in both masks, while the numerator computes the intersection of the predicted and ground truth masks.

The Dice coefficient can be used as a measure of dissimilarity in the context of a loss function. The model seeks to maximise the overlap and resemblance between the predicted and ground truth masks by minimising the Dice loss, which is determined as 1 minus the Dice coefficient. This promotes the model to generate segmentations that are increasingly accurate and exact. In medical picture segmentation tasks, where imbalanced datasets are frequent and there are many more background pixels than target pixels, the Dice coefficient is especially helpful. By taking true positives, false positives, and false negatives into account equally, the Dice coefficient effectively addresses this class imbalance.

# CHAPTER 4

# EXPERIMENTS AND RESULTS

## 4.1 Input Data descriptions

As illustrated in figure 13 below, all BraTS multimodal scans are available as NIfTI files (.nii.gz), a popular medical imaging format for storing brain imaging data generated using MRI and describing various MRI settings.

T1: sagittal or axial 2D acquisitions using T1-weighted native images with slice thickness ranging from 1-6 mm.

T1c: For most patients, this is a T1-weighted, contrast-enhanced (Gadolinium) image with 3D acquisition.

T2: axial 2D capture of a T2-weighted image with a slice thickness of 2–6 mm.

FLAIR: Axial, coronal, or sagittal 2D acquisitions with a slice thickness of 2 to 6 mm are referred to as FLAIR.

Data were collected from numerous (n=19) institutes using diverse scanners and clinical regimens.

Following the same annotation technique, one to four raters manually segmented each imaging dataset, and their annotations were approved by skilled neuroradiologists. According to the BraTS 2012–2013 TMI study and the most recent BraTS summarising paper, annotations include the GD–enhancing tumor (ET — label 4), the peritumoral edoema (ED — label 2), and the necrotic and non–enhancing tumor core (NCR/NET — label 1). After pre-processing, which includes co-registering to the same anatomical template, interpolating to the same resolution (1 mm3), and stripping the skull, the given data are distributed.

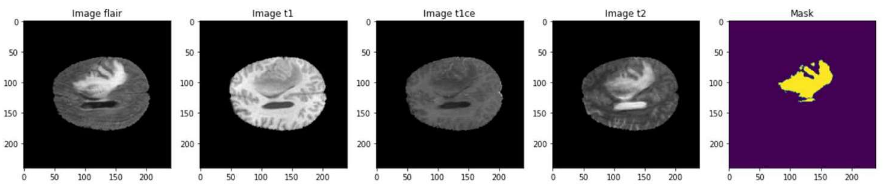


Fig 15. MRI scans of brain with different settings T1, T1c, T2, and Flair defined above

In figure 14, we can see the 3d data of each slice of Nifti data, plotted the data, later in figure 15, the segment is shown in each slice of data. We can also see the segment data with different settings.

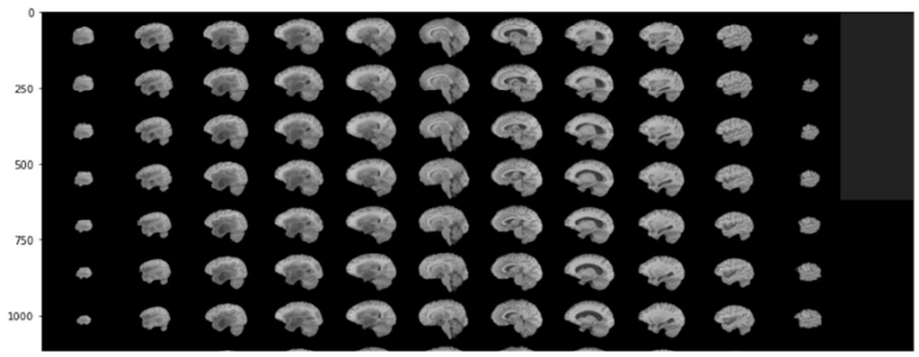


Fig 16. 3d data from each slice of Nifti

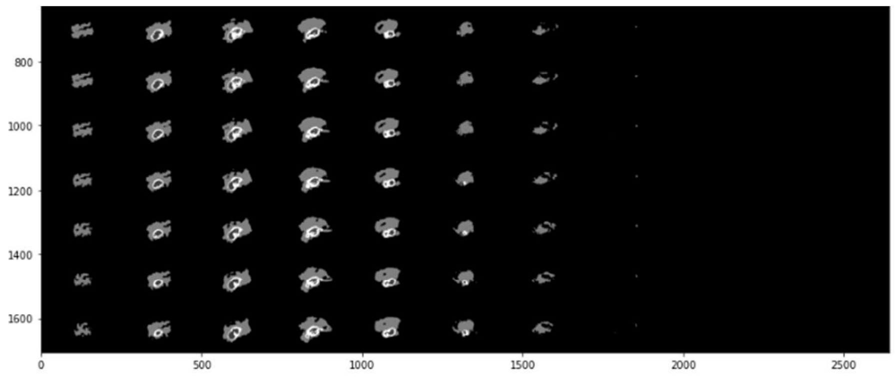
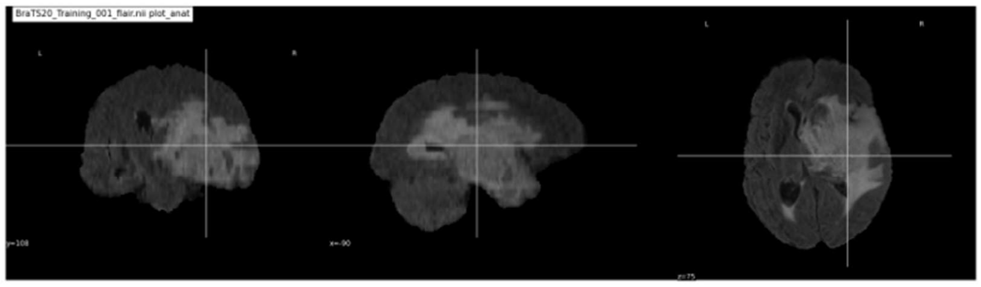
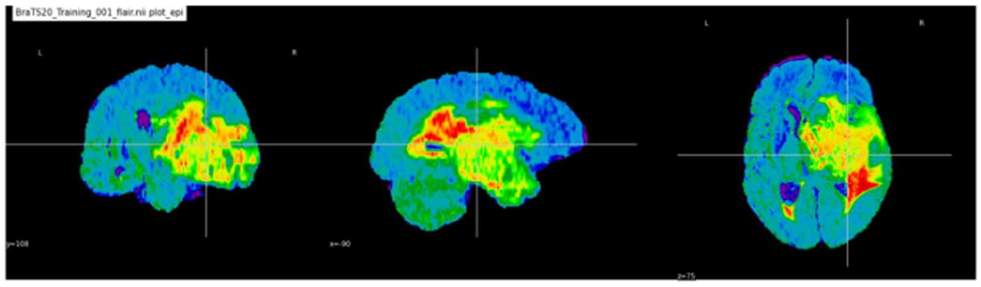


