

3D SEGMENTATION OF MRI MULTIMODAL FOR DETECTION OF BRAIN TUMOUR

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
with Specialisation in Artificial Intelligence

By

SANJAY D (Reg.No - 40731085)

VAYIGANDALA DEVARSH (Reg.No - 40733031)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR RAJIV GANDHISALAI,

CHENNAI — 600119

APRIL-2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SANJAY D (40731085)** and **VAYIGANDLA DEVARSH (40733031)** who carried out the Project work entitled "**3D SEGMENTATION OF MRI MULTIMODAL FOR DETECTION OF BRAIN TUMOUR**" under my supervision from NOVEMBER 2023 to APRIL 2024.

Internal Guide

Dr. E. MURALI, M.Tech., Ph.D

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Chemical - 600 119.

Submitted for Viva Voce Examination held on

15/4

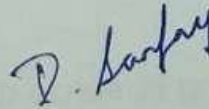
Internal Examiner

External Examiner

DECLARATION

I, **SANJAY D (Reg.No - 40731085)**, hereby declare that the Project entitled "**3D SEGMENTATION OF MRI MULTIMODAL FOR DETECTION OF BRAIN TUMOUR**" done by me under the guidance of **Dr. E. MURALI, M.Tech., Ph. D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Artificial Intelligence.**

DATE:



PLACE: Chennai

SIGNATURE OF THE CANDIDATE

DECLARATION

I **SANJAY D (Reg.No - 40731085)**, hereby declare that the Project entitled **“3D SEGMENTATION OF MRI MULTIMODAL FOR DETECTION OF BRAIN TUMOUR”** done by me under the guidance of **Dr. E. MURALI, M.Tech., Ph. D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Artificial Intelligence.**

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. E. MURALI, M.Tech., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science** who were helpful in many ways for the completion of the project.

ABSTRACT

Identifying brain tumors from MRI images can be challenging due to several factors, and it typically requires expertise in neuroimaging and collaboration between radiologists, neurologists, and other specialists. Some tumors may have a subtle or atypical appearance on MRI, making them difficult to distinguish from normal brain tissue. Brain tumors can be highly heterogeneous in terms of shape, size, and internal composition. Tumors can have different types of tissues, such as necrotic and enhancing regions, making it challenging to accurately delineate their boundaries. Brain tumors can be highly heterogeneous in terms of shape, size, and internal composition. Tumors can have different types of tissues, such as necrotic, cystic, or enhancing regions, making it challenging to accurately delineate their boundaries.

Deep learning has shown promising results in various medical imaging tasks, including the identification and classification of brain tumors, such as gliomas, from MRI images.

In this article, we examined the performance of prominent image segmentation models UNET, LINKNET FPN, PSPNET, and UNET+FPN, and we also constructed an ensemble model by averaging the predictions of the aforementioned models on 3D brain tumor segmentation. The above models are evaluated on the BraTS 2021 (Brain Tumour Segmentation) dataset, and comparing the metrics of each model reveals that UNET and, an enhanced model that combines UNET and FPN, UNET+FPN has the highest accuracy of 0.9959 and 0.9954 respectively

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF FIGURES	VII
	LIST OF TABLES	VIII
	LIST OF ABBREVIATION	IX
1	INTRODUCTION	1
2	LITERATURE SURVEY	4
3	METHODOLOGY	9
	3.1 Data preprocessing	9
	3.2 Training	9
	3.3 Evaluation Metrics	10
4	RESULTS	13
	4.1 UNET	13
	4.2 FPN	14
	4.3 LINKNET	16
	4.4 PSPNET	19
	4.5 UNET+FPN	21
	4.6 Comparison of Evaluation Metrics of Models	22
5	CONCLUSION	24
6	REFERENCE	25
7	APPENDIX	27
	A.SOURCE CODE	27
	B.SCREEN SHOTS	41
	C.RESEARCH PAPER	44
	D.CERTIFICATE	51

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	The Architecture of Ensemble Model	9
3.2	Visual representation of MRI	10
3.3	Overall Workflow of training of Models	12
4.1	Graph for training log of UNET	13
4.2	Graph for training log of FPN	16
4.3	Graph for training log of LinkNet	18
4.4	Graph for training log of PSPNet	20
4.5	Segmentation Result of models	22
B.1	Importing Required Libraries	41
B.2	Splitting Dataset into Training and Testing Set	41
B.3	Building the Model	42
B.4	Training and Loading the Model	42
B.5	Graph for Training Log of Model	43
B.6	Prediction on Testing Images	43

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO
2.1	Literature Review	7

LIST OF ABBREVIATION

ABBREVIATION	EXPANSION
MRI	Magnetic resonance image
AI	Artificial intelligence
FLAIR	Fluid-attenuated inversion recovery
BraTS	Brain Tumor Segmentation
FPN	Feature Pyramid Network
PSPNET	PYRAMID SCENE PRASING NETWORK
CamVid	Cambridge-driving labelled video database.
IOU	Intersection over union

CHAPTER 1

INTRODUCTION

Brain tumor segmentation is a critical task in medical image analysis that plays a pivotal role in the diagnosis, treatment planning, and monitoring of patients with brain tumors. Accurate delineation of tumor boundaries from magnetic resonance imaging (MRI) scans is essential for assessing tumor size, location, and progression. Currently, new cases of brain and CNS cancer have been identified, accounting for around 3% of all malignancies [1]. These situations are five times more common in European nations than in Asian countries. Glioma is the most common primary brain tumor. In adults, glioma accounts for 30% to 40% of all brain tumors and 80% of brain malignant tumors [2]. When malignant or non-cancerous brain tumors develop, they can cause brain damage. A brain tumor is a solid neoplasm inside the skull produced by uncontrolled and abnormal development of tissue or cells [4]. Brain tumors occur due to abnormal cell growth in the brain. Brain tumors develop as a result of aberrant cell proliferation in the brain. The anatomy of the brain is exceedingly intricate, with multiple areas in charge of diverse nervous system processes [5]. Tumors can develop anywhere in the brain or skull, including the protective linings, the brain's base, and other areas. There are several types of brain tumors, which vary depending on the tissue from which they develop [6][7].

Magnetic resonance imaging (MRI) is a widely utilized technique in radiology. It has become the ideal imaging modality for brain tumor detection and therapy due to its benefits of non-invasive, non-ionizing radiation, and high-contrast soft tissue imaging [8]. Magnetic resonance imaging (MRI) features are regarded as one of the preferable approaches to identify [9][10][11] a brain tumor because it produces a more detailed and clear picture than CT scans. Different

contrast images generated by multimodal MRI procedures give complementary information that aids in segmenting the brain tumor and its surrounding tissues, which is crucial in the diagnosis and treatment of brain tumors. Brain tumors are classified into three types: necrotic and non-enhancing tumors, peritumoral edema, and enhancing tumors. Native or T1-weighted MRI sequences are the most often used, followed by post-contrast T1-weighted (T1ce), T2-weighted, and Fluid Attenuated Inversion Recovery (FLAIR) volumes. Each of the modalities mentioned above has a distinct contrast, which aids in segmenting [8] various parts of tumors based on their features. Deep learning approaches, especially Convolutional Neural Networks (CNNs), have demonstrated remarkable success in various image analysis tasks, including medical image segmentation. CNNs excel at learning hierarchical features from data, enabling them to capture intricate patterns and spatial relationships within images. In the context of brain tumor segmentation, deep learning models can learn to distinguish between healthy and pathological tissue, as well as different tumor subtypes.

The goal of segmentation is to modify the way different parts of a picture are represented, making it simpler to comprehend sections of the image with distinct attributes. After dividing the brain image into thus many different pieces, each region becomes spatially continuous [12][13]. Traditionally, this segmentation was done manually by trained professionals, a time-consuming and subjective process prone to errors.

Brain tumors exhibit diverse morphological characteristics, and their precise segmentation from medical images is challenging due to variations in shape, size, and intensity. As a result, automated segmentation of brain MRI images can considerably improve diagnostic and treatment operations, especially when radiologists and competent professionals are few [14] [15]. However, precise tumor segmentation remains a difficulty due to glioma variability in terms of shape, size, and appearance, as well as the unclear and fuzzy

boundary that exists between cancer and brain tissue [16]. The MRI data's intensity fluctuation adds to the difficulties [17]. As a result, it is still subject to development, allowing for more research into improved segmentation approaches and accuracy. Our study considered five image segmentation models.

UNET, LINK NET, FPN, PSPNET, and UNET+FPN for localizing and detecting the braintumor from the 3D MRI images. The main objectives of this study are:

- Automatic brain tumor segmentation method that uses informative image slices taken from 3D multimodal MRI volume to reduce computational time and increase segmentation accuracy
- Comparative analysis of image segmentation models based on the evaluation metrics for brain segmentation
- A hybrid model combined of above-mentioned models using an average ensemble technique and study of its performance on test data

CHAPTER 2

LITERATURE SURVEY

On November 25, 1884, MR. Rickman J. carried out the first recognized removal of the primary brain tumor, which was discovered on the same day. Harvey Cushing, known as "The Father of Neurosurgery" in the 1900s, performed surgeries on brain tumors. This section provides a brief overview of the proposed model-based designs and the standard U-Net used for segmenting MRI images. Numerous alternative segmentation architectures have been proposed, utilizing methods such as thresholding, region-growing, edge-based detection, mathematical morphology, and more, for the early detection of tumors. The U-Net model was specifically designed for identifying and segmenting diseases from medical images.

The article discusses the importance of segmenting brain tumors from MRI scans and suggests that AI methods can help automate this process. The study aims to develop a deep-learning model specifically for brain tumor segmentation. The authors review existing segmentation models and highlight the popularity of the 3D U-Net model. The proposed model is similar to the 3D U-Net architecture and consists of 6 encoding and decoding layers. It is trained on the BRATS dataset using dice loss and focal loss. The model achieves high accuracy, loss, and IOU scores. Comparative experiments show that the proposed model outperforms other models. Ablation studies are conducted to analyze the impact of different layers. Overall, the proposed model achieves state-of-the-art performance in brain tumor segmentation [20].

The paper focuses on semantic image segmentation, assigning class labels to pixels in an image for scene understanding. Existing methods are accurate but too slow for real-time applications like self-driving cars. The authors propose LinkNet, a CNN architecture that achieves real-time performance while maintaining accuracy. LinkNet uses an encoder-decoder structure, with the encoder based on pre-trained ResNet-18 and

the decoder upsampling features to full resolution. Shortcut connections preserve spatial information. The authors trained LinkNet on CamVid and Cityscapes datasets, using cross-entropy loss, data augmentation, and class weighting. They optimized with RMSProp and trained on 4 GPUs. LinkNet achieved state-of-the-art accuracy on CamVid and competitive results on Cityscapes. It demonstrated real-time performance on GPUs and the NVIDIA TX1 embedded system [21].

The "BMC Medical Imaging" presents a proposed algorithm to improve the U-Net network and the improved strategy of FPN, as well as data enhancement technology, to segment the brain tumor region effectively. The authors applied a Dice loss function to solve the imbalance between classes. They used three evaluation indexes to evaluate the model's performance, which are the dice similarity coefficient (DSC), the Jaccard index, and accuracy. The results showed that the algorithm proposed in this paper had better performance compared to the traditional U-Net network in segmented brain tumor regions. The authors used the NVIDIA GeForce RTX Quadro K620 single GPU for hardware implementation, and the software environment was the Keras 2.2.4 library at the back end of TensorFlow 2.1. The input image size of the network model was 256X256X3, and the batch size was set to 8, while the number of training epochs was set to 150. The Adam optimizer was used to update the parameters. The paper also described how the network was trained under the same settings to ensure the fairness of the experimental results. Overall, the paper provides a detailed description of the proposed algorithm and the experiments conducted to evaluate its performance. Although more experiments may be needed to validate the precise effectiveness of this approach, the paper showcases an improvement in the segmented brain tumor region utilizing the proposed U-Net network with FPN and data enhancement techniques [22].

A revised U-Net framework has been developed to effectively detect and segment brain tumors in MRI images with exceptional precision. The performance of this model was assessed using authentic images

from the BRATS 2020 datasets provided by Medical Image Computing and Computer-Assisted Interventions, resulting in an impressive test accuracy of 99.4%. Furthermore, the research paper emphasizes the superior performance of the proposed U-Net model compared to alternative deep learning-based models. In conclusion, this study introduces a dependable and precise algorithm for automated brain tumor detection and segmentation, offering significant implications for medical image processing and cancer diagnosis [23]. The models they have used do not have a specialty but how he took the data. Normally everyone took the data set that was pre-defined but the author took only a few images of MRI and he took the images and took the images in different sortings like the sagittal, axial, and coronal using deep learning. The "Open Computer Science 2022" by Suchismita Das et al. details the model architecture used for tumor segmentation in medical images. The model utilizes a U-net-based architecture with contracting and expanding paths. The authors have also implemented various data augmentation techniques to avoid over-fitting of the model. The study provides a comparative analysis of the proposed model with other state-of-the-art models used in tumor segmentation in medical images. However, the PDF pages do not contain information about specific dates of the publication or the author's conclusions of the study [24].

The models they have used do not have a specialty but how he took the data. Normally everyone took the data set that was pre-defined but the author took only a few images of MRI and he took the images and took the images in different sortings like the sagittal, axial, and coronal using deep learning for individual link net. The images are grayscale and the foreground of the images is located at the center. Images are captured from different views of the skull, thus the size and position of the tumors vary in different angles. To improve the segmentation accuracy, the authors sorted the data into sagittal, coronal, and axial images and trained three separate LinkNet networks. About 900 images were used for training in each group and around 200 were selected for test purposes [25].

