Segmentation Model for Brain Tumor Using MRI.

Student Name: Prem Vikas.

Student No: R00241672.

Supervisor: Dr Bruno Andrade.

For the module DATA9003 – Research Project as part of the Master of Science in Data Science and Analytics, Department of Mathematics, 25-08-2024.

Declaration of Authorship

I, Prem Vikas, declare that this thesis titled 'Segmentation Model for Brain Tumor Using MRI' and the work presented in it are my own. I confirm that,

- This work was done wholly or mainly while in candidature for the Masters' degree at Munster Technological University
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated
- Where I have consulted the published work of others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work
- I have acknowledged all main sources of help
- I understand that my project documentation may be stored in the library at MTU, and may be referenced by others in the future

Signed: Prem Vikas

Date: 25-08-2024

Table of Contents

Table of Figures

Abstract.

The precise categorization of brain tumors in MRI images is a crucial step in determining the nature of the disease and further therapy and prognosis in a patient. This thesis explores the different deep learning approaches that can be used for the segmentation of brain tumor, with particular emphasis on CNN and UNets architectures. First, a simple CNN was built to show that the model can learn features from the MRI data, though the segmentation was less accurate and poor at identifying the edges of tumors.

To overcome these limitations, UNet architecture is introduced that uses its encoder decoder network with skip connections for better accurate segmentation. The results showed that the proposed UNet model offered a much better performance than the basic CNN in terms of the Dice Coefficient and IoU. Additional improvements were then done by adding batch normalization in the UNet architecture to increase stability, decrease overfitting, and enhance the performance of the model on various datasets.

It also used K-means clustering, a conventional segmentation technique, as the benchmark for comparison. The outcomes highlighted the effectiveness of deep learning models, where the UNet with batch normalization yielded higher accuracy, precision, recall, and AUC than corresponding shallower models. However, the study also pointed out several opportunities for improvement based on the findings including enhancing the précised recall of the model to meet clinical level requirements.

For future work, one should consider complex architectures like Attention UNet, Transformers, and their combinations to enhance segmentation performance and robustness. Conclusively, this thesis has touched on the medical image analysis field and the future ability of deep learning models to transform the diagnosis and treatment of brain tumor patients and, in the process save lives.

1.Introduction.

In the rapidly advancing field of medical diagnostics, the early and precise detection of brain tumors remains a critical challenge that directly influences patient outcomes. Brain tumors, including benign and malignant ones, are among the most challenging diseases in the current medical practice because they affect the central nervous system. Due to the ability of early and accurate detection to influence treatment outcomes and predictors of survival, there is no argument that screening will ultimately improve health outcomes. Stepping into the practical implementation of Brain Imaging MRI has become the most preferred technique due to its non-invasive technique in generating high-resolution images of the brain's anatomy and pathology. However, the process of interpretation of these images, specifically, the separation of tumor from normal tissue, remains complex.

The challenges have been observed due to the high variability in the shape, size, and appearance of different brain tumors, intra-tumor heterogeneity, and background artifacts and noises in the MRI images. Manual contouring where radiologists draw around the tumor has been the most common method in the past. Although this approach holds great promise, it is not without its demerits. It is a time-consuming process requiring a huge amount of work and is influenced by inter-observer variability, which means that different radiologists will possibly obtain different segmentation results based on the experience and perception. Because of this, there is the aspect of variability which can be rated to create some level of ambiguity in diagnosing and could influence the treatment plan.

Deep learning, a specialized field of AI, has demonstrated a lot of potentials in confronting the issues of medical image analysis within the last few years. Convolutional neural networks are a type of deep neural networks that have proved very effective in image recognition. CNNs can learn features at multiple levels of abstraction directly from images and can be effectively used for tasks like tumor segmentation. Medical image segmentation has benefited from the UNet architecture since 2015, and has become the basis for many models.

In this thesis, an approach to brain tumour segmentation will be proposed with the aimed creation of more effective and efficient deep learning architectures that extend on the UNet's fundamental successes, but also present theoretical flaws. The novel idea of this work is based on enhancing UNet with cross-attention mechanisms and separable convolution layers. In the case of architectures of this type, these modifications are carried out in order to improve the model's ability to understand both the detailed features of input data and the general context in which they are used, as well as minimizing the computational burden that results from the utilization of standard CNN operations.

One of the mutual attention approaches that were developed on the basis of the BERT model come from the field of natural language and were recently introduced to computer vision tasks. Generally, in the medical image segmentation, cross attention helps the model to pay the attention on certain part of the image thus making it easy to differentiate between tumor and non-tumor areas. This is the case, especially in cases of brain tumour segmentation since the limits between the tumor and the healthy tissue are often blurred. This work is devoted to enhancing the precision and stability of the tumor segmentation performance across various medical datasets owing to the new modification of the UNet architecture, which is referred to as cross-attention.