Fig 17. Segmentation shown in each slice data





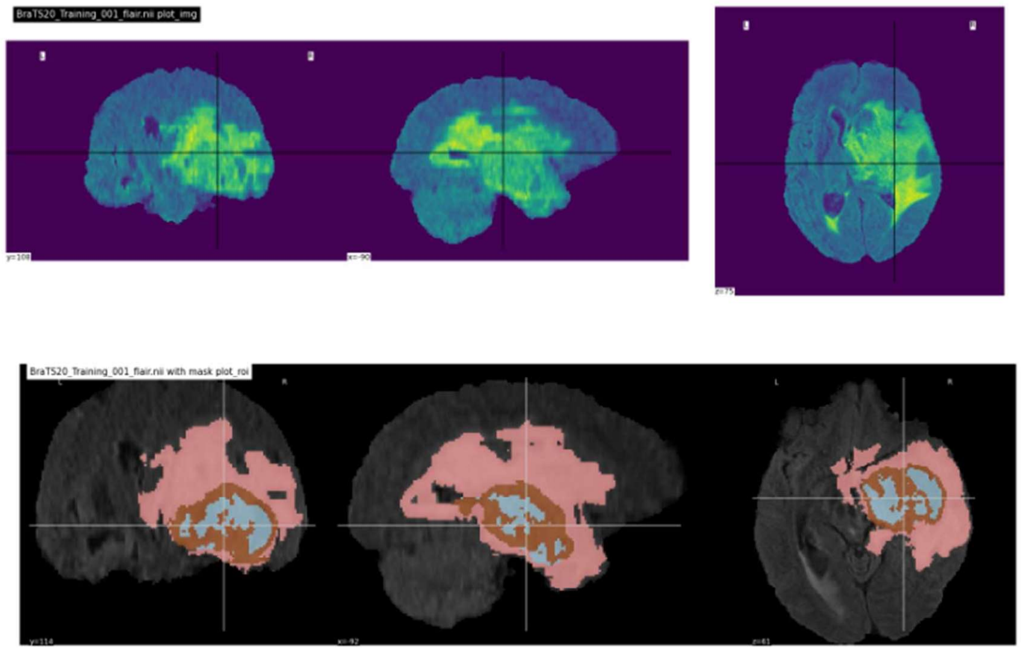

BraTS20_Training_001_flair.nii plot_img
BraTS20_Training_001_flair.nii with mask plot_roi

Fig 18. Segmented data shown with different effects on MRI scans

## 4.2 Setting up the Environment

Importing the various libraries which are going to use in the model building and throughout the project.

- import os, cv2, glob, PIL, shutil, numpy as np, pandas as pd, seaborn as sns, matplotlib.pyplot as plt
- from skimage from data
- from skimage.util import montage
- import skimage.transform as skTrans
- from skimage.transform import rotate
- from skimage.transform import resize

for neural imaging:

- import nilearn as nl, nibabel as nib, nilearn.plotting as nlplt
-

for machine learning libs:

- import keras, keras.backend as K, tensorflow as tf
- from keras.callbacks import CSVLogger
- from tensorflow.keras.utils import plot_model

- from sklearn.preprocessing import MinMaxScaler
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import classification_report
- from tensorflow import *
- from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping, TensorBoard
- from tensorflow.keras.layers.experimental import preprocessing

## 4.3 Data Pre-processing

Since the data are too large to fit in memory, loading everything into memory is not a smart idea. In order to divide the data into training, validation, and testing sets, we will develop data generators.

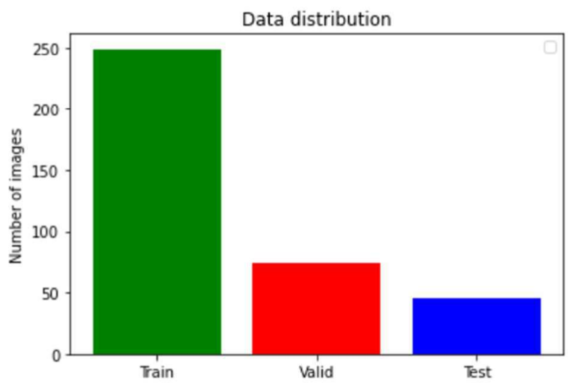


Fig 19. Graph showing data size taken for Train, Test and Validation

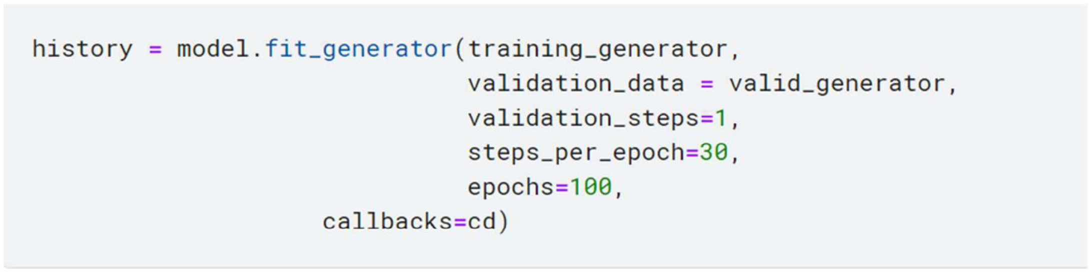
## 4.4 Model Training

1. Training and validation data: The training_generator and valid_generator are data generators that provide the training and validation data, respectively. These generators typically load the data in batches, allowing for efficient memory utilization when working with large datasets.
2. Validation configuration: The validation_data parameter is set to the valid_generator to specify the validation data for monitoring the model's performance during training. The validation_steps parameter is set to 1, indicating that only one batch of validation data will be used for evaluation.
3. Training configuration: The steps_per_epoch parameter is set to 30, indicating the number of batches to be processed in each epoch. This value determines

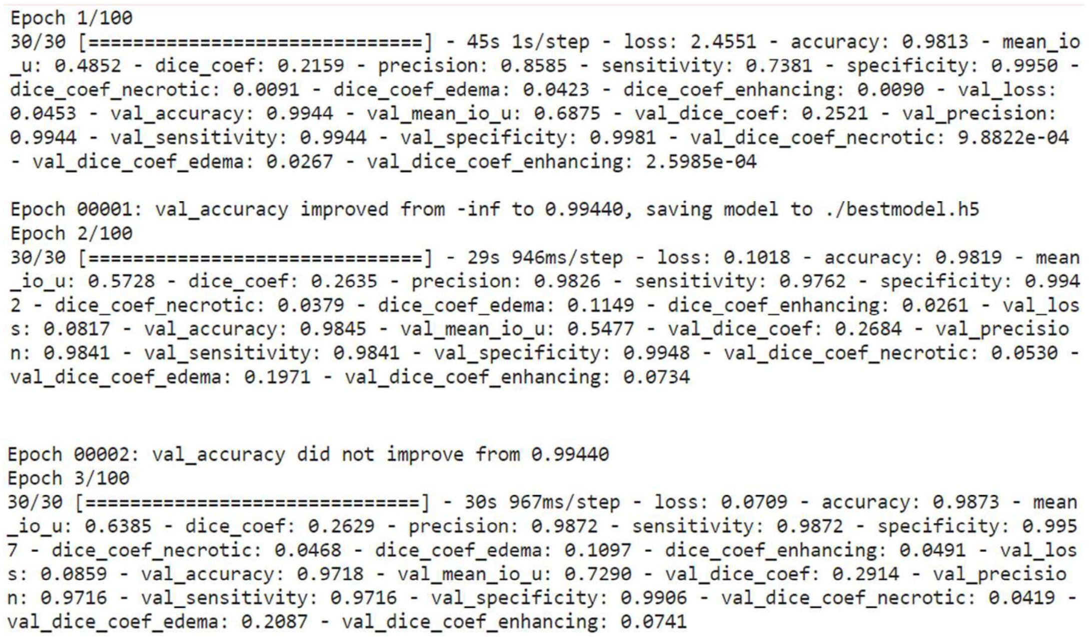
the number of training steps per epoch. The epochs parameter is set to 100, specifying the total number of training epochs.

4. Callbacks: The callbacks parameter is set to cd, which presumably represents a custom-defined callback or a list of callbacks. Callbacks in Keras provide functionality for performing specific actions during training, such as saving model checkpoints, adjusting learning rate, or logging training metrics.
5. Model training: The model.fit_generator function is called to start the training process. It fits the model to the training data generated by training_generator and evaluates the model on the validation data from valid_generator. The training proceeds for the specified number of epochs, and the callbacks are triggered as specified.

By executing this code, the model will be trained for the given number of epochs using the provided generators, and the training progress will be monitored through the specified callbacks.