Based on pyramid scene parsing (PSP) for semantic segmentation. The proposed model uses convolutional neural networks (CNN) and pyramid pooling modules to extract spatial information from images. PSPNet architecture is employed to improve the prediction accuracy of the model. The paper compared the performance of PSPNet with UNet on a public dataset and found that PSPNet obtains higher segmentation accuracy. The paper also discusses the effectiveness of pyramid pooling modules in the PSP structure and the visualization of pyramid structures. Overall, the paper mainly focuses on the development, evaluation, and visualization of the proposed PSPNet architecture for semantic segmentation [26].

The paper titled "Brain Tumor Segmentation Using Deep Learning" proposes a new method for automatically segmenting brain tumors in MR images. The paper does not require preprocessing steps, and it evaluates the segmentation results using the Dice criterion. The study was conducted using a server with an Intel Core i7-4790K processor, 32 GB of RAM, and two NVIDIA GeForce GTX Titan X GPU cards with scalable link interface (SLI). The authors trained a deep neural network called LinkNet on different angles of the dataset. They found that separating the dataset by angle resulted in better segmentation accuracy. The paper concludes that the proposed method is a useful tool for doctors in segmenting brain tumors in MR images. The study was published on October 15, 2018 [27].

Research work	Algorithm	Data set	Result
They had created a mask model that classify the regions . [20]	U NET	BRATS 2020	The proposed model with 6 encoder-decoder layers achieved the best performance Unet accuracy of 0.9759 loss of 0.824

They had created a PSP net better than a unet but it can skip some layers[21]	PSP NET	BRATS-2018	PSP Net performance in the whole tumour segmentation, with a Dice coefficient of 0.4719.
A hybrid model which is a combination of existing models	U-NET, SEG NET	BRATS 2017	The accuracy ranges from 0.9740 to 0.9901 for different sub-regions
the visualization of PSPNet features using Gradient-weighted Class Activation Mapping (GradCAM)	PSP NET GRADCAM	BRATS 2018	achieving an overall (DSC) of 0.870 on the test set of the BRATS2018 dataset
A U-shaped variation of LinkNet that utilizes residual module and switches from stacking to adding deep and shallow feature synthesis approaches.[9]	Modified LINK NET	BRATS 2019	The overall accuracy achieved on the BRATS dataset was 99.51%, and the overall accuracy on the MRI dataset was 99.63%.

Table 2.1:Literature Review

CHAPTER 3

METHODOLOGY

The primary goal of this research is to create a brain tumor segmentation algorithm that can segment the four sub-tumor sections for all modalities by comparing current deep learning models and also an ensemble model created by using the average ensemble technique by combining the predictions of each model figure 3.1. The dataset used for this study is BraTS 2021, which consists of 1250 image files each image file has comprises multimodal MRI scans, including T1-weighted, T2-weighted with contrast enhancement, T2-weighted, and fluid-attenuated inversion recovery (FLAIR) images, each image as the shape of (128,128,155) These different modalities provide complementary information about the brain and tumor tissue.

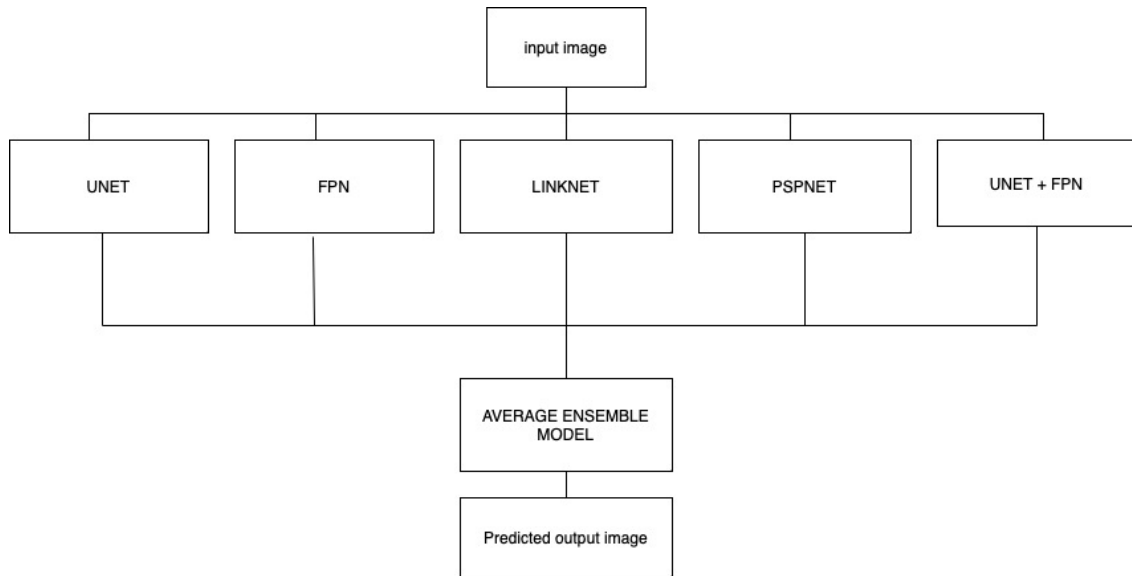


Figure 3.1 : Architecture of ensemble model

3.1 Data preprocessing:

The dataset is split into training, validation and testing sets. The MRI images FLAIR, T1-weighted (T1ce) are used as input data(X) and segmented used for ground truth image (y). Each image is reshaped and sliced with image size of 128. One-hot encoding on the segmentation labels. It converts the segmentation data into a one-hot encoded format with four classes. Nibabel library is used for reading and writing neuroimaging file formats. It supports

various file formats commonly used in neuroimaging, including NIfTI and Analyze . Nibabel is useful for loading MRI data from files, extracting metadata, and converting data to NumPy arrays for further processing and visualization. The visual representation of MRI image data shown in fig(3.2) .

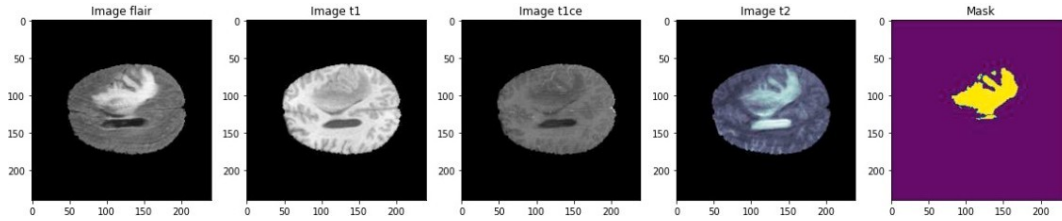


Figure 3.2 :Visual representation of MRI Image

3.2 Training:

Each model was trained on 30 epochs using 300 training images and 250 validation images per epoch to monitor and analyze model performance throughout training. During model training, the callbacks function is utilized; the early callbacks function is used to verify validation accuracy with patience of 5 to stop training if there is no progress in validation accuracy after 5 epochs.. Category cross entropy is a loss function that is often used for multiclass classification tasks in which each input sample belongs to one of many classes. The categorical crossentropy loss is suitable when the labels are one-hot encoded. Adam optimiser implemented, which is an adaptive learning rate optimization algorithm that combines the advantages of both AdaGrad and RMSProp with learning rate of 0.001.

3.3 Evaluation Metrics:

To evaluate the performance of the proposed models, we considered the Mean Intersection Over Union (IoU), Dice Coefficient (DSC), Sensitivity, Specificity, and Accuracy as the figures of merit. These metrics can be formulated using TP, FP, TN , and FN, which are the abbreviations of the number of instances for true positives , false positives, true negatives, and false negatives, respectively. The metrics are defined as follows

3.3.1 Intersection Over Union :

This is calculated by the ratio of the area of overlap (true positives) to the total area covered by both the predicted and ground truth regions (sum of true positives, false positives, and false negatives). It quantifies how well the predicted region aligns with the ground truth region.

$$IOU = \frac{Truepositive}{Falsenegative + Falsepotive + Truenegative} \quad (1)$$

3.3.2 Dice Coefficient:

This is also a calculation of overlap-based metric to measure the spatial overlap between the ground truth mask and predicted mask.

$$Dc = \frac{2 * target}{traget + prediction} \quad (2)$$

3.3.3 Sensitivity:

It is the percentage of actual positive values correctly identified.

$$Sensitivity = \frac{TP}{TP + FN} \quad (3)$$

3.3.4 Specificity:

It s the percentage of actual negative values correctly identified.

$$Specificity = \frac{TN}{TN + TP} \quad (4)$$

3.3.5 Accuracy:

The is a probabilistic metric that measures the degree to which the segmentation results agree with the ground truth segmentation mask.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

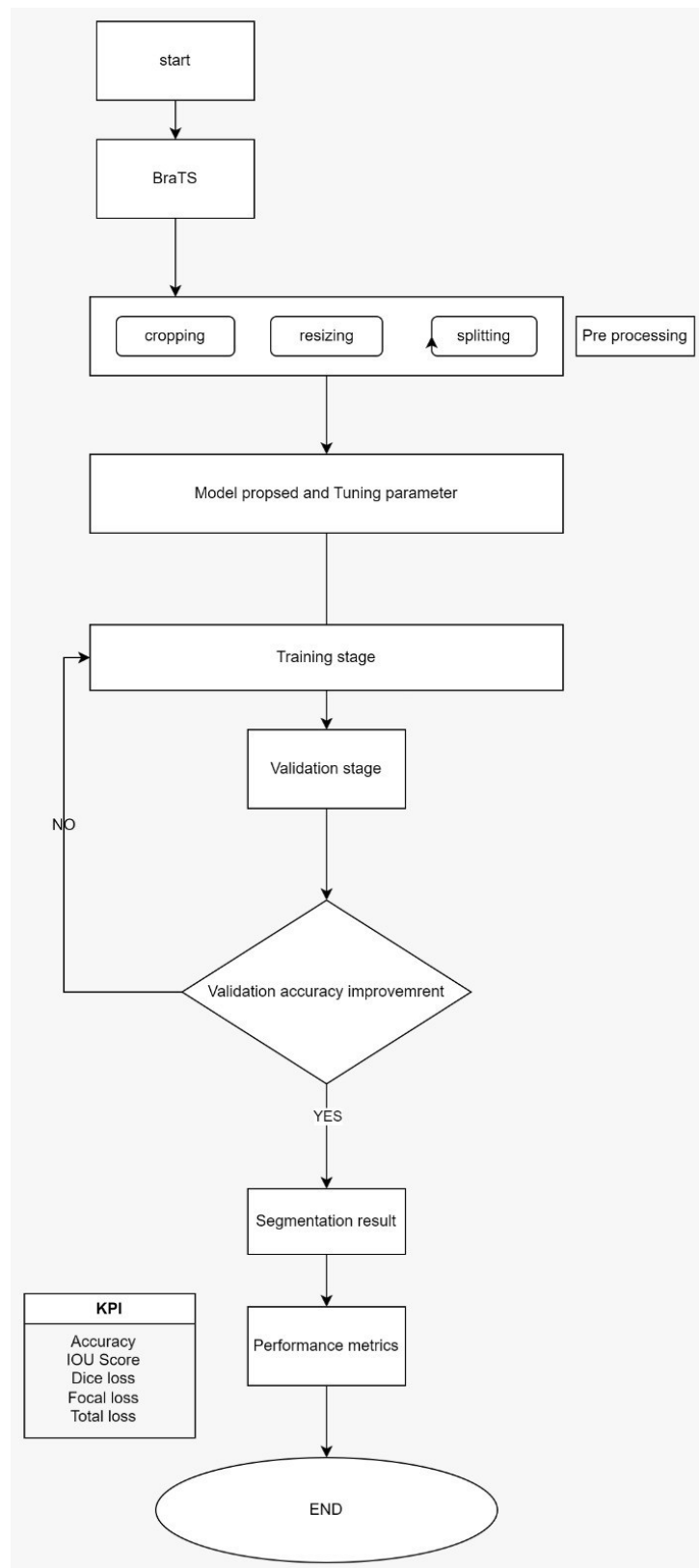


Figure 3.3: Overall workflow of training for Models

CHAPTER 4

RESULT

In this part, we will go over the experimental findings, network architecture for all of the models, model parameters, training details, and segmentation outcomes. Finally, a comprehensive comparison of proposed deep learning models is done using several assessment measures.

4.1 UNET :

Ronneberger et al. introduced UET, an architecture for semantic segmentation that employs a Fully Convolution Network Model[18]. The main objective of semantic image segmentation is to assign a class to every pixel in an image, representing its meaning. This task is commonly called dense prediction since we are making predictions for each pixel [19]. The contracting path and expanding path together make up the model. The contracting path has 5 encoders. The first encoder has two convolution layers with a kernel size of 3x3. The remaining encoders have max pooling with a kernel size of 2x2 and two convolution layers with a kernel size of 3x3.

The expanding path has four decoders. Each decoder has three convolution layers where one layer has a kernel size of 2 and the remaining layer has a kernel size of 3. Each convolution has a ReLU function and the kernel initializer.

The middle layer of the Model has a dropout of 0.5. The output layer of the model has a convolution of 1x1 to divide each component feature vector into a relevant number of classes.