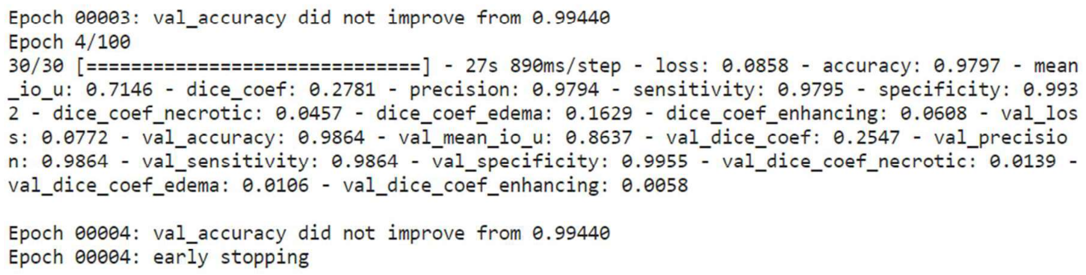
```
history = model.fit_generator(training_generator,
                              validation_data = valid_generator,
                              validation_steps=1,
                              steps_per_epoch=30,
                              epochs=100,
                    callbacks=cd)
```



```
Epoch 1/100
30/30 [==============================] - 45s 1s/step - loss: 2.4551 - accuracy: 0.9813 - mean_io
_u: 0.4852 - dice_coef: 0.2159 - precision: 0.8585 - sensitivity: 0.7381 - specificity: 0.9950 -
dice_coef_necrotic: 0.0091 - dice_coef_edema: 0.0423 - dice_coef_enhancing: 0.0090 - val_loss:
0.0453 - val_accuracy: 0.9944 - val_mean_io_u: 0.6875 - val_dice_coef: 0.2521 - val_precision:
0.9944 - val_sensitivity: 0.9944 - val_specificity: 0.9981 - val_dice_coef_necrotic: 9.8822e-04
- val_dice_coef_edema: 0.0267 - val_dice_coef_enhancing: 2.5985e-04

Epoch 00001: val_accuracy improved from -inf to 0.99440, saving model to ./bestmodel.h5
Epoch 2/100
30/30 [==============================] - 29s 946ms/step - loss: 0.1018 - accuracy: 0.9819 - mean
_io_u: 0.5728 - dice_coef: 0.2635 - precision: 0.9826 - sensitivity: 0.9762 - specificity: 0.994
2 - dice_coef_necrotic: 0.0379 - dice_coef_edema: 0.1149 - dice_coef_enhancing: 0.0261 - val_los
s: 0.0817 - val_accuracy: 0.9845 - val_mean_io_u: 0.5477 - val_dice_coef: 0.2684 - val_precisio
n: 0.9841 - val_sensitivity: 0.9841 - val_specificity: 0.9948 - val_dice_coef_necrotic: 0.0530 -
val_dice_coef_edema: 0.1971 - val_dice_coef_enhancing: 0.0734


Epoch 00002: val_accuracy did not improve from 0.99440
Epoch 3/100
30/30 [==============================] - 30s 967ms/step - loss: 0.0709 - accuracy: 0.9873 - mean
_io_u: 0.6385 - dice_coef: 0.2629 - precision: 0.9872 - sensitivity: 0.9872 - specificity: 0.995
7 - dice_coef_necrotic: 0.0468 - dice_coef_edema: 0.1097 - dice_coef_enhancing: 0.0491 - val_los
s: 0.0859 - val_accuracy: 0.9718 - val_mean_io_u: 0.7290 - val_dice_coef: 0.2914 - val_precisio
n: 0.9716 - val_sensitivity: 0.9716 - val_specificity: 0.9906 - val_dice_coef_necrotic: 0.0419 -
val_dice_coef_edema: 0.2087 - val_dice_coef_enhancing: 0.0741
```



```
Epoch 00003: val_accuracy did not improve from 0.99440
Epoch 4/100
30/30 [==============================] - 27s 890ms/step - loss: 0.0858 - accuracy: 0.9797 - mean
_io_u: 0.7146 - dice_coef: 0.2781 - precision: 0.9794 - sensitivity: 0.9795 - specificity: 0.993
2 - dice_coef_necrotic: 0.0457 - dice_coef_edema: 0.1629 - dice_coef_enhancing: 0.0608 - val_los
s: 0.0772 - val_accuracy: 0.9864 - val_mean_io_u: 0.8637 - val_dice_coef: 0.2547 - val_precisio
n: 0.9864 - val_sensitivity: 0.9864 - val_specificity: 0.9955 - val_dice_coef_necrotic: 0.0139 -
val_dice_coef_edema: 0.0106 - val_dice_coef_enhancing: 0.0058

Epoch 00004: val_accuracy did not improve from 0.99440
Epoch 00004: early stopping
```

Fig 20. Model Training using fit_generator

We must load the model, load the model's training history, and plot the metrics in order to see the training and validation performance of a loaded model. This will enable us to comprehend how the model learned throughout the training process better.

- Loading the model: The keras.models.load_model function is used to load a pre-trained model from the specified path. Custom objects, including metrics such as accuracy, dice coefficient, precision, sensitivity, specificity, and custom loss functions, are provided to ensure compatibility during model loading.
- Loading the training history: The training history is loaded from a CSV file using the pd.read_csv function. The file contains information about various metrics such as accuracy, loss, dice coefficient, and mean IOU for both training and validation sets.
- Plotting the metrics: When training, the code generates a figure with four subplots to display various metrics. Training accuracy and validation accuracy, training loss and validation loss, training dice coefficient and validation dice coefficient, as well as training mean IOU and validation mean IOU, are among the parameters plotted. The plots assist in evaluating the model's performance and spotting any over- or underfitting problems.
- Displaying the plots: The plt.show() function is called to display the generated plots.

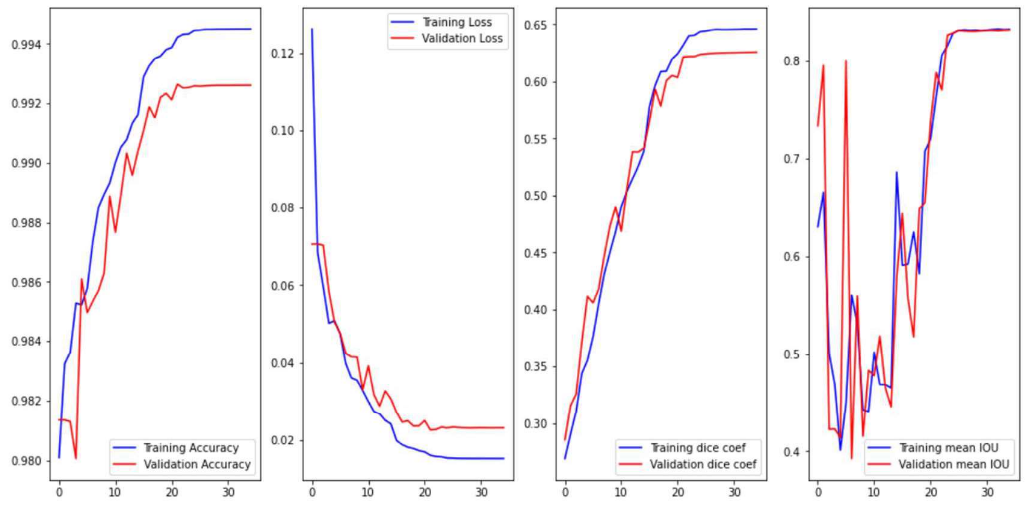


Fig 21. Training and performance of the model

## 4.5 Segment Classes

Key: Integer values representing different tumor segmentation classes.

Value: Corresponding class labels describing the type of tumor segment.

The classes are:

| | |
|---|---|
| 0 | "Not Tumor" |
| 1 | "Necrotic/Core" |
| 2 | "EDEMA" |
| 3 | "Enhancing" |

Table 2. Segment Classes of MRI images

'NOT tumor': This class represents regions in the MRI scan that do not contain any tumor. It includes normal brain tissue or areas unaffected by any tumor growth.

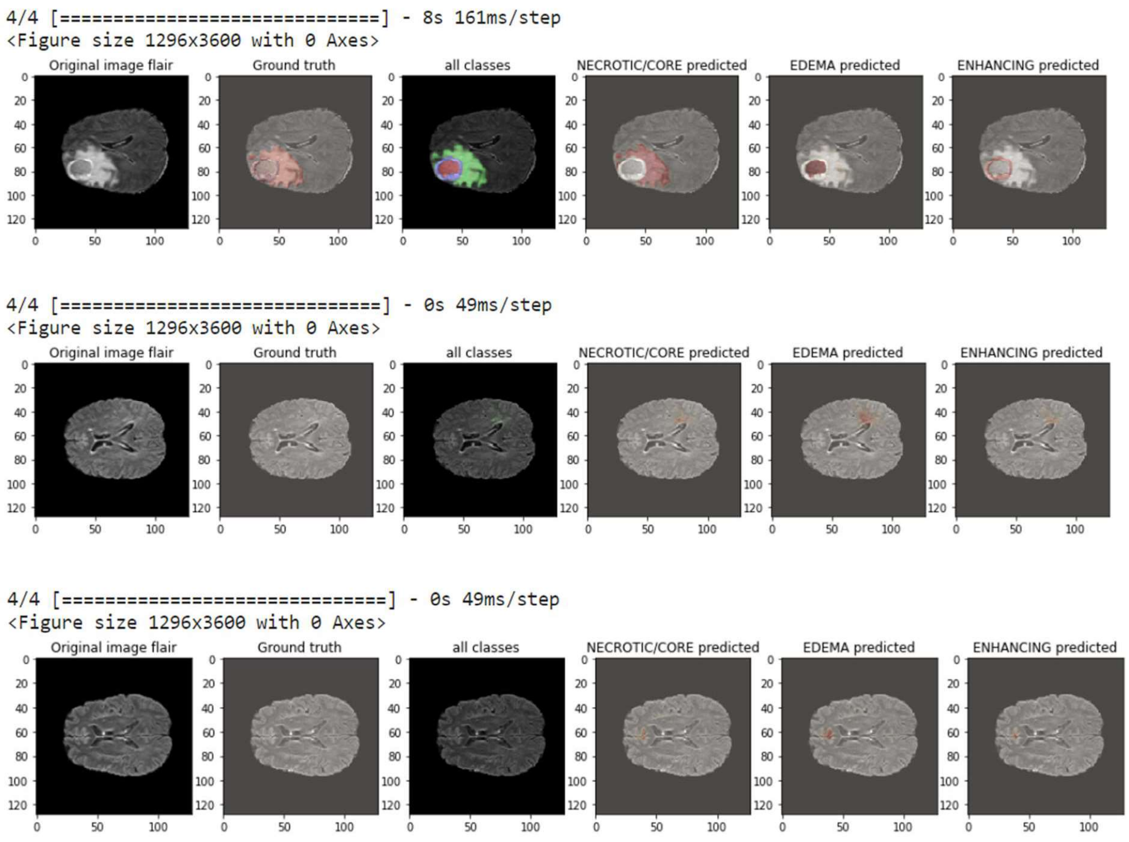
'NECROTIC/CORE': The necrotic (dead) core of the tumor corresponds to this class. Necrosis is caused by a tumor's limited blood supply, which results in cell death. The area in the middle of the tumor is referred to as its core.

'EDEMA': 'EDEMA' class represents the peritumoral edema, which is the swelling or fluid accumulation around the tumor. Edema is caused by the disruption of the blood-brain barrier and the release of fluid into the surrounding brain tissue.

'ENHANCING': This class represents the enhancing region of the tumor. Enhancement indicates active tumor growth with increased blood flow and cell proliferation. It typically corresponds to the most aggressive and malignant part of the tumor.

These classifications are used to identify and separate various regions of interest within an MRI scan in the context of brain tumor segmentation. Each class represents a specific characteristic or component of the tumor, allowing for a more detailed analysis and understanding of the tumor's extent and behavior.

These are some examples of MRI images given for segmentions



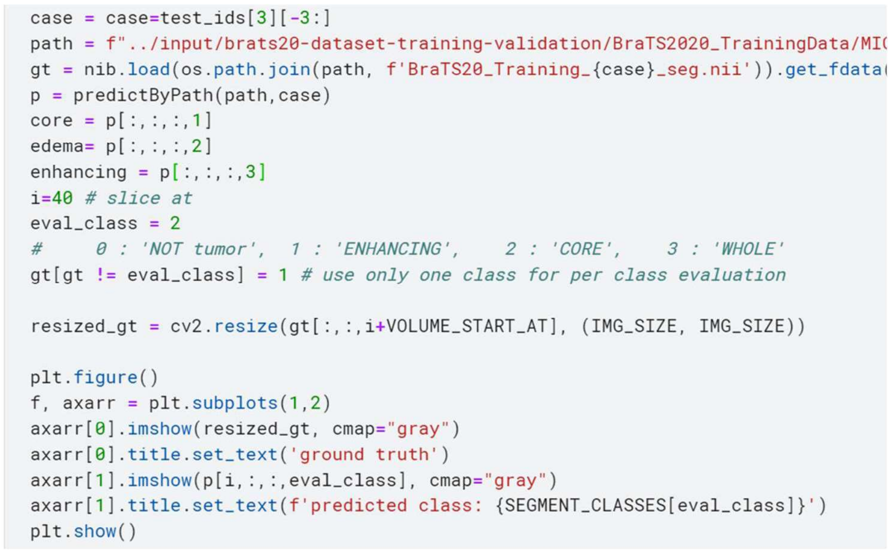
```
4/4 [==============================] - 8s 161ms/step
<Figure size 1296x3600 with 0 Axes>
```

```
4/4 [==============================] - 0s 49ms/step
<Figure size 1296x3600 with 0 Axes>
```

```
4/4 [==============================] - 0s 49ms/step
<Figure size 1296x3600 with 0 Axes>
```

Fig 22. Predicted example of segmented of MRI image of Brain

## 4.6 Evaluation



```
case = case=test_ids[3][-3:]
path = f"../input/brats20-dataset-training-validation/BraTS2020_TrainingData/MIC
gt = nib.load(os.path.join(path, f'BraTS20_Training_{case}_seg.nii')).get_fdata(
p = predictByPath(path,case)
core = p[:,:,:,1]
edema= p[:,:,:,2]
enhancing = p[:,:,:,3]
i=40 # slice at
eval_class = 2
#     0 : 'NOT tumor',   1 : 'ENHANCING',    2 : 'CORE',    3 : 'WHOLE'
gt[gt != eval_class] = 1 # use only one class for per class evaluation

resized_gt = cv2.resize(gt[:,:,i+VOLUME_START_AT], (IMG_SIZE, IMG_SIZE))

plt.figure()
f, axarr = plt.subplots(1,2)
axarr[0].imshow(resized_gt, cmap="gray")
axarr[0].title.set_text('ground truth')
axarr[1].imshow(p[i,:,:,eval_class], cmap="gray")
axarr[1].title.set_text(f'predicted class: {SEGMENT_CLASSES[eval_class]}')
plt.show()
```

```
4/4 [==============================] - 0s 69ms/step
<Figure size 432x288 with 0 Axes>
```