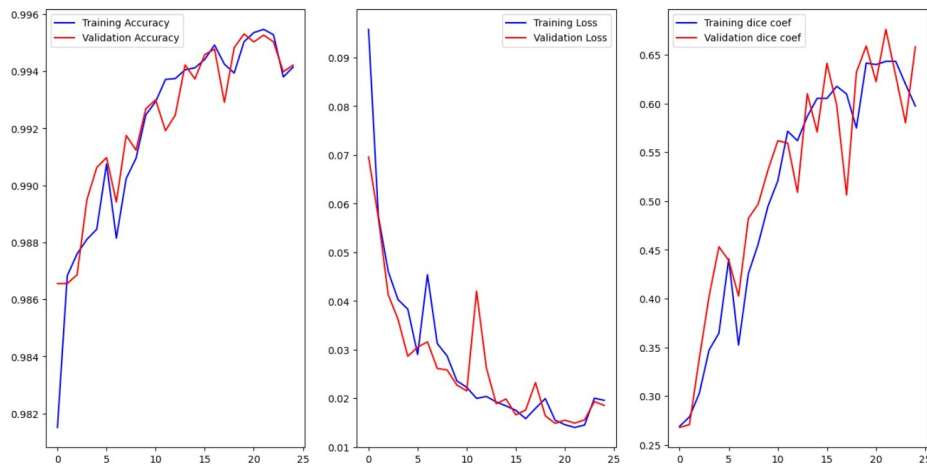


Figure 4.1 : Graph for training log of UNET

On seeing the graph (Fig 4.1) It is evident that the model's accuracy improves with each epoch of training. Additionally, the validation accuracy consistently surpasses the training accuracy. This suggests that the model is effectively generalizing to new data and is not overfitting to the training data. The loss graph certainly demonstrates that the model's loss decreases with each epoch of training. Furthermore, the validation loss consistently remains lower than the training loss.

4.2 FPN :

The Feature Pyramid Network (FPN) is a popular architecture used for object detection tasks in computer vision. FPN addresses the need for precise localization and strong semantic features by constructing a pyramid of feature maps at different scales. Each level of the pyramid captures semantic information at a specific scale and has a different resolution[20]. To achieve this, FPN utilizes two main techniques were used Firstly, the top-down pathway involves upsampling high-resolution but semantically weak features from shallower layers and combining them with lower-resolution but semantically strong features from deeper layers. This fusion of spatial detail and semantic context enhances the overall performance.

Secondly, FPN employs lateral connections, which directly add high-resolution features from shallower layers to their corresponding counterparts in the pyramid. This preserves fine-grained spatial information, further improving the detection accuracy.

To construct the pyramid, FPN adds 1x1 convolutional layers on top of each high-level feature map from the CNN. These layers bring the features to the same resolution as the high-resolution feature maps by upsampling. The upsampled features are then directly added to their corresponding counterparts at each level of the pyramid through lateral connections.

In the top-down pathway, the upsampled features are element-wise summed with the corresponding lower-level features in the pyramid. This integration of information from different scales and resolutions enhances

the ability of FPN to detect objects accurately and efficiently.

The FPN architecture in the bottom-to-up path is composed of four convolutions. These convolutions have a respective number of filters [256, 512, 1024, 2056] and strides of [4, 8, 16, 32], with a kernel size of 3x3. On the other hand, the top to down path has four conditions, each with 256 filters and a kernel size of 1x1. Additionally, there are three samplings: up1, up2, and up3.

Each sampling has 256 filters and kernel sizes of 2x2, 2x2, and 8x8 respectively. The up samplings up1, up2, and up3 are combined with their nearest neighbors in the bottom to up path using the Add() skip connection.

Furthermore, the model includes a decoder path consisting of four decoders. Each decoder has two convolutions and samplings. The convolutions have 128 filters and a kernel size of [128, 64, 32] to recover the spatial information from each Add() layer.

Finally, the output of the Up sampling is concatenated into a single layer. Then, a convolution with a softmax activation function is applied to generate the final output. This convolution consists of 4 filters with a kernel size of 1x1. By using a 1x1 convolution, the operation is independently applied to each spatial location without considering neighboring pixels.

The utilization of a 1x1 convolution effectively combines the channels in the concatenated layer, resulting in a reduction of the channel to 4. Subsequently, the softmax function is applied element-wise, producing a probability distribution across the 4 channels. Except for the output layer, the FPN model consists of 18 convolutions, each of which uses the 'ReLU' function.

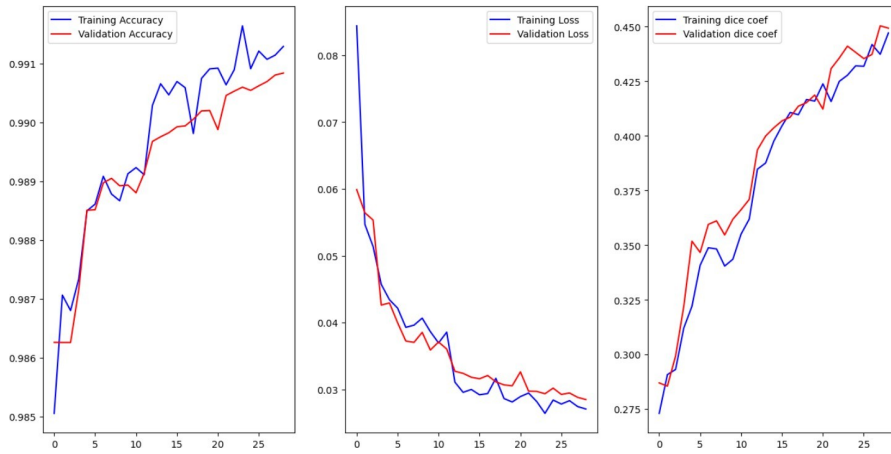


Figure 4.2 : Graph for training log of FPN

The graph Fig (4.2) shows that training accuracy and validation Accuracy are increasing over time, indicating that the model is learning effectively and improving its prediction capabilities. Both training Loss and validation Loss decrease over time, indicating an improvement in model training. training dice coefficient and validation dice coefficient increase over time, indicating an improvement in the model's segmentation capability

4.3 LINKNET:

LinkNet is a compact deep neural network architecture specifically designed for semantic segmentation. Its purpose is to divide an image into various sections and assign labels to each section. LinkNet consists of two primary components:

1. **Encoder:** This component takes the input image and extracts relevant features from it. It employs a convolutional neural network (CNN) such as VGG or ResNet for this task. As the network delves deeper, the extracted features become increasingly abstract and encompass higher-level details about the image.
2. **Decoder:** This component utilizes the features obtained from the encoder to predict the segmentation mask. It performs up sampling on the features and combines them with information from earlier stages of the encoder. This process aids in recovering spatial information and enhancing the accuracy of the predictions.

The proposed model in this is composed of five encoder blocks. Each block comprises three convolutional layers with varying kernel sizes and strides. The number of filters is progressively increased in each subsequent block. The initial convolutional layer, conv1, utilizes 32 filters, a kernel size of 7x7, and a stride of 2. Additionally, batch normalization is applied to the output of the first convolutional layer. To reduce spatial dimensions, max pooling with a pool size of 3x3 and a stride of 2 is employed. The encoder part commences with a sequence of convolutional layers, encoder1, with different kernel sizes and strides, followed by batch normalization. Residual connections are established through the add operation. This process is repeated for the remaining encoders (encoder2, encoder3, encoder4, encoder5), which possess larger filter sizes of 64, 128, 256, and 512 respectively there is four decoder blocks, Each decoder block consists of convolutional layers, batch normalization, and transposed convolutional layers.

The number of filters decreases with each subsequent block. Skip connections are incorporated between the corresponding encoder and decoder blocks, facilitating the preservation of spatial information and enhancing the gradient flow during the training process. The ultimate output block generates the segmentation map by utilizing transposed convolutional layers, culminating in a softmax activation function that yields class probabilities for individual pixels. The kernel size employed for this purpose is 4. A convolution with a size of 1x1 is performed on the output of the last encoder block

(encoder5) to reduce the number of channels. To normalize the activations, batch normalization is applied to the transposed convolution (Conv2DTranspose) with a kernel size of 3 and a stride of 2, which is used to upsample the feature map. After the transposed convolution, batch normalization is applied again. Another 1x1 convolution is then applied to further adjust the number of channels. To preserve spatial information and facilitate gradient flow during training, a skip connection is established by adding the output of the decoders to the corresponding encoder blocks.

The model mentioned above was terminated early after 5 epochs due to the absence of any improvement in val_accuracy, to prevent model overfitting.

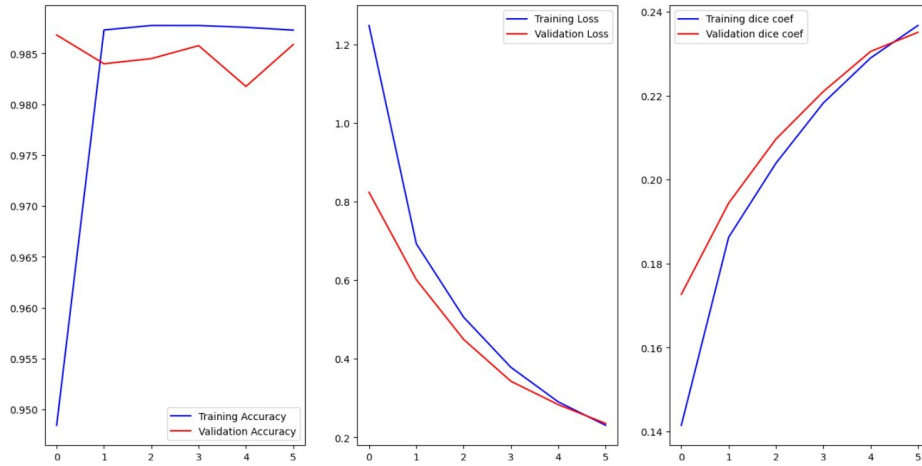


Figure 4.3 : Graph for training log of LinkNet

When viewing Figure 4.3, Initially, the training accuracy is around 0.965, but it quickly rises to around 0.985 within the first epoch. Following that, it remains pretty steady for the rest of the epochs. On the contrary, the validation accuracy starts at a higher value of around 0.975 but shows slight fluctuations during the epochs. Moving to the middle graph represents the training and validation loss over five epochs. Both metrics exhibit a decreasing trend over time, indicating an improvement in the model's performance with each epoch. The training loss starts at a higher value but decreases more rapidly compared to the validation loss. As the epochs progress, the training loss converges towards the validation loss. Lastly, the third graph on the right demonstrates the training and validation dice coefficients over five epochs. Both coefficients show an increasing trend over time, suggesting that the model becomes more proficient in accurately segmenting images. In summary, these three graphs offer valuable insights into the performance of the machine learning model. They show that the model's

performance improves over time, with significant enhancements in training accuracy and loss in the first epoch. Although the validation accuracy and loss fluctuate slightly, the model's ability to correctly segment images shows improvement. These graphs can be utilized to further optimize the model's performance.

4.4 PSPNET:

The PSPNet, also known as the Pyramid Scene Parsing Network, is a deep-learning architecture designed specifically for image segmentation. In this task, each pixel in an image is assigned a category label. Developed in 2016, the PSPNet has proven to be highly effective in capturing global context information, resulting in more precise and detailed segmentation outcomes compared to previous methods. The image is processed through a series of convolutional layers and pooling operations, which extract features from the image at various scales. The key innovation of the PSPNet is the Pyramid Pooling Module, which aggregates features from different regions of the image at multiple scales using four parallel pooling branches. One of these branches, known as global average pooling, captures the overall essence of the image.

The Pyramid Scene Parsing Network utilizes convolutional blocks, base feature maps, pyramid feature maps, and the last convolutional module functions. The convolutional block function defines a block with dilated convolutions and a residual connection, taking an input tensor X and a tuple of filters specifying the number of filters for each convolutional layer in the block. This block consists of three convolutional layers with Leaky ReLU activations and batch normalization. A skip connection is added by element-wise addition of the output of the third convolutional layer and the input tensor. The final output is then passed through a ReLU activation. On the other hand, the base feature maps function creates the base feature maps by stacking two convolutional blocks. The first block has filters [32, 32, 64], and the second block has [64, 64, 128] filters. The output of the second block is returned as the base feature maps.

The pyramid feature maps function is responsible for generating pyramid feature maps from an input layer. It utilizes the `base_feature_maps` function to obtain the base feature maps. Afterward, it applies global average pooling on the base feature maps and reshapes the resulting feature maps. To create feature maps at different scales, multiple

convolution and upsampling operations are performed. Finally, the output is obtained by concatenating the original base feature maps with the feature maps created at different scales.

Convolution in the pyramid feature map function utilized the interpolation parameter 'bilinear'.

This parameter determines how pixel values are estimated during image resizing or transformation. Bilinear interpolation involves considering the values of the four closest known pixels to approximate the value of a new pixel. It calculates a weighted average of these four pixels, giving more weight to the closer ones. The last function of the convolutional module takes an input layer and constructs the final convolutional module. It utilizes the `pyramid_feature_maps` function to obtain the pyramid feature maps. To generate the final output with 4 channels, a 1x1 convolutional layer with softmax activation is applied. The architecture of the model utilizes dilated convolutions, residual connections, and pyramid feature maps to capture multi-scale information. The base feature maps capture lower-level features, while the pyramid feature maps incorporate information at different scales. The final convolutional module produces a softmax output for segmentation with 4 channels. During the training of the PSP Net model, it was stopped by callbacks after five epochs.