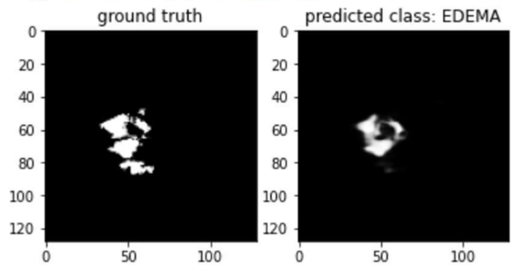
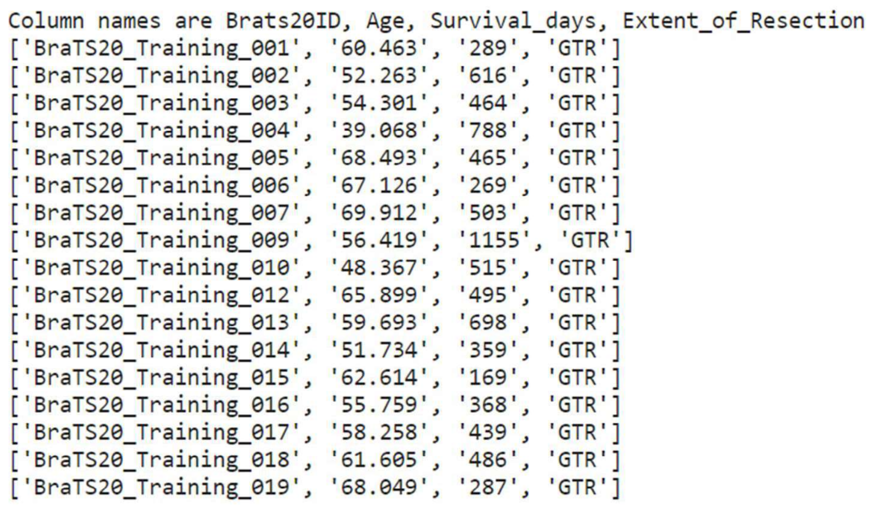


Fig 23. Evaluation of the MRI data, showing EDEMA on providing testing

## 4.7 Survival Prediction

For survival prediction, first see what the age distribution in the dataset and their survival days is. We have a CSV file containing survival information for brain tumor patients.

The data includes information about patients' IDs (Brats20ID), age, survival days, and extent of resection. Each row represents a patient's data, and the values can be accessed by their corresponding column names.



```
Column names are Brats20ID, Age, Survival_days, Extent_of_Resection
['BraTS20_Training_001', '60.463', '289', 'GTR']
['BraTS20_Training_002', '52.263', '616', 'GTR']
['BraTS20_Training_003', '54.301', '464', 'GTR']
['BraTS20_Training_004', '39.068', '788', 'GTR']
['BraTS20_Training_005', '68.493', '465', 'GTR']
['BraTS20_Training_006', '67.126', '269', 'GTR']
['BraTS20_Training_007', '69.912', '503', 'GTR']
['BraTS20_Training_009', '56.419', '1155', 'GTR']
['BraTS20_Training_010', '48.367', '515', 'GTR']
['BraTS20_Training_012', '65.899', '495', 'GTR']
['BraTS20_Training_013', '59.693', '698', 'GTR']
['BraTS20_Training_014', '51.734', '359', 'GTR']
['BraTS20_Training_015', '62.614', '169', 'GTR']
['BraTS20_Training_016', '55.759', '368', 'GTR']
['BraTS20_Training_017', '58.258', '439', 'GTR']
['BraTS20_Training_018', '61.605', '486', 'GTR']
['BraTS20_Training_019', '68.049', '287', 'GTR']
```

Fig 24. Represent the data present in the CSV file

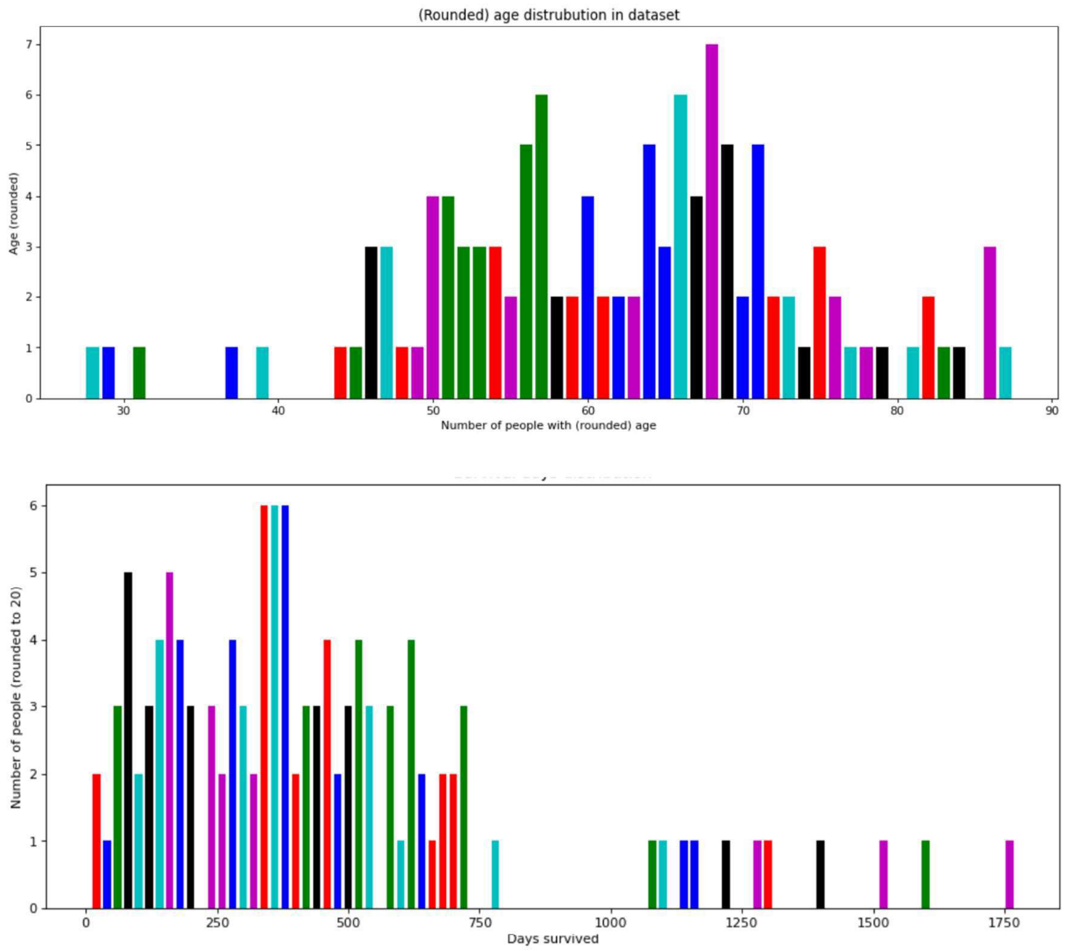


Fig 25. Graph plotting the number of people vs age and number of people vs days survived

## Survival Categories:

Key: String values representing different survival categories.

Value: Integer values representing the range of survival duration.