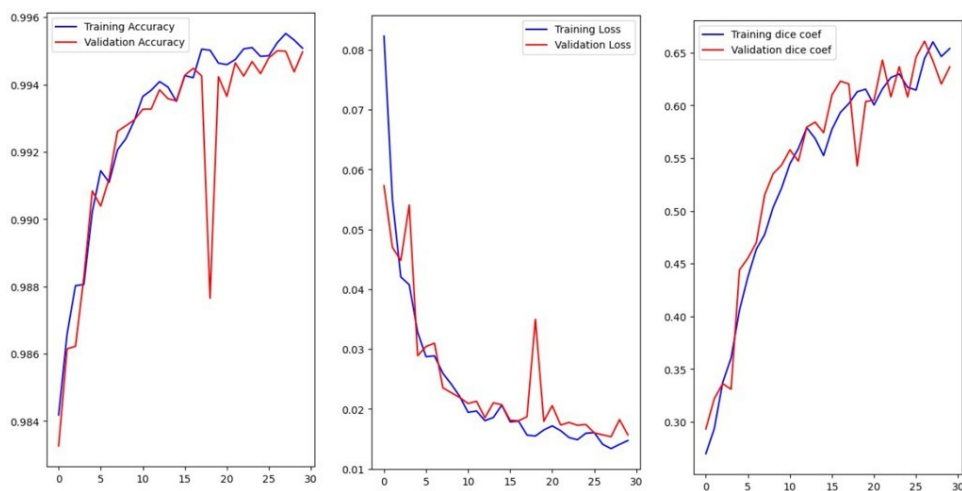


Figure 4.4 : Graph for training log of PSPNet

The graph Figure 4.4 shows The training Accuracy is increasing while validation Accuracy decreases after epoch 2, indicating a potential issue like overfitting. in red. Both training and validation loss decrease over epochs but validation Loss increases sharply at epoch 5, another indication of a problem with model training. Both training and validation dice coefficients increase but there's a noticeable gap between them, suggesting that the model performs better on training data than validation data. In summary, the model is not learning effectively and might be overfitting.

4.5 UNET+FPN:

Upon comparing the architecture of FPN and U-Net, it becomes evident that they sharesimilarities. The horizontal connection present in FPN can be achieved through the horizontal connection in U-Net. This feature enables the expansion of the U-Net model using the FPN structure. By leveraging the U-Net structure to its fullest potential, the U-Net model effectively utilizes information from various scales

Initially, the training loss decreases rapidly and then stabilizes. On the other hand, the validation loss (represented by the red line) experiences a spike b4efore decreasing and stabilizing as well. This pattern indicates a potential case of overfitting at the beginning, where the model is overly fitted to the training data. On the other hand, the graph on the right side presents the dice coefficients for the training and validation datasets throughout the epochs.

Higher values in the graph indicate improved performance in terms of similarity between the predicted and actual results in tasks like image segmentation. As time progresses, both coefficients gradually increase but with some fluctuations, suggesting that the model is continuously learning and making adjustments.

4.6 Comparison of Evaluation Metrics of Models:

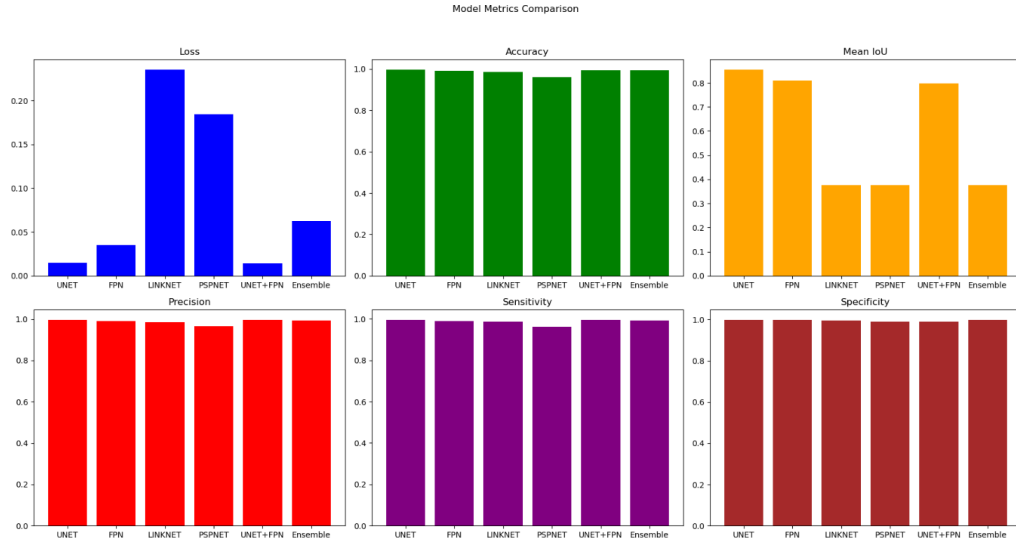
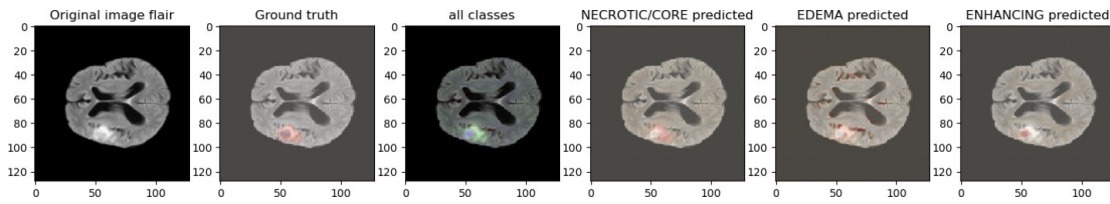
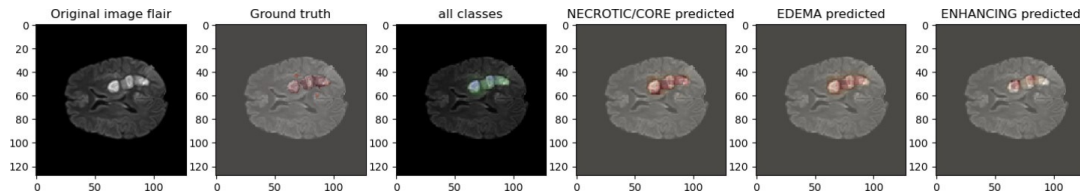


Figure 4.5 Evaluation metrics comparison between the models

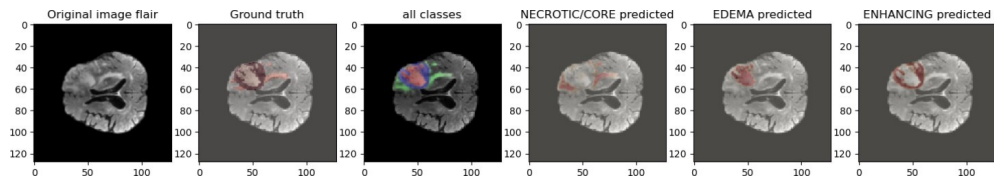
Figure 4.5 shows that models UNET, FPN, and UNET+FPN have the lowest loss, suggesting superior model performance, whereas LINKNET and PSPNET have the largest loss, showing poor model performance. In this experimental study PSPNET has the lowest accuracy compared to other models, Except PSPNET every model has similar accuracy with slight differences. In comparison to the other three models, UNET, FPN, and UNET+FPN have the greatest Mean IoU score. Except for PSPNET, all models in the research had the same accuracy and sensitivity score.



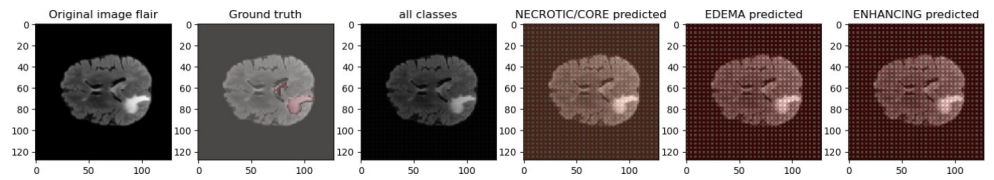
(a)



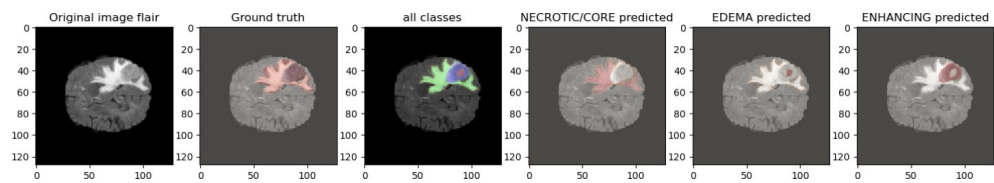
(b)



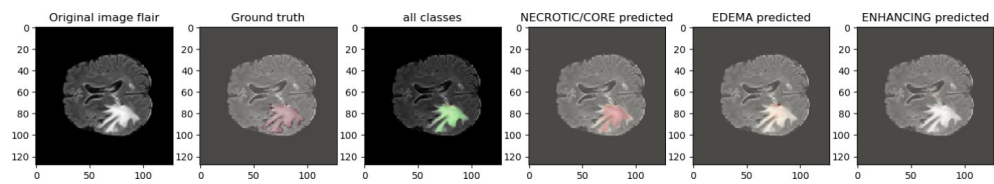
(c)



(d)



(e)



(f)

Figure 4.6 Segmentation results of models (a) UNET, (b) FPN, (c) LINKNET (d) PSPNET, (e) Ensemble model, (f) UNET+FPN

CHAPTER 5

CONCLUSION

The main objective of this study is to evaluate the effectiveness of various deep learning models for image segmentation, specifically for the rapid identification of brain tumors. The process of segmenting medical images with the assistance of radiologists can be extremely time-consuming and resource-intensive, making it impractical in remote areas. Therefore, the implementation of advanced automation techniques for brain tumor detection can greatly benefit a significant number of cases. In order to address this issue, various models were utilized in this article, namely UNET, Linknet, FPN, PSPnet, and UNET+FPN. Each model demonstrated different levels of accuracy during the data testing phase. Additionally, an ensemble model was created by averaging the predictions from each individual model. The model UNET, UNET+FPN exhibited the highest accuracy among all models, while the UNET+FPN greater accuracy model followed closely behind. However, it is worth noting that PSPnet and Linknet performed inadequately in comparison to the other models. FPN and Ensemble model achieved stable performance. As a result, physicians can strategize treatment and monitor the progression of cancers during the diagnostic stage. The specific area of interest is divided into segments, and the presence of the tumor can be determined using an exceptionally precise model. This method offers the benefit of enhancing image segmentation and spatial localization to a greater extent, resulting in superior performance compared to previous systems. Furthermore, it requires less computational time and can be trained at a faster pace compared to networks with fewer parameters. In the forthcoming studies, the emphasis will be on employing diverse classifier techniques to enhance precision while reducing error rates. Additionally, this approach could be tailored to predict the prognosis of individuals afflicted with brain tumors. In future research, the utilization of larger and more varied datasets may prove beneficial in conducting tests under real-world circumstances and clinical trials.

REFERENCE

1. Zeljkovic V, et al. (2014). "Automatic brain tumor detection and segmentation in MRimages," 2014 Pan American Health Care Exchanges (PAHCE).
2. Ronneberger, O., Fischer, P., Brox, T (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: Medical Image Computing and Computer- Assisted Intervention. pp. 234–241 (2015).
3. A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 2017, pp. 1-4, doi: 10.1109/ VCIP.2017.8305148.[3]
4. Bakas S, Reyes M, Jakab A, Bauer S, Rempfler M, Crimi A, et al. (2018). "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprintarXiv:1811.02629 (2018).
5. Kumar S, Negi S, Singh JN, Gaurav A (2018). "Brain Tumor Segmentation and Classification Using MRI Images via Fully ConvolutionNeural Networks," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN),pp. 1178-1181.
6. M. B. Naceur, R. Saouli, M. Akil and R. Kachouri, "Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in MRI images",Comput. Methods Programs Biomed., vol. 166, pp. 39-49, Nov. 2018.
7. G. Wang, W. Li, S. Ourselin and T. Vercauteren, "Automatic brain tumor segmentation based on cascaded convolutional neural networks with uncertainty estimation", Frontiers Comput. Neurosci., vol. 13, pp. 56, Aug. 2019.
8. Cherguif H, Riffi J, Mahraz,M A, Yahyaouy A, Tairi A (2019). "Brain Tumour Segmentation Based on Deep Learning," 2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS), pp. 1-8
9. Kenneth A, et al.(2019). "Challenges to curing primary brain tumours." Nature reviews. Clinical oncology vol. 16,8 (2019): 509-520.
- 10.K. K. Farmanfarma, M. Mohammadian, Z. Shahabinia, S. Hassanipour, and H. Salehiniya, "Brain cancer in the world: an epidemiological review," World Cancer Res. J., vol. 6, no. 5, pp. 1–5, 2019.

11. S. Das, "Brain Tumor Segmentation from MRI Images Using Deep Learning Framework," in: Progress in Computing, Analytics and Networking, Springer, Singapore, 2020, pp. 105–114.
12. Nema S, Dudhane A, Murala S, et al. RescueNet: An unpaired GAN for brain tumor segmentation[J]. Biomed Signal Process Control. 2020;55:101641–52.
13. Das, G. Nayak, S. Saxena, S. C. Satpathy, "Effect of learning parameters on the performance of U-Net Model in segmentation of Brain tumor," Multimed. Tools Appl., pp. 1–19, 2021.
14. S. Das, S. Bose, G. K. Nayak, S. C. Satpathy, and S. Saxena, "Brain tumor segmentation and overall survival period prediction in glioblastoma multiformed using radiomic features," Concurrency Comput.: Pract. Experience, p. e6501.
15. S. Das, G. Nayak, S. Saxena, S. C. Satpathy, "Effect of learning parameters on the performance of U-Net Model in segmentation of Brain tumor," Multimed. Tools Appl., pp. 1–19, 2021.
16. Zhao, M., Xin, J., Wang, Z., Wang, X., & Wang, Z. (2022). Interpretable Model Based on Pyramid Scene Parsing Features for Brain Tumor MRI Image Segmentation. *Computational and Mathematical Methods in Medicine*, 2022.[2]
17. Kajal, M., & Mittal, A. (2022). A modified U-Net based architecture for brain tumour segmentation on BRATS 2020
18. Das, S., Bose, S., Nayak, G. K., & Saxena, S. (2022). Deep learning-based ensemble model for brain tumor segmentation using multi-parametric MR scans. *Open Computer Science*, 12(1), 211-226.
19. Sun, H., Yang, S., Chen, L., Liao, P., Liu, X., Liu, Y., & Wang, N. (2023). Brain tumor image segmentation based on improved FPN. *BMC Medical Imaging*, 23(1), 172.
20. Sangui, S., Iqbal, T., Chandra, P. C., Ghosh, S. K., & Ghosh, A. (2023). 3D MRI Segmentation using U-Net Architecture for the detection of Brain Tumor. *Procedia Computer Science*, 218, 542-553.

APPENDIX

A.SOURCE CODE

```
import numpy as np

import nibabel as nib

import matplotlib.pyplot as plt

import keras

import keras.backend as K

from keras.callbacks import CSVLogger

import tensorflow as tf

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.keras.models import *

from tensorflow.keras.layers import *

from tensorflow.keras.optimizers import *

from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau, EarlyStopping, TensorBoard

from tensorflow.keras.layers.experimental import preprocessing

import cv2

import pandas as pd

IMG_SIZE=128

SEGMENT_CLASSES = {

    0 : 'NOT tumor',

    1 : 'NECROTIC/CORE',

    2 : 'EDEMA',

    3 : 'ENHANCING'}
```

```

VOLUME_SLICES = 128

VOLUME_START_AT = 22

TRAIN_DATASET_PATH = '/Users/sanjaydilli/Downloads/archive/
BraTS2021_Training_Data'

niimg = nl.image.load_img(TRAIN_DATASET_PATH + 'BraTS2021_00000/
BraTS2021_00000_flair.nii')

nimask = nl.image.load_img(TRAIN_DATASET_PATH +
'BraTS20_Training_001/BraTS20_Training_001_seg.nii')

fig, axes = plt.subplots(nrows=4, figsize=(30, 40))

nlplt.plot_anat(niimg,
                title='BraTS20_Training_001_flair.nii plot_anat',
                axes=axes[0])

nlplt.plot_epi(niimg,
               title='BraTS20_Training_001_flair.nii plot_epi',
               axes=axes[1])

dd

nlplt.plot_img(niimg,
               title='BraTS20_Training_001_flair.nii plot_img',
               axes=axes[2])

nlplt.plot_roi(nimask,
               title='BraTS20_Training_001_flair.nii with mask plot_roi',
               bg_img=niimg,
               axes=axes[3], cmap='Paired')

plt.show()

import os

train_and_val_directories = [f.path for f in os.scandir(TRAIN_DATASET_PATH)
if f.is_dir()]

```

```

def pathListIntolds(dirList):
    x = []
    for i in range(0,len(dirList)):
        x.append(dirList[i][dirList[i].rfind('/')+1:])
    return x

train_and_test_ids = pathListIntolds(train_and_val_directories);
train_test_ids, val_ids = train_test_split(train_and_test_ids,test_size=0.2)
train_ids, test_ids = train_test_split(train_test_ids,test_size=0.15)

def dice_coef(y_true, y_pred, smooth=1.0):
    class_num = 4
    for i in range(class_num):
        y_true_f = K.flatten(y_true[:,:,:,:i])
        y_pred_f = K.flatten(y_pred[:,:,:,:i])
        intersection = K.sum(y_true_f * y_pred_f)

        loss = ((2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f)
+ smooth))

        if i == 0:
            total_loss = loss
        else:
            total_loss = total_loss + loss

    total_loss = total_loss / class_num
    return total_loss

def dice_coef_necrotic(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:,:,:,:1] * y_pred[:,:,:,:1]))

    return (2. * intersection) / (K.sum(K.square(y_true[:,:,:,:1])) +
K.sum(K.square(y_pred[:,:,:,:1])) + epsilon)

```

```

def dice_coef_edema(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:,:,:2] * y_pred[:,:,:2]))
    return (2. * intersection) / (K.sum(K.square(y_true[:,:,:2])) +
K.sum(K.square(y_pred[:,:,:2])) + epsilon)

def dice_coef_enhancing(y_true, y_pred, epsilon=1e-6):
    intersection = K.sum(K.abs(y_true[:,:,:3] * y_pred[:,:,:3]))
    return (2. * intersection) / (K.sum(K.square(y_true[:,:,:3])) +
K.sum(K.square(y_pred[:,:,:3])) + epsilon)

def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def sensitivity(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    return true_positives / (possible_positives + K.epsilon())

def specificity(y_true, y_pred):
    true_negatives = K.sum(K.round(K.clip((1-y_true) * (1-y_pred), 0, 1)))
    possible_negatives = K.sum(K.round(K.clip(1-y_true, 0, 1)))
    return true_negatives / (possible_negatives + K.epsilon())

keras = tf.compat.v1.keras
Sequence = keras.utils.Sequence

class DataGenerator(Sequence):
    def __init__(self, list_IDs, dim=(IMG_SIZE,IMG_SIZE), batch_size = 1,
n_channels = 2, shuffle=True):
        self.dim = dim

```

```

self.batch_size = batch_size

self.list_IDs = list_IDs

self.n_channels = n_channels

self.shuffle = shuffle

self.on_epoch_end()

def __len__(self):
    return int(np.floor(len(self.list_IDs) / self.batch_size))

def __getitem__(self, index):
    indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]

    Batch_ids = [self.list_IDs[k] for k in indexes]

    X, y = self.__data_generation(Batch_ids)

    return X, y

def on_epoch_end(self):
    'Updates indexes after each epoch'

    self.indexes = np.arange(len(self.list_IDs))

    if self.shuffle == True:
        np.random.shuffle(self.indexes)

def __data_generation(self, Batch_ids):
    X = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim,
self.n_channels))

    y = np.zeros((self.batch_size*VOLUME_SLICES, 128, 128))

    Y = np.zeros((self.batch_size*VOLUME_SLICES, *self.dim, 4))

    for c, i in enumerate(Batch_ids):
        case_path = os.path.join(TRAIN_DATASET_PATH, i)

        data_path = os.path.join(case_path, f'{i}_flair.nii.gz');

        flair = nib.load(data_path).get_fdata()

```

```

data_path = os.path.join(case_path, f'{i}_t1ce.nii.gz');
ce = nib.load(data_path).get_fdata()

data_path = os.path.join(case_path, f'{i}_seg.nii.gz');
seg = nib.load(data_path).get_fdata()

for j in range(VOLUME_SLICES):
    X[j+(VOLUME_SLICES*c),:,:,0] = cv2.resize(flair[:,j+VOLUME_START_AT],
(IMG_SIZE, IMG_SIZE))
    X[j+(VOLUME_SLICES*c),:,:,1] = cv2.resize(ce[:,j+VOLUME_START_AT],
(IMG_SIZE, IMG_SIZE))
    y[j +VOLUME_SLICES*c,:,:) = cv2.resize(seg[:,j+VOLUME_START_AT],
(IMG_SIZE, IMG_SIZE))
    y[y==4] = 3;
    y = tf.one_hot(y, 4);
    return X/np.max(X), y

training_generator = DataGenerator(train_ids)
valid_generator = DataGenerator(val_ids)
test_generator = DataGenerator(test_ids)

csv_logger = CSVLogger('training.log', separator=',', append=False)

callbacks = [keras.callbacks.EarlyStopping(monitor='loss', min_delta=0,
patience=10, verbose=1, mode='auto'),
keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=10, min_lr=0.0001, verbose=1), csv_logger]

![[Screenshot%202024-03-15%20at%2011.08.56%E2%80%AFAM.png]
(attachment:Screenshot%202024-03-15%20at%2011.08.56%E2%80%AFAM.
png)]

def build_unet(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(inputs)

```

```

conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv1)

pool = MaxPooling2D(pool_size=(2, 2))(conv1)

conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(pool)

conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(conv)

pool1 = MaxPooling2D(pool_size=(2, 2))(conv)

conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool1)

conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv2)

pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool2)

conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv3)

pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(pool4)

conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(conv5)

drop5 = Dropout(dropout)(conv5)

up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer =
ker_init)(UpSampling2D(size = (2,2))(drop5))

merge7 = concatenate([conv3,up7], axis = 3)

conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge7)

```

```

conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(conv7)

up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer =
ker_init)(UpSampling2D(size = (2,2))(conv7))

merge8 = concatenate([conv2,up8], axis = 3)

conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge8)

conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv8)

up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer =
ker_init)(UpSampling2D(size = (2,2))(conv8))

merge9 = concatenate([conv,up9], axis = 3)

conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(merge9)

conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = ker_init)(conv9)

up = Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer =
ker_init)(UpSampling2D(size = (2,2))(conv9))

merge = concatenate([conv1,up], axis = 3)

conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(merge)

conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer
= ker_init)(conv)

conv10 = Conv2D(4, (1,1), activation = 'softmax')(conv)

return Model(inputs = inputs, outputs = conv10)

input_layer = Input((IMG_SIZE, IMG_SIZE, 2))

model = build_unet(input_layer, 'he_normal', 0.2)

model.compile(loss = "categorical_crossentropy",
optimizer=tf.keras.optimizers.legacy.Adam(learning_rate=0.001), metrics

```

```

['accuracy',tf.keras.metrics.MeanIoU(num_classes=4),
dice_coef, precision, sensitivity, specificity, dice_coef_necrotic,
dice_coef_edema ,dice_coef_enhancing] )

model.summary()

history = model.fit(training_generator,

                    epochs=30,

                    steps_per_epoch= 300,

                    callbacks= callbacks,

                    validation_data = valid_generator

                    )

model.save('UNET_2')

model = keras.models.load_model('/Users/sanjaydilli/Downloads/UNET.h5',
custom_objects={ 'accuracy' : tf.keras.metrics.MeanIoU(num_classes=4),

                  "dice_coef": dice_coef,

                  "precision": precision,

                  "sensitivity":sensitivity,

                  "specificity":specificity,

                  "dice_coef_necrotic": dice_coef_necrotic,

                  "dice_coef_edema": dice_coef_edema,

                  "dice_coef_enhancing": dice_coef_enhancing

                  }, compile=False)

model.compile(loss="categorical_crossentropy",
optimizer=keras.optimizers.Adam(learning_rate=0.001), metrics =
['accuracy',tf.keras.metrics.MeanIoU(num_classes=4), dice_coef, precision,
sensitivity, specificity, dice_coef_necrotic, dice_coef_edema,
dice_coef_enhancing] )

history = pd.read_csv('/Users/sanjaydilli/Documents/UNET/training.log',
sep=',', engine='python')

```

```

hist=history
acc=hist['accuracy']
val_acc=hist['val_accuracy']
epoch=range(len(acc))
loss=hist['loss']
val_loss=hist['val_loss']
train_dice=hist['dice_coef']
val_dice=hist['val_dice_coef']
f,ax=plt.subplots(1,3,figsize=(16,8))
ax[0].plot(epoch,acc,'b',label='Training Accuracy')
ax[0].plot(epoch,val_acc,'r',label='Validation Accuracy')
ax[0].legend()
ax[1].plot(epoch,loss,'b',label='Training Loss')
ax[1].plot(epoch,val_loss,'r',label='Validation Loss')
ax[1].legend()
ax[2].plot(epoch,train_dice,'b',label='Training dice coef')
ax[2].plot(epoch,val_dice,'r',label='Validation dice coef')
ax[2].legend()
plt.show()
print("Evaluate on test data")
results = model.evaluate(test_generator, callbacks= callbacks)
print("test loss, test acc:", results)
def predictByPath(case_path,case):
    files = next(os.walk(case_path))[2]
    X = np.empty((VOLUME_SLICES, IMG_SIZE, IMG_SIZE, 2))

```

```

vol_path = os.path.join(case_path, f'BraTS2021{case}_flair.nii.gz');
flair=nib.load(vol_path).get_fdata()
vol_path = os.path.join(case_path, f'BraTS2021{case}_t1ce.nii.gz');
ce=nib.load(vol_path).get_fdata()

for j in range(VOLUME_SLICES):

    X[j,:,:,:0] = cv2.resize(flair[:, :, j+VOLUME_START_AT],
(IMG_SIZE,IMG_SIZE))

    X[j,:,:,:1] = cv2.resize(ce[:, :, j+VOLUME_START_AT],
(IMG_SIZE,IMG_SIZE))

return model.predict(X/np.max(X), verbose=1)

def showPredictsByld(case, start_slice = 60):

    path = f"/Users/sanjaydilli/Downloads/archive/BraTS2021_Training_Data/
BraTS2021{case}"

    gt = nib.load(os.path.join(path, f'BraTS2021{case}_seg.nii.gz')).get_fdata()

    origImage = nib.load(os.path.join(path, f'BraTS2021{case}
_flair.nii.gz')).get_fdata()

    p = predictByPath(path,case)

    core = p[:, :, :, 1]

    edema= p[:, :, :, 2]

    enhancing = p[:, :, :, 3]

    plt.figure(figsize=(18, 50))

    f, axarr = plt.subplots(1,6, figsize = (18, 50))

    for i in range(6):

        axarr[i].imshow(cv2.resize(origImage[:, :, start_slice+VOLUME_START_AT],
(IMG_SIZE, IMG_SIZE)), cmap="gray", interpolation='none')

        axarr[0].imshow(cv2.resize(origImage[:, :, start_slice+VOLUME_START_AT],
(IMG_SIZE, IMG_SIZE)), cmap="gray")