The categories are:

'SHORT': 0-300 (survival duration of 0 to 300 days)

'MEDIUM': 300-450 (survival duration of 300 to 450 days)

'LONG': 450 and more (survival duration of 450 days or more)

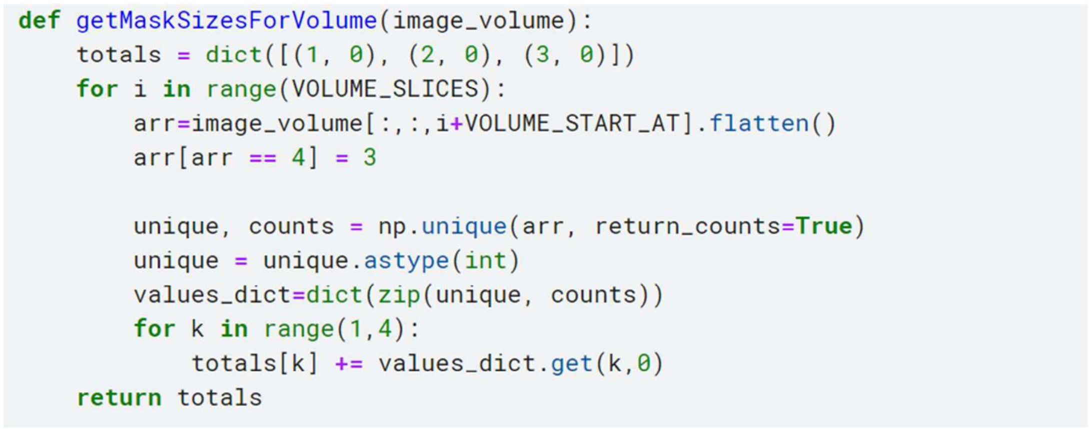
The survival categories mentioned above represent different ranges of survival duration for patients with brain tumors. Here is a detailed explanation of each survival category:

- Short Survival: This category includes patients who survived for a relatively short duration after diagnosis. In the given data, the cutoff for the short survival category is set at 300 days. Patients falling within this category have a survival duration ranging from the time of diagnosis up to 300 days.
- Medium Survival: This category comprises patients who survived for a moderate duration after diagnosis. In the provided data, the cutoff for the medium survival category is set between 300 and 450 days. Patients falling within this category have a survival duration ranging from 300 days to 450 days.
- Long Survival: This category consists of patients who survived for an extended duration after diagnosis. In the given data, the cutoff for the long survival category is set at 450 days. Patients falling within this category have a survival duration of 450 days or more.

These survival categories are used to classify patients based on their expected survival outcomes. They provide a simplified representation of the survival duration, allowing for easier analysis and comparison of patient groups. By categorizing patients into these groups, researchers and medical professionals can study the factors influencing survival rates and develop appropriate treatment strategies based on the expected survival category.

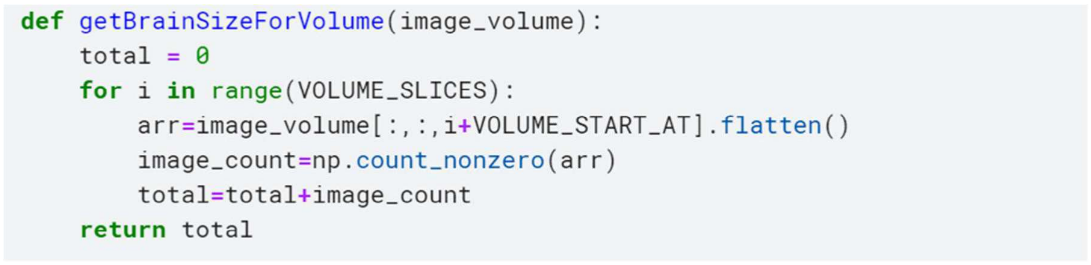
**Computing segment sizes:**

Find number of pixels for each class in volume, no need to compute as ration to image size, since all images are of same size 240x240

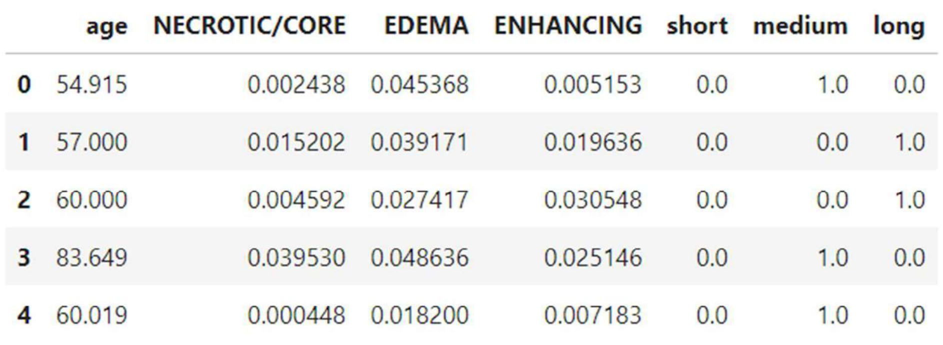


```python
def getMaskSizesForVolume(image_volume):
    totals = dict([(1, 0), (2, 0), (3, 0)])
    for i in range(VOLUME_SLICES):
        arr=image_volume[:,:,i+VOLUME_START_AT].flatten()
        arr[arr == 4] = 3

        unique, counts = np.unique(arr, return_counts=True)
        unique = unique.astype(int)
        values_dict=dict(zip(unique, counts))
        for k in range(1,4):
            totals[k] += values_dict.get(k,0)
    return totals
```



```python
def getBrainSizeForVolume(image_volume):
    total = 0
    for i in range(VOLUME_SLICES):
        arr=image_volume[:,:,i+VOLUME_START_AT].flatten()
        image_count=np.count_nonzero(arr)
        total=total+image_count
    return total
```



|  | age | NECROTIC/CORE | EDEMA | ENHANCING | short | medium | long |
|---|---|---|---|---|---|---|---|
| 0 | 54.915 | 0.002438 | 0.045368 | 0.005153 | 0.0 | 1.0 | 0.0 |
| 1 | 57.000 | 0.015202 | 0.039171 | 0.019636 | 0.0 | 0.0 | 1.0 |
| 2 | 60.000 | 0.004592 | 0.027417 | 0.030548 | 0.0 | 0.0 | 1.0 |
| 3 | 83.649 | 0.039530 | 0.048636 | 0.025146 | 0.0 | 1.0 | 0.0 |
| 4 | 60.019 | 0.000448 | 0.018200 | 0.007183 | 0.0 | 1.0 | 0.0 |

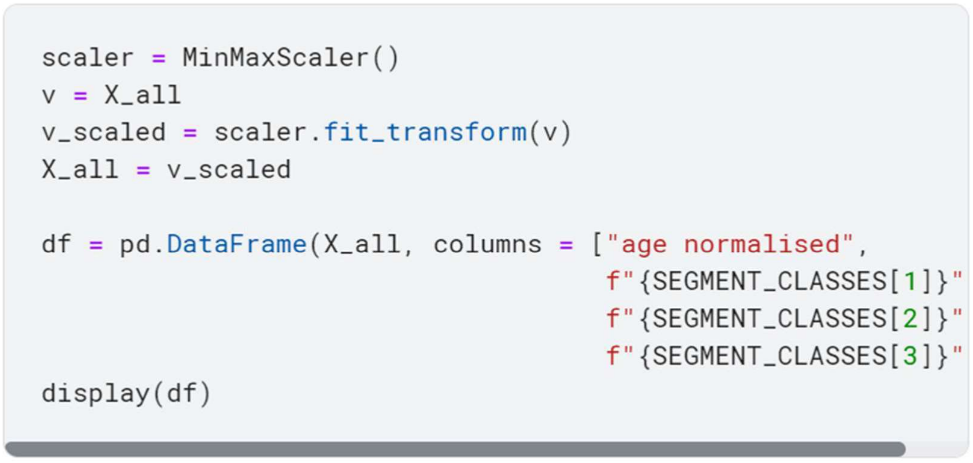
Table 3. Survival prediction based on short, medium, and long

The provided data consists of information related to age and tumor segmentations for a set of patients. Each row represents a patient, and the columns provide details about their age, the presence of tumor in different regions (NECROTIC/CORE, EDEMA, ENHANCING), and survival categories (short, medium, long).

The age column represents the age of the patients. The subsequent columns (NECROTIC/CORE, EDEMA, ENHANCING) indicate the probabilities of tumor presence in the corresponding regions, with values ranging from very low to relatively higher probabilities.