```

```

axarr[0].title.set_text('Original image flair')

curr_gt=cv2.resize(gt[:, :, start_slice+VOLUME_START_AT], (IMG_SIZE,
IMG_SIZE), interpolation = cv2.INTER_NEAREST)

axarr[1].imshow(curr_gt, cmap="Reds", interpolation='none', alpha=0.3)
# ,alpha=0.3,cmap='Reds'

axarr[1].title.set_text('Ground truth')

axarr[2].imshow(p[start_slice, :, :, 1:4], cmap="Reds", interpolation='none',
alpha=0.3)

axarr[2].title.set_text('all classes')

axarr[3].imshow(edema[start_slice, :, :], cmap="OrRd", interpolation='none',
alpha=0.3)

axarr[3].title.set_text(f'{SEGMENT_CLASSES[1]} predicted')

axarr[4].imshow(core[start_slice, :, :], cmap="OrRd", interpolation='none',
alpha=0.3)

axarr[4].title.set_text(f'{SEGMENT_CLASSES[2]} predicted')

axarr[5].imshow(enhancing[start_slice, :, :], cmap="OrRd",
interpolation='none', alpha=0.3)

axarr[5].title.set_text(f'{SEGMENT_CLASSES[3]} predicted')

plt.show()

showPredictsById(case=test_ids[1][-6:])
showPredictsById(case=test_ids[2][-6:])
showPredictsById(case=test_ids[3][-6:])
showPredictsById(case=test_ids[4][-6:])
showPredictsById(case=test_ids[5][-6:])

```

```

def FPN(input_layer):

    conv1 = Conv2D(256,3, activation = 'relu', padding = 'same',strides=4)
(input_layer)

    conv2 = Conv2D(512, 3, activation = 'relu', padding = 'same',strides=8)
(conv1)

    conv3 = Conv2D(1024, 3, activation = 'relu', padding = 'same',strides=16)
(conv2)

    conv4 = Conv2D(2056, 3, activation = 'relu', padding = 'same',strides=32)
(conv3)

    conv5 = Conv2D(256,1, activation = 'relu', padding = 'same')(conv4)
    conv6 = Conv2D(256, 1, activation = 'relu', padding = 'same')(conv3)
    conv7 = Conv2D(256, 1, activation = 'relu', padding = 'same')(conv2)
    conv8 = Conv2D(256, 1, activation = 'relu', padding = 'same')(conv1)


    up1 = UpSampling2D(size = (2,2))(conv5)
    b1 = Add()(up1,conv6])
    up2 = UpSampling2D(size = (2,2))(b1)
    b2 = Add()(up2,conv7])
    up3 = UpSampling2D(size = (8,8))(b2)
    b3 = Add()(up3,conv8])


    c1 = Conv2D(128,3, activation = 'relu', padding = 'same')(conv5)
    c1 = Conv2D(128,3, activation = 'relu', padding = 'same')(c1)
    c1 = UpSampling2D(size = (128,128))(c1)


    c2 = Conv2D(128,3, activation = 'relu', padding = 'same')(b1)
    c2 = Conv2D(128,3, activation = 'relu', padding = 'same')(c2)
    c2 = UpSampling2D(size = (64,64))(c2)

```

```

c3 = Conv2D(128,3, activation = 'relu', padding = 'same')(b2)
c3 = Conv2D(128,3, activation = 'relu', padding = 'same')(c3)
c3 = UpSampling2D(size = (32,32))(c3)

c4 = Conv2D(128,3, activation = 'relu', padding = 'same')(b3)
c4 = Conv2D(128,3, activation = 'relu', padding = 'same')(c4)
c4 = UpSampling2D(size = (4,4))(c4)

c5 = tf.keras.layers.concatenate([c1,c2,c3,c4])
c6 = Conv2D(4, (1,1), activation = 'softmax')(c5)

return c6

input_layer = Input((IMG_SIZE, IMG_SIZE, 2))

```

B. SCREENSHOT

```
In [2]: import numpy as np
import nibabel as nib
import matplotlib.pyplot as plt

import keras
import keras.backend as K
from keras.callbacks import CSVLogger
import tensorflow as tf
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.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping, TensorBoard
from tensorflow.keras.layers.experimental import preprocessing
import cv2
```

Fig.B.1. Importing Required Libraries

```
In [5]: import os
train_and_val_directories = [f.path for f in os.scandir(TRAIN_DATASET_PATH) if f.is_dir()]

In [6]: def pathListIntoIds(dirList):
x = []
for i in range(0, len(dirList)):
x.append(dirList[i][dirList[i].rfind('/')+1:])
return x

train_and_test_ids = pathListIntoIds(train_and_val_directories);

train_test_ids, val_ids = train_test_split(train_and_test_ids, test_size=0.2)
train_ids, test_ids = train_test_split(train_test_ids, test_size=0.15)

In [7]: def dice_coef(y_true, y_pred, smooth=1.0):
class_num = 4
for i in range(class_num):
y_true_f = K.flatten(y_true[:, :, i])
y_pred_f = K.flatten(y_pred[:, :, i])
intersection = K.sum(y_true_f * y_pred_f)
loss = ((2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth))
if i == 0:
total_loss = loss
else:
total_loss = total_loss + loss
total_loss = total_loss / class_num
return total_loss

def dice_coef_necrotic(y_true, y_pred, epsilon=1e-6):
intersection = K.sum(K.abs(y_true[:, :, 1] * y_pred[:, :, 1]))
return (2. * intersection) / (K.sum(K.square(y_true[:, :, 1])) + K.sum(K.square(y_pred[:, :, 1])) + epsilon)

def dice_coef_edema(y_true, y_pred, epsilon=1e-6):
intersection = K.sum(K.abs(y_true[:, :, 2] * y_pred[:, :, 2]))
return (2. * intersection) / (K.sum(K.square(y_true[:, :, 2])) + K.sum(K.square(y_pred[:, :, 2])) + epsilon)

def dice_coef_enhancing(y_true, y_pred, epsilon=1e-6):
intersection = K.sum(K.abs(y_true[:, :, 3] * y_pred[:, :, 3]))
return (2. * intersection) / (K.sum(K.square(y_true[:, :, 3])) + K.sum(K.square(y_pred[:, :, 3])) + epsilon)

def precision(y_true, y_pred):
true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
precision = true_positives / (predicted_positives + K.epsilon())
return precision

def sensitivity(y_true, y_pred):
true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
return true_positives / (possible_positives + K.epsilon())
```

Fig.B.2. Splitting Dataset into Training and Testing Set

```

In [10]: def build_unet(inputs, ker_init, dropout):
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(inputs)
    conv1 = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv1)

    pool = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool)
    conv = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    pool1 = MaxPooling2D(pool_size=(2, 2))(conv)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv2)

    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv3)

    pool4 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(pool4)
    conv5 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv5)
    drop5 = Dropout(dropout)(conv5)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = (
    merge7 = concatenate([conv3, up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = (
    merge8 = concatenate([conv2, up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = (2
    merge9 = concatenate([conv, up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv9)

    up = Conv2D(32, 2, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(UpSampling2D(size = (2,
    merge = concatenate([conv1, up], axis = 3)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(merge)
    conv = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initializer = ker_init)(conv)

    conv10 = Conv2D(4, (1,1), activation = 'softmax')(conv)

    return Model(inputs = inputs, outputs = conv10)

input_layer = Input((IMG_SIZE, IMG_SIZE, 2))

```

```

In [11]: model = build_unet(input_layer, 'he_normal', 0.2)
model.compile(loss="categorical_crossentropy", optimizer=tf.keras.optimizers.legacy.Adam(learning_rate=0.001), metri
dice
dice
model.summary()

```

Fig.B.3. Building The Model

```

history = model.fit(training_generator,
                    epochs=30,
                    steps_per_epoch= 300,
                    callbacks= callbacks,
                    validation_data = valid_generator
                    )

model.save('UNET_2')

model = keras.models.load_model('/Users/sanjaydilli/Downloads/UNET.h5',
                                custom_objects={ 'accuracy': tf.keras.metrics.MeanIoU(num_classes=4),
                                                  "dice_coef": dice_coef,
                                                  "precision": precision,
                                                  "sensitivity": sensitivity,
                                                  "specificity": specificity,
                                                  "dice_coef_necrotic": dice_coef_necrotic,
                                                  "dice_coef_edema": dice_coef_edema,
                                                  "dice_coef_enhancing": dice_coef_enhancing
                                                  }, compile=False)

2024-01-27 22:45:09.117592: I metal_plugin/src/device/metal_device.cc:1154] Metal device set to: Apple M1 Pro
2024-01-27 22:45:09.117639: I metal_plugin/src/device/metal_device.cc:296] systemMemory: 16.00 GB
2024-01-27 22:45:09.117648: I metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 5.33 GB
2024-01-27 22:45:09.117710: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:303] Coul
d not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel may not have been built with NUMA suppo
rt.
2024-01-27 22:45:09.117741: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:269] Crea
ted TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical PluggableDevice (
device: 0, name: METAL, pci bus id: <undefined>)

```

Fig.B.4. Training and Loading The Model



Fig.B.5.Graph for Training Log of Model

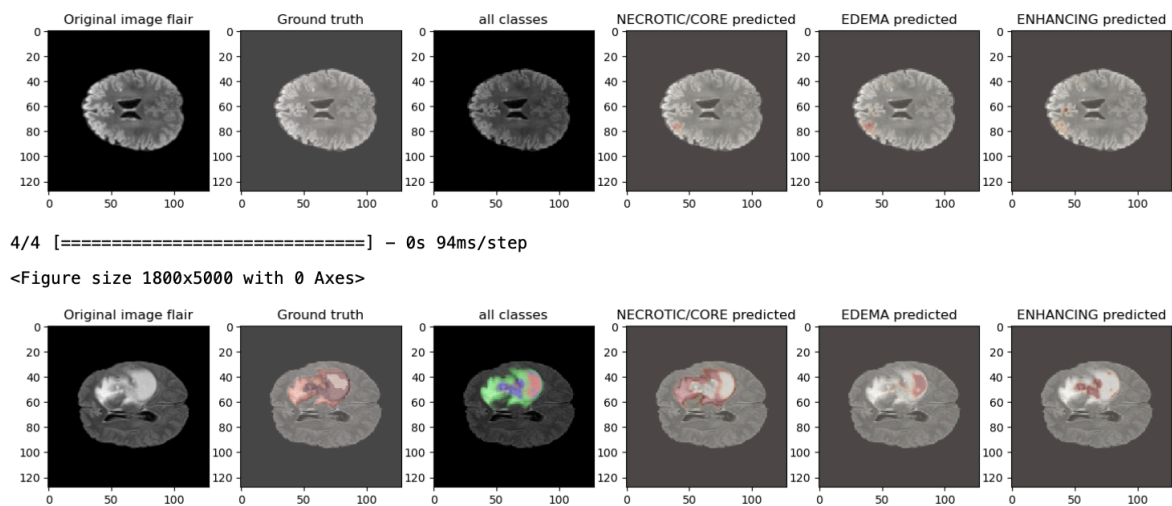


Fig.B.6.Prediction on Testing Images

C. RESEARCH PAPER

3D SEGMENTATION OF MRI USING MULTIMODAL FOR DETECTION OF BRAIN TUMOUR

Sanjay D
*Department of Computer Science
and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
sanjaydilli9@gmail.com*

J Cruz Antony
*Department of Computer Science
and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
jcruzantony@gmail.com*

V Devarsh
*Department of Computer Science
and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
vdevarsh2016@gmail.com*

R Vignesh
*Department of Computer Science
and Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
vignesh.cse@sathyabama.ac.in*

Murali E
*Department of Computer Science
and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
emurali88@gmail.com*

Abstract—Determining brain tumors from MRI can be challenging due to several factors, and it typically requires expertise in neuroimaging and collaboration between radiologists, neurologists, and other specialists. Some tumors may have a subtle or a typical appearance on MRI, making them difficult to distinguish from normal brain tissue. Brain tumors can be highly heterogeneous in terms of shape, size, and internal composition. Tumors can have different types of tissues, such as necrotic and enhancing regions, making it challenging to accurately delineate their boundaries. Brain tumors can be highly heterogeneous in terms of shape, size, and internal composition. Tumors can have different types of tissues, such as necrotic, cystic, or enhancing regions, making it challenging to accurately delineate their boundaries. Deep learning has shown promising results in various medical imaging tasks, including the identification and classification of brain tumors, such as gliomas, from MRI images. In this paper, an attempt to design an ensemble model for the prediction of 3D brain tumor segmentation and performance is going to be evaluated using deep learning models such as UNET, FPN, and UNET+FPN. The above models are evaluated on the BraTS 2021 (Brain Tumour Segmentation) dataset, and comparing the metrics of each model reveals that UNET and, an enhanced model that combines UNET and FPN, UNET+FPN has the highest accuracy of 0.9959 and 0.9954 respectively.