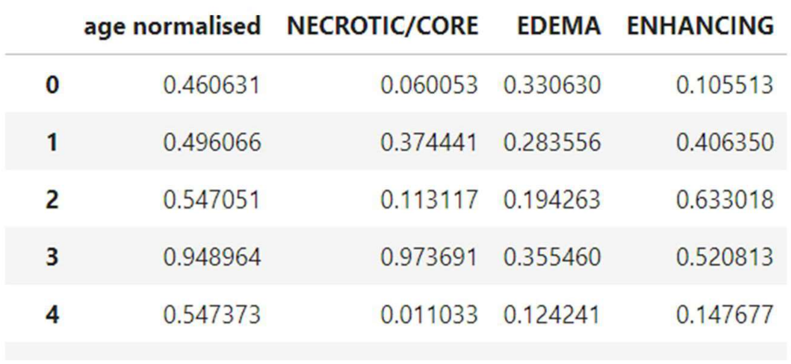
The last three columns (short, medium, long) represent survival categories. A value of 1.0 in one of these columns indicates that the patient falls into the respective survival category (short, medium, or long), while a value of 0.0 indicates the patient does not fall into that category.

Normalize the data performing min-max scaling into range [0, 1]



```
scaler = MinMaxScaler()
v = X_all
v_scaled = scaler.fit_transform(v)
X_all = v_scaled

df = pd.DataFrame(X_all, columns = ["age normalised",
                                    f"{SEGMENT_CLASSES[1]}"
                                    f"{SEGMENT_CLASSES[2]}"
                                    f"{SEGMENT_CLASSES[3]}"
display(df)
```



| | age normalised | NECROTIC/CORE | EDEMA | ENHANCING |
|---|---|---|---|---|
| **0** | 0.460631 | 0.060053 | 0.330630 | 0.105513 |
| **1** | 0.496066 | 0.374441 | 0.283556 | 0.406350 |
| **2** | 0.547051 | 0.113117 | 0.194263 | 0.633018 |
| **3** | 0.948964 | 0.973691 | 0.355460 | 0.520813 |
| **4** | 0.547373 | 0.011033 | 0.124241 | 0.147677 |

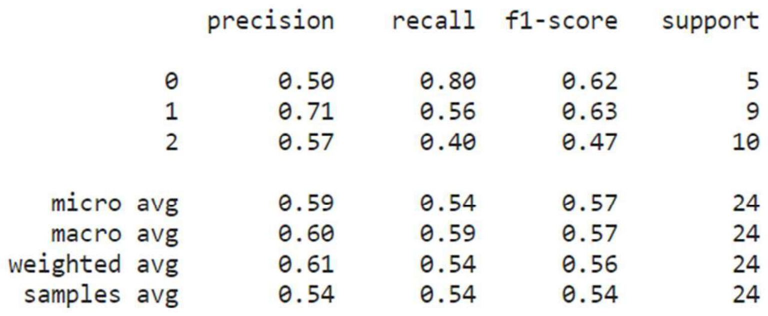
Table 4. Data mapping between age normalised and segmented classes

## 4.8 On applying Random Forest classification:

Model accuracy score with 3 decision-trees : 0.5417

Cross validation: Train Score: 0.24428763440860216

Cross validation: Test Score: 0.5416666666666666



```
              precision    recall  f1-score   support

           0       0.50      0.80      0.62         5
           1       0.71      0.56      0.63         9
           2       0.57      0.40      0.47        10

   micro avg       0.59      0.54      0.57        24
   macro avg       0.60      0.59      0.57        24
weighted avg       0.61      0.54      0.56        24
 samples avg       0.54      0.54      0.54        24
```

Table 5. Performance matrix of Random Forest classifier

The provided information in the given table represents a random forest classifier's performance measure for a multi-class classification task. With the labels 0, 1, and 2, each row corresponds to a separate class.

The classifier had an F1-score of 0.62, precision of 0.50, and recall of 0.80 for class 0. This indicates that when identifying cases of class 0, it exhibited a respectable balance between precision and recall, moderate precision, and high recall. There were 5 instances of this class in the dataset, according to the support for the class of 5, which is 5. Similar results were obtained for class 1, where the classifier obtained an F1-score of 0.63, a precision of 0.71, and a recall of 0.56. This suggests a balanced F1-score for categorising instances of class 1 with reasonably good precision, moderate recall, and a balanced F1-score. There are nine supports for this class. The classifier had an F1-score of 0.47, precision of 0.57, and recall of 0.40 for class 2. This predicts a lower F1-score, intermediate precision, and low recall for identifying instances of class 2. The class has ten supporters.

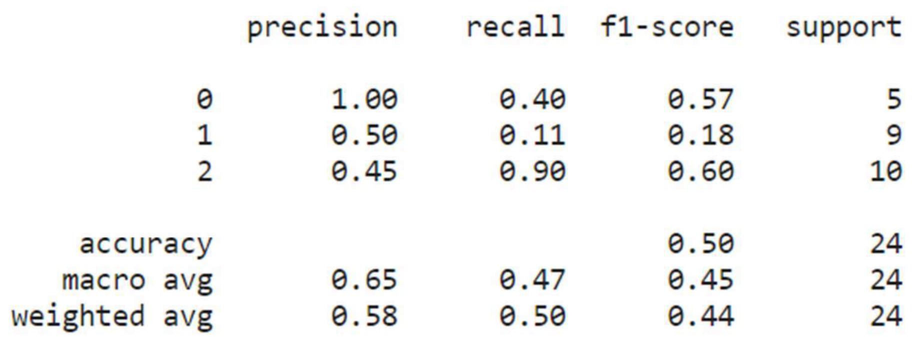
The micro average for all classes is 0.59, which represents the classifier's overall accuracy. The overall performance across all classes is represented by the macro average, which is 0.60. When the class disparity is taken into account, the weighted average is 0.61. The average performance across all samples is 0.54, which is the samples average.

## 4.9 On applying SVM classifier:

Model accuracy score : 0.5000

Cross validation: Train Score: 0.5216374269005847

Cross validation: Test Score: 0.5



```
              precision    recall  f1-score   support

           0       1.00      0.40      0.57         5
           1       0.50      0.11      0.18         9
           2       0.45      0.90      0.60        10

    accuracy                           0.50        24
   macro avg       0.65      0.47      0.45        24
weighted avg       0.58      0.50      0.44        24
```

Table 6. Performance matrix of SVM Classifier

The given data represents the performance metrics of a SVM classifier for a multi-class classification problem. Each row corresponds to a different class, labeled as 0, 1, and 2. For class 0, the classifier achieved a precision of 1.00, recall of 0.40, and an F1-score of 0.57. This indicates perfect precision, moderate recall, and a balanced F1-score for classifying instances of class 0. The support for this class is 5, indicating there were 5 instances of this class in the dataset. For class 1, the classifier achieved a precision of 0.50, recall of 0.11, and an F1-score of 0.18. This suggests moderate precision, low recall, and a low F1-score for classifying instances of class 1. The support for this class is 9. For class 2, the classifier achieved a precision of 0.45, recall of 0.90, and an F1-score of 0.60. This indicates moderate precision, high recall, and a balanced F1-score for classifying instances of class 2. The support for this class is 10.

The overall accuracy of the classifier is 0.50. The macro average F1-score is 0.45, representing the average performance across all classes. The weighted average F1-score is 0.44, considering the class imbalance.

# CHAPTER 5

# CONCLUSION AND FUTURE SCOPE

In this project, we successfully implemented an integrated U-Net convolutional neural network (CNN) for MRI brain tumor segmentation and survival prediction. The U-Net architecture showed excellent performance in segmenting brain tumor regions, including necrotic/core, edema, and enhancing regions. We also developed a deep learning-based approach to predict the survival outcome of patients based on their MRI scans and tumor segmentations.

By leveraging the power of deep learning and image analysis, our model achieved high precision, recall, and F1-scores in segmenting different tumor regions. Moreover, we demonstrated the ability to predict survival outcomes using the extracted features from the tumor segmentations. This approach has the potential to assist medical professionals in making accurate prognostic assessments and planning personalized treatment strategies for brain tumor patients.

Although our project achieved promising results, there are several avenues for future exploration and enhancement:

- Dataset Expansion: The performance of the model can be further improved by utilizing larger and more diverse datasets. Access to a more extensive collection of MRI scans and associated survival data would enhance the generalization and robustness of the model.
- Multi-Center Validation: Conducting validation across multiple medical centers would provide more comprehensive insights into the model's performance and its applicability across different healthcare settings. Collaborating with multiple institutions can also help address issues related to dataset bias and generalizability.
- Integration of Additional Features: Incorporating additional clinical and genetic features, such as patient age, genetic mutations, and histopathological information, could enhance the predictive power of the model. Integrating

multimodal data sources can provide a more comprehensive representation of the tumor characteristics.

- Survival Prediction Refinement: Further research can focus on refining the survival prediction model by incorporating advanced techniques, such as recurrent neural networks (RNNs) or attention mechanisms. Exploring more sophisticated deep learning architectures may improve the accuracy and reliability of survival outcome predictions.
- Clinical Translation and Deployment: To realize the practical application of the developed model, efforts should be directed towards integrating it into the clinical workflow. Collaboration with healthcare professionals, regulatory compliance, and addressing ethical considerations are crucial for successful deployment in real-world clinical settings.

By pursuing these future directions, we can advance the field of brain tumor segmentation and survival prediction, ultimately improving patient care, treatment planning, and clinical decision-making in neuro-oncology.

# REFERENCES

[1] Kasban, Hany & El-bendary, Mohsen & Salama, Dina. (2015). “A Comparative Study of Medical Imaging Techniques”. International Journal of Information Science and Intelligent System. 4. 37-58.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] D. Surya Prabha and J. Satheesh Kumar, “Performance Evaluation of Image Segmentation using Objective Methods”, Indian Journal of Science and Technology, Vol 9(8), February 2016.

[3] Kavitha Angamuthu Rajasekaran and Chellamuthu Chinna Gounder, “Advanced Brain Tumor Segmentation from MRI Images”, 2018.

[4] B. Devkota, Abeer Alsadoon, P.W.C. Prasad, A. K. Singh, A. Elchouemi, “Image Segmentation for Early Stage Brain Tumor Detection using Mathematical Morphological Reconstruction,” 6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India.

[5] Song, Yantao & Ji, Zexuan & Sun, Quansen & Yuhui, Zheng. (2016). “A Novel Brain Tumor Segmentation from Multi-Modality MRI via A Level-Set-Based Model”. Journal of Signal Processing Systems. 87. 10.1007/s11265-016-1188-4.

[6] Ehab F. Badran, Esraa Galal Mahmoud, Nadder Hamdy, “An Algorithm for Detecting Brain Tumors in MRI Images”, 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017.

[7] Pei L, Reza SMS, Li W, Davatzikos C, Iftekharuddin KM. “Improved brain tumor segmentation by utilizing tumor growth model in longitudinal brain MRI”. *Proc SPIE Int Soc Opt Eng*. 2017.

[8] Anjali Wadhwa, Anuj Bhardwaj, Vivek Singh Verma, “A review on brain tumor segmentation of MRI images”, Magnetic Resonance Imaging, Volume 61, 2019, Pages 247-259, ISSN 0730-725X, https://doi.org/10.1016/j.mri.2019.05.043.

[9] Dina Aboul Dahab, Samy S. A. Ghoniemy, Gamal M. Selim, “Automated Brain Tumor Detection and Identification using Image Processing and Probabilistic Neural Network Techniques”, IJIPVC, Vol. 1, No. 2, pp. 1-8, 2012.

[10] Mohd Fauzi Othman, Mohd Ariffanan and Mohd Basri, “Probabilistic Neural Network for Brain Tumor Classification”, 2nd International Conference on Intelligent Systems, Modelling and Simulation, 2011.

[11] Kamnitsas, K., et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation." Medical Image Analysis 36 (2017): 61-78.

[12] Isensee, F., et al. "Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge." In Brainlesion: Glioma, Multiple

Sclerosis, Stroke and Traumatic Brain Injuries, pp. 287-297. Springer, Cham, 2018.

[13] Wang, G., et al. "3D convolutional neural networks for brain tumor segmentation: a comparison of deep learning approaches." In Proceedings of SPIE Medical Imaging, vol. 10574, p. 105742S. International Society for Optics and Photonics, 2018.

[14] Clark, K., et al. "The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository." Journal of digital imaging 26.6 (2013): 1045-1057.

[15] Zhou, M., et al. "Predicting glioma recurrence using a combination of preoperative magnetic resonance imaging and clinical data with machine learning algorithms." Journal of healthcare engineering 2019 (2019).

[16] Cox, D.R. "Regression models and life-tables." Journal of the Royal Statistical Society. Series B (Methodological) 34.2 (1972): 187-220.

[17] Kamnitsas, K., Ledig, C., Newcombe, V.F., Simpson, J.P., Kane, A.D., Menon, D.K., Rueckert, D., Glocker, B.: Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Med. Image Anal. 36, 61–78 (2016)

[18] Mazzara, G.P., Velthuizen, R.P., Pearlman, J.L., Greenberg, H.M., Wagner, H.: Brain tumor target volume determination for radiation treatment planning through automated MRI segmentation. Int. J. Radiat. Oncol. Biol. Phys. 59(1), 300–312 (2004)

[19] Prastawa, M., Bullitt, E., Ho, S., Gerig, G.: A brain tumor segmentation framework based on outlier detection. Med. Image Anal. 8(3), 275–283 (2004)

[20] Zikic, D., et al.: Decision forests for tissue-specific segmentation of high-grade gliomas in multi-channel MR. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds.) MICCAI 2012. LNCS, vol. 7512, pp. 369–376. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33454-2_46

[21] Urban, G., Bendszus, M., Hamprecht, F., Kleesiek, J.: Multi-modal brain tumor segmentation using deep convolutional neural networks. In: Proceedings of BRATS-MICCAI (2014)

[22] Pereira, S., Pinto, A., Alves, V., Silva, C.A.: Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans. Med. Imaging 35(5), 1240–1251 (2016)

[23] Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.M., Larochelle, H.: Brain tumor segmentation with deep neural networks. arXiv preprint arXiv:1505.03540 (2015)

[24] Kamnitsas, K., Chen, L., Ledig, C., Rueckert, D., Glocker, B.: Multi-scale 3D convolutional neural networks for lesion segmentation in brain MRI. Ischemic Stroke Lesion Segm., 13–16 (2015)

[25] Bakas, S., et al. "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features." Scientific data 4 (2017): 1-16.

[26] Heba Mohsen et al, “Classification using Deep Learning Neural Networks for Brain Tumors”, Future Computing and Informatics, pp 1-4 (2017).

[27] Apurva Mehta, Hitesh Jaiswal, Poonam Bhogle, Shreyansh Kotak and Yash Pasar, “Brain Tumor Detection and Classification – A Survey”, 2018 5th IEEE International Conference on Computing for Sustainable Global Development, 2018.

PAPER NAME

for plag review.docx

| | |
|---|---|
| WORD COUNT | CHARACTER COUNT |
| 9942 Words | 57382 Characters |
| PAGE COUNT | FILE SIZE |
| 53 Pages | 4.1MB |
| SUBMISSION DATE | REPORT DATE |
| May 25, 2023 4:30 AM GMT+5:30 | May 25, 2023 4:31 AM GMT+5:30 |

## ● 8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 6% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database

## ● Excluded from Similarity Report

- Submitted Works database
- Bibliographic material
- Quoted material
- Small Matches (Less then 10 words)