Keywords—UNET, FPN, Ensemble Model

I. INTRODUCTION

Brain tumor segmentation is a critical task in medical image analysis that plays a pivotal role in the diagnosis, treatment planning, and monitoring of patients with brain tumors. Accurate delineation of tumor boundaries from magnetic resonance imaging (MRI) scans is essential for assessing tumor size, location, and progression. Magnetic resonance imaging (MRI) features are as regarded one of the preferable approaches to identify [1] a brain tumor because it produces a more detailed and clear picture

than CT scans. Different contrast images generated by multimodal MRI procedures give complementary information that aids in segmenting the brain tumor and its surrounding tissues, which is crucial in the diagnosis and treatment of brain tumors. Deep learning approaches, especially Convolutional Neural Networks (CNNs), have demonstrated remarkable success in various image analysis tasks, including medical image segmentation. CNNs excel at learning hierarchical features from data, enabling them to capture intricate patterns and spatial relationships within images. In the context of brain tumor segmentation, deep learning models can learn to distinguish between healthy and pathological tissue, as well as different tumor subtypes. The goal of segmentation is to modify the way different parts of a picture are represented, making it simpler to comprehend sections of the image with distinct attributes. After dividing the brain image into thus many different pieces, each region becomes spatially continuous. Traditionally, this segmentation was done manually by trained professionals, a time-consuming and subjective process prone to errors. Brain tumors exhibit diverse morphological characteristics, and their precise segmentation from medical images is challenging due to variations in shape, size, and intensity. As a result, automated segmentation of brain MRI images can considerably improve diagnostic and treatment operations, especially when radiologists and competent professionals are few. However, precise tumor segmentation remains difficult due to glioma variability in terms of shape, size, and appearance, as well as the unclear and fuzzy boundary that exists between cancer and brain tissue [2]. The MRI data's intensity fluctuation adds to the difficulties. As a result, it is still subject to development, allowing for more research into improved segmentation approaches and accuracy. In this study, image segmentation models such as UNET, FPN, and UNET+FPN for confining and detecting the brain

tumor from the 3D MRI images. The main objectives of this study are:

- Automatic brain tumor segmentation method that uses informative image slices taken from 3D multimodal MRI volume to reduce computational time and increase segmentation accuracy
- Comparative analysis of image segmentation models based on the evaluation metrics for brain segmentation
- A hybrid model combined above mentioned models using an average ensemble technique and study of its performance on test data

II. RELATED WORKS

This paper attempts to design an automated AI-based brain tumor from an MRI scan. The study aims to develop a deep-learning model specifically for brain tumor segmentation. The authors review existing segmentation models and highlight the popularity of the 3D U-Net model. The proposed model is similar to the 3D U-Net architecture and consists of 6 encoding and decoding layers. It is trained on the BRATS dataset using dice loss and focal loss. The model achieves high accuracy, loss, and IOU scores. Comparative experiments show that the proposed model outperforms other models. Ablation studies are conducted to analyze the impact of different layers. Overall, the proposed model achieves state-of-the-art performance in brain tumor segmentation [5].

The paper focuses on semantic image segmentation, assigning class labels to pixels in an image for scene understanding. Existing methods are accurate but too slow for real-time applications like self-driving cars. The authors propose LinkNet, a CNN architecture that achieves real-time performance while maintaining accuracy. LinkNet uses an encoder-decoder structure, with the encoder based on pre-trained ResNet-18 and the decoder upsampling features to full resolution. Shortcut connections preserve spatial information. The authors trained LinkNet on CamVid and Cityscapes datasets, using cross-entropy loss, data augmentation, and class weighting[4].

The "BMC Medical Imaging" presents a proposed algorithm to improve the U-Net network and the improved strategy of FPN, as well as data enhancement technology, to segment the brain tumor region. Although more experiments may be needed to validate the precise effectiveness of this approach, the paper showcases an improvement in the segmented brain tumor region utilizing the proposed U-Net network with FPN and data

enhancement techniques [8]. To offer valid MRI scan results, deep learning approaches to merge deep neural networks with a CNN model. The CNN layout that was demonstrated was based on a three-layer fully connected neural network [12] [13]. The F-score was 97.33%, while the accuracy was 96.05% [14]. Developed a 3D CNN architecture for tumor extraction and employed transfer learning for classification, resulting in an accuracy of 98.32 on the BRATS 2015 dataset [15].

Based on pyramid scene parsing (PSP) for semantic segmentation. The proposed model uses convolutional neural networks (CNN) and pyramid pooling modules to extract spatial information from images. Overall, the paper mainly focuses on the development, evaluation, and visualization of the proposed PSPNet architecture for semantic segmentation [6]. The authors of reference [11] have solved a binary classification difficulty with concerning MRI imaging of brain malignancies. They used recurrent feature elimination (RFE) after extracting features using ALEXnet and VGG16. To fulfill the categorization problem, they eventually deployed a Support Vector Machine (SVM), which achieved an overall accuracy of 96%.[9]. The author employed superpixel algorithms for tumor segmentation and identification subsequent to transfer learning. The tumor was divided into two categories using the superpixel approach. The average dice index that resulted from this was 0.93 when compared to the ground truth data.

III. METHODOLOGY

The primary goal of this research is to create a brain tumor segmentation algorithm that can segment the four sub-tumor sections for all modalities by comparing current deep learning models and also an ensemble model created by the using average ensemble technique by combining the predictions of each model. The dataset used for this study is BraTS 2021, which consists of 1250 image files each image file comprises multimodal MRI scans, including T1-weighted, T2-weighted with contrast enhancement, T2-weighted, and fluid-attenuated inversion recovery (FLAIR) image, each image as the shape of (128,128,155) These different modalities provide complementary information about the brain and tumor tissue. The overall workflow of training models is mentioned in Fig.1 .

A. Data Preprocessing

The dataset is split into training, validation, and testing sets. The MRI images FLAIR, T1-weighted (T1ce) are used as input data(X) and segmented used for ground truth image (y). Each image is reshaped and sliced with an image size of 128. One-hot encoding on the segmentation labels. It converts the segmentation data into a one-hot encoded format with four classes.

B. Training

Each model was trained on 30 epochs using 300 training images and 250 validation images per

epoch. During model training, the callbacks function is utilized; the early callbacks function is used to verify validation accuracy with patiences of 5 in order to stop training if there is no progress in validation accuracy after 5 epochs. The categorical cross entropy loss is used as a loss function. Adam optimizer implemented, with a learning rate of 0.001.

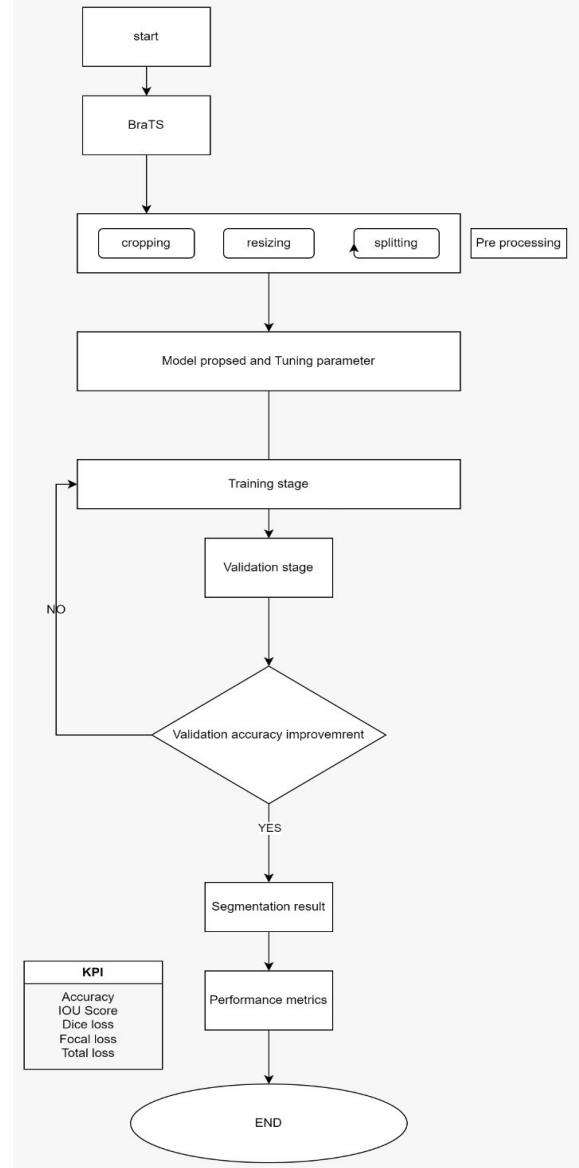
C. Evaluation Metrics

Accuracy, Mean Intersection Over Union, Sensitivity, Dice Coefficient, and Specificity were all used to assess the suggested model's performance [10]. These metrics may be articulated as TP, FP, TN, and FN, which are acronyms for the numbers of true positives, false positives, true negatives, and false negatives, respectively. These enhancement techniques [8].To offer valid MRI scan results, deep learning approaches to merge deep neural networks with a CNN model. The CNN layout that was demonstrated was based on a three-layer fully connected neural network [12] [13]. The F-score was 97.33%, while the accuracy was 96.05% [14]. Developed a 3D CNN architecture for tumor extraction and employed transfer learning for classification, resulting in an accuracy of 98.32 on the BRATS 2015 dataset [15].

Based on pyramid scene parsing (PSP) for semantic segmentation. The proposed model uses convolutional neural networks (CNN) and pyramid pooling modules to extract spatial information from images. Overall, the paper mainly focuses on the development, evaluation, and visualization of the proposed PSPNet architecture for semantic segmentation [6]. The authors of reference [11] have solved a binary classification difficulty with concerning MRI imaging of brain malignancies. They used recurrent feature elimination (RFE) after extracting features using ALEXnet and VGG16. To fulfill the categorization problem, they eventually deployed a Support Vector Machine (SVM), which achieved an overall accuracy of 96%. [9]. The author employed superpixel algorithms for tumor segmentation and identification subsequent to transfer learning. The tumor was divided into two categories using the superpixel approach. The average dice index that resulted from this was 0.93 when compared to the ground truth data.

The paper focuses on semantic image segmentation, assigning class labels to pixels in an image for scene understanding. Existing methods are accurate but too slow for real-time applications like self-driving cars. The authors propose LinkNet, a CNN architecture that achieves real-time performance while maintaining accuracy. LinkNet uses an encoder-decoder structure, with the encoder based on pre-trained ResNet-18 and the decoder upsampling features to full resolution. Shortcut connections preserve spatial information. The authors trained LinkNet on CamVid and Cityscapes

datasets, using cross-entropy loss, data augmentation, and class weighting[4]. metrics are used for evaluating image whereas other



are not compatible and are only used for binary classification. The metrics are defined as follows

a) Intersection Over Union is measured as the ratio of the overlap area to the entire area covered by both the predicted and ground truth areas. It quantifies how closely the anticipated region matches the ground truth region.

$$IOU = \frac{TP}{TP + FP + FN} \quad (1)$$

b) The dice coefficient is a computation of an overlap-based metric that measures the spatial overlap between the ground truth mask and forecast mask.

$$DC = \frac{2 * target \cap prediction}{target + prediction} \quad (2)$$

c) Specificity is the proportion of real negative values that were accurately detected.

$$Specificity = \frac{TN}{TN + FP} \quad (3)$$

d) Sensitivity is the proportion of real positive values that were accurately detected.

$$Sensitivity = \frac{TP}{TP + FN} \quad (4)$$

e) Accuracy is a probability statistic that evaluates the rate to which segmentation results suit the base true segmentation mask.

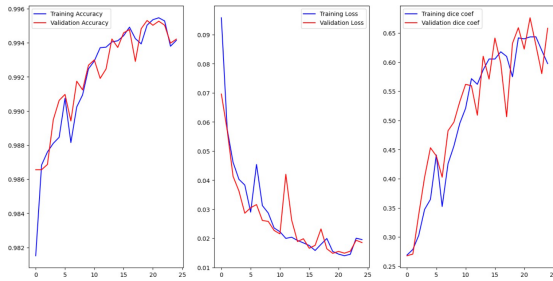
$$Accuracy = \frac{TN + TP}{TN + TP + FP + FN} \quad (5)$$

IV. RESULT

In this part, we will go over the experimental findings, network architecture for all of the models, model parameters, training details, and segmentation outcomes. In last, a comprehensive comparison of proposed deep learning models is done using several assessment measures.

A. UNET

Ronneberger et al. introduced UNET, an architecture for semantic segmentation that employs a Fully Convolution Network Model [3]. The contracting path and expanding path together make up the model contracting path has 5 encoders. The first encoder has two convolution layers with a kernel size of 3x3. The remaining encoders have max pooling with a kernel size of 2x2 and two convolution layers with a kernel size of 3x3. The expanding path has four encoders each encoder has Three convolution layers where one layer has a kernel sign of 2 and the remaining layer has a kernel size of 3. Each convolution has a Relu function and the kernel initializer. The middle layer of the Model has a dropout of 0.5. The output layer of the model has a convolution of 1x1 to divide each component feature vector into a relevant number of classes.



2. Graph for training log of UNET

On seeing the graph Fig. 2 It is evident that the model's accuracy improves with each epoch of training. Additionally, the validation accuracy

consistently surpasses the training accuracy. This suggests that the model is effectively generalizing to new data and is not overfitting to the training data. The loss graph certainly demonstrates that the model's loss decreases with each epoch of training. Furthermore, the validation loss consistently remains lower than the training loss. This indicates that the model is successfully minimizing the error between predicted and actual values and is not overfitting to the training data.

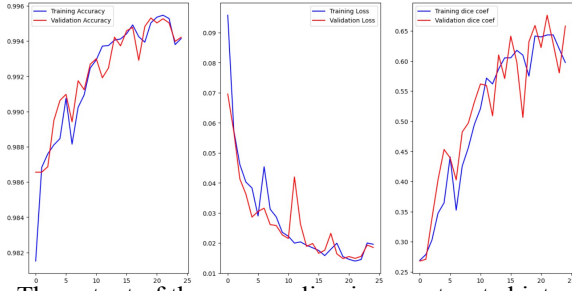
B. FPN

FPN addresses the need for precise localization and strong semantic features by constructing a pyramid of feature maps. To achieve this, FPN utilizes two main techniques were used Firstly, the top-down pathway involves upsampling high-resolution but semantically weak features from shallower layers and combining them with lower-resolution but semantically strong features from deeper layers. FPN employs lateral connections, which directly add high-resolution features from shallower layers to their corresponding counterparts in the pyramid. To construct the pyramid, FPN adds 1x1 convolutional layers on top of each high-level feature map from the CNN. These layers bring the features to the same resolution as the high-resolution feature maps by upsampling. The upsampled features are then directly added to their corresponding counterparts at each level of the pyramid through lateral connections.

In the top-down pathway, the upsampled features are element-wise summed with the corresponding lower-level features in the pyramid. This integration of information from different scales and resolutions enhances the ability of FPN to detect objects. The FPN architecture in the bottom-to-up path is composed of four convolutions. These convolutions have a respective number of filters [256, 512, 1024, 2056] and strides of [4, 8, 16, 32], with a kernel size of 3x3. On the other hand, the top-to-down path has four conditions, each with 256 filters and a kernel size of 1x1. Additionally, there are three samplings: up1, up2, and up3. Each sampling has 256 filters and kernel sizes of 2x2, 2x2, and 8x8 respectively. The up samplings up1, up2, and up3 are combined with their nearest neighbors in the bottom-to-up path using the Add() skip connection.

Furthermore, the model includes a decoder path consisting of four decoders. Each decoder has two convolutions and samplings. The convolutions have 128 filters and a kernel size of [128, 64, 32] to recover the spatial information from each Add() layer. suggests that the model is effectively generalizing to new data and is not overfitting to the training data. The loss graph certainly demonstrates that the model's loss decreases with each epoch of training. Furthermore, the validation loss consistently remains lower than the training loss.

minimizing the error between predicted and actual values and is not overfitting to the training data.

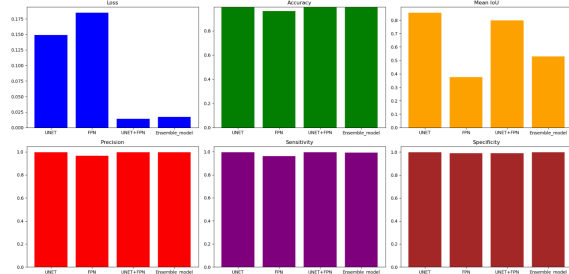


The output of the up-sampling is concatenated into a single layer. Then, a convolution with a softmax activation function is applied to generate the final output.

The graph Fig. 3 shows that training accuracy and validation Accuracy are increasing over time, indicating that the model is learning effectively and improving its prediction capabilities. Both training Loss and validation Loss decrease over time, indicating an improvement in model training. training dice coefficient and validation dice coefficient increase over time, indicating an improvement in the model's segmentation capability.

C. UNET+FPN

Upon comparing the architecture of FPN and U-Net, it becomes evident that they share similarities. The horizontal connection present in FPN can be achieved through the horizontal connection in U-Net. This feature enables the expansion of the U-Net model using the FPN structure. By leveraging the U-Net structure to its fullest potential, the U-Net model



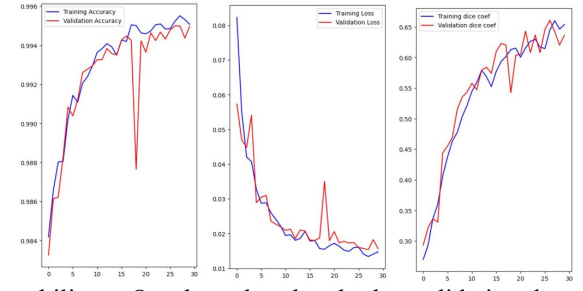
5. Comparison of Evaluation Metrics Of

effectively utilizes information from various scales. In the middle section of the encoder and decoder components of the UNET, we implemented a bottom-to-top pathway of the FPN. This pathway is responsible for restoring information from lower resolution to higher resolution. The FPN pathway consists of four upsamplings with a kernel size of (2,2) and four 1x1 convolutions with filter sizes of [32,64,128,256]. Each upsampling operation is accompanied by skip connections that connect the encoder and decoder of the UNET.

The graph Fig. 4 illustrates that, In the beginning, both accuracies rise rapidly, but after around epoch 10, they reach a plateau, suggesting that the model is

effectively learning but starting to stabilize its performance.

Initially, the training loss decreases and then



stabilizes. On the other hand, the validation loss experiences a spike before decreasing and stabilizing as well. This pattern indicates a potential case of overfitting at the beginning, where the model is overly fitted to the training data. On the other hand, the graph on the right side presents the dice coefficients for the training and validation datasets

throughout the epochs. Higher values in the graph indicate improved performance in terms of similarity between the predicted and actual results in tasks like image segmentation. As time progresses,

both coefficients gradually increase but with some fluctuations, suggesting that the model is continuously learning and making adjustments.

D. ENSEMBLE MODEL

In the end, developed an ensemble model consisting of five sub-models: UNET, FPN, and UNET+FPN. This ensemble model combines the outputs of these sub-models to generate a single output. The input layer of the ensemble model is set to (128,128,2). The average layer calculates the average of the outputs from the sub-models.

G. Comparison of Evaluation Metrics of Models

Fig. 5 shows that models UNET+FPN, and Ensemble model have the lowest loss, suggesting superior model performance. Every model has similar accuracy with slight differences. In comparison the models, UNET, UNET+FPN have the greatest Mean IoU score. Except for FPN with slight differences, all models in the research had the same accuracy and sensitivity score. Evaluation Metrics of models are mentioned in Table 1

I. Evaluation Metrics of Models

Metrics	UNET	FPN	UNET+FPN	ENSEMBLE MODEL
LOSS	0.149	0.0622	0.0144	0.0171
ACCURACY	0.9959	0.9929	0.9953	0.9945
MEAN IOU	0.8538	0.3750	0.7981	0.5308
PRECISION	0.9963	0.9948	0.9967	0.9961
SENSITIVITY	0.9955	0.9913	0.9942	0.9933

SPECIFICITY	0.9988	0.9983	0.9889	0.9987
-------------	--------	--------	--------	--------

5. Prediction of models on test image
UNET(a), FPN(b), UNET+FPN (c), Ensemble model(d)

I. CONCLUSION

The main objective of this study is to evaluate the effectiveness of various deep learning models for image segmentation, specifically for the rapid identification of brain tumors. The process of segmenting medical images with the assistance of radiologists can be extremely time-consuming and resource-intensive, making it impractical in remote areas. Therefore, the implementation of advanced automation techniques for brain tumor detection can greatly benefit a significant number of cases.

In order to address this issue, various models were utilized in this article, namely UNET, FPN, and UNET+FPN. Each model demonstrated different levels of accuracy during the data testing phase. Additionally, an ensemble model was created by averaging the predictions from each individual model. The model UNET, UNET+FPN exhibited the highest accuracy among all models, while the UNET+FPN has greater accuracy model followed closely behind. FPN and Ensemble model achieved stable performance

As a result, physicians are able to strategize treatment and monitor the progression of cancers during the diagnostic stage. The specific area of interest is divided into segments, and the presence of the tumor can be determined using an exceptionally precise model. This method offers the benefit of enhancing image segmentation and spatial it becomes evident that they share similarities. The horizontal connection present in FPN can be achieved through the horizontal connection in U-Net. This feature enables the expansion of the U-Net model using the FPN structure. By leveraging the U-Net structure to its fullest potential, the U-Net model effectively utilizes information from various scales. In the middle section of the encoder and decoder components of the UNET, we implemented a bottom-to-top pathway of the FPN. This pathway is responsible for restoring information from lower resolution to higher resolution. The FPN

automation techniques for brain tumor detection can greatly benefit a significant number of cases.

In order to address this issue, various models were utilized in this article, namely UNET, FPN, and UNET+FPN. Each model demonstrated different levels of accuracy during the data testing phase. Additionally, an ensemble model was created by averaging the predictions from each individual model. The model UNET, UNET+FPN exhibited the highest accuracy among all models, while the UNET+FPN has greater accuracy model followed closely behind. FPN and Ensemble model achieved stable performance

As a result, physicians are able to strategize treatment and monitor the progression of cancers during the diagnostic stage. The specific area of interest is divided into segments, and the presence of the tumor can be determined using an exceptionally precise model. This method offers the benefit of enhancing image segmentation and spatial localization to a greater extent, resulting in superior performance compared to previous systems. Furthermore, it requires less computational time and can be trained at a faster pace compared to networks with fewer parameters.

In the forthcoming studies, the emphasis will be on employing diverse classifier techniques to enhance precision while reducing error rates. Additionally, this approach could be tailored to predict the prognosis of individuals afflicted with brain tumors. In future research, the utilization of larger and more varied datasets may prove beneficial in conducting tests under real-world circumstances and clinical trials.

REFERENCE

1. S. Das, S. Bose, G. K. Nayak, S. C. Satapathy, and S. Saxena, "Brain tumor segmentation and overall survival period prediction in glioblastoma multiforme using radiomic features," *Concurrency Comput.: Pract. Experience*, p. e6501.
2. M. B. Naceur, R. Saouli, M. Akil and R. Kachouri, "Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in MRI images", *Comput. Methods Programs Biomed.*, vol. 166, pp. 39-49, Nov. 2018.
3. Ronneberger, O., Fischer, P., Brox, T (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention*. pp. 234-241 (2015).
4. A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 2017, pp. 1-4, doi: 10.1109/VCIP.2017.8305148
5. Kajal M., & Mittal, A. (2022). A modified U-Net based architecture for brain tumour segmentation on BRATS 2020
6. Zhao, M., Xin, J., Wang, Z., Wang, X., & Wang, Z. (2022). Interpretable Model Based on Pyramid Scene Parsing Features for Brain Tumor MRI Image Segmentation. *Computational and Mathematical Methods in Medicine*, 2022.
7. Amirreza S, Bansal S, Zhen L, Irfan E, Byron B. (2017). "One-Shot Learning for Semantic Segmentation". arXiv preprint arXiv:1709.03410.
8. Sun, H., Yang, S., Chen, L., Liao, P., Liu, X., Liu, Y., & Wang, N. (2023). Brain tumor image segmentation based on improved FPN. *BMC Medical Imaging*, 23(1), 172.
9. Mesut T, Zafer C, Burhan E, (2020). "Classification of Brain MRI Using Hyper Column Technique with Convolutional NeuralNetwork and Feature Selection Method". *Expert Systems with Applications*. 149
10. Bhushan C, Haldar JP, Choi S, Joshi AA, Shattuck DW, Leahy RM (2015). "Co-registration and distortion correction of diffusion and anatomical images based on inverse contrast normalization". *Neuroimage*. 115 pp:269-280.
11. Ahuja S, Panigrahi BK, Gandhi T (2020). "Transfer Learning Based Brain Tumour Detection and Segmentation

- using Superpixel Technique," 2020 International Conference on Contemporary Computing and Applications (IC3A), 2020, pp. 244-249
12. Choudhury CL, Mahanty C, Kumar R, Mishra BK, (2020). "Brain Tumour Detection and Classification Using Convolutional Neural Network and Deep Neural Network," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), pp. 1-4
 13. Ghosh SK, Biswas B, Ghosh A (2020). "Restoration of Mammograms by Using Deep Convolutional Denoising Auto-Encoders". In: Behera, H., Nayak, J., Naik, B., Pelusi, D. (eds) Computational Intelligence in Data Mining. Advances in Intelligent Systems and Computing, vol 990. Springer, Singapore. https://doi.org/10.1007/978-981-13-8676-3_38
 14. Rehman A, Khan MA, Saba T, Mehmood Z, Tariq U, Ayesha N (2021). "Microscopic brain tumor detection and classification using 3D CNN and feature selection architecture". Microsc Res Tech. 84: 133– 149.
 15. Tripathi P, Singh VK, Trivedi MC (2021). "Brain tumor segmentation in magnetic resonance imaging using OKM approach", Materials Today: Proceedings, Volume 37, Pages 1334-1340.

D.CERTIFICATE

**PRESIDENCY UNIVERSITY**
Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi | Approved By BCI
Bengaluru

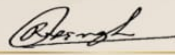
**2024 IEEE INTERNATIONAL CONFERENCE ON RECENT INNOVATION
IN SMART AND SUSTAINABLE TECHNOLOGY**

**IEEE**
March 15-16, 2024
Bengaluru, Karnataka, India

CERTIFICATE OF PARTICIPATION

This certificate is presented to V Devarsh for presenting the
research paper titled 3D Segmentation Of MRI Using Multimodal For Detection Of Brain Tumour
at the 2024 IEEE International Conference on Recent Innovation in Smart and Sustainable
Technology (ICRISST 2024), held at Presidency University, Bengaluru, Karnataka, India, on
March 15-16, 2024.


Dr. Joshi Manohar
HoD EEE
Publication Chair


Dr. Rajiv Ranjan Singh
HoD ECE
Technical Program Chair


Dr. Abdul Sharief
Dean - SOE
Patron ICRISST 2024