Separable convolutions, another key component of the proposed model, suggest how to decrease the computational complexity of CNNs without a negative impact on the accuracy. In standard convolutional layers, every filter convolves through all the incoming channels thereby entailing a computation of many parameters. Separable convolutions decompose this operation into two smaller convolutions: a convolution where a single filter is applied to each input channel respectively and then a point wise convolution in which the output of the depth wise convolution is summed up. This decomposition minimizes the number of parameters of the model and the computational complexity, both to train and to implement it.

The implications of this research go beyond the managerial and technological aspects of the problem. Neural tumors are essentially the worst forms of cancer or tumors with the highest mortality rates and include primarily malignant gliomas. The glioblastomas, which are the most frequent and the most malignant of the primary intracranial tumors, have a median survival of 15 months despite optimal therapy. This is because delineation of the tumor, especially at the initial stages of diagnosis forms the basis of the later treatment planning procedures such as surgery, radiotherapy and chemotherapy. Geometric descriptions of the tumor are important for measuring tumor size and considering the strategies of resection and radiosurgery. Failure in segmentation results in wrong treatment regimens; this is a dangerous situation as it may lead to the patient's demise.

The urgency of improving brain tumor segmentation is underscored by the high incidence and mortality rates associated with these tumors. According to the World Health Organization, brain and central nervous system tumors account for approximately 2% of all cancers, but they are responsible for a disproportionate number of cancer-related deaths. This is due, in part, to the fact that many brain tumors are diagnosed at an advanced stage, when treatment options are limited and less effective. By advancing the state of the art in brain tumor segmentation, this research aims to improve both the accuracy of diagnoses and the effectiveness of treatments.

An important feature of this study is the inclusion of a rich set of data from an NCBI public database containing MRI brain images along with the corresponding mask images acts as a ground truth to train the neural networks. The selection of the dataset is critical, because it defines the quality and variety of the input data that are to be used in training of the models. Among the data features used in this research, the size, type, and location of tumors are diverse enough to cover all possible clinical situations and prevent the overfitting of the model.

Another methodology employed in this research combines cutting-edge techniques from machine learning and statistical analysis, with a focus on convolutional neural networks (CNNs) tailored for automatic tumor segmentation. The study is methodical where the authors propose a model, conduct design and implementation on the model and then conduct experiments for measuring the efficiency of the model. These experiments are aimed at comparing the performance of the suggested models with the existing ones in terms of accuracy measured by the Dice similarity coefficient, and precision and recall rates as well as time performance.

Besides proposing a segmentation model for brain tumor, this thesis also deals with another problem of applying AI in clinical context. Part of the challenge in AI adoption in healthcare is that a lot of AI models are complicated and hard to understand and therefore are not trusted. In response to this, the research also conducts an interpretability analysis to understand how the model arrives at certain conclusions and how its outputs are helpful to clinicians. This analysis is intended to increase confidence with modelling AI for medical image analysis to advance AI into its clinical use.

This research holds much prospective in contributing to knowledge. Thus, creating a closer and faster model to segment the brain tumor, this thesis helps in the AI’s advancing effort in providing healthcare benefits to society. The model designed in presented work could be applied not only to the segmentation of brain tumor but also to other diseases such as other types of cancer, cardiovascular diseases, neurological diseases, etc. In addition, the methods evolved in this study could be integrated with the other fields of AI, like natural language and autonomous systems, where precise calculations must be made quickly and effectively.

In undertaking this research, we are driven by a dual imperative: pushing the boundaries of what AI can achieve in medical imaging and addressing the practical needs of the medical community for faster, more accurate diagnostic tools. As such, this thesis not only contributes to the academic field of data science and analytics but also stands to make a tangible impact on medical practice, offering a glimpse into the future of AI-enabled healthcare.

1.1 Dataset.

For this project, a comprehensive dataset was sourced from a public repository, consisting of MRI brain images paired with corresponding mask images serving as ground truth for training neural networks. The dataset was chosen for its diversity and relevance to the research objective of developing a robust model for brain tumor segmentation.

The data collection process involved downloading and organizing the MRI images and their corresponding masks. Each MRI image was pre-processed to normalize the intensities, which facilitated more stable and efficient neural network training.

The pre-processed dataset was then split into training, validation, and test sets. The training set was used to train the model, the validation set was used to tune the model parameters, and the test set was used to evaluate the model's performance. This systematic approach ensured that the model could generalize well to unseen data, which is critical for clinical applications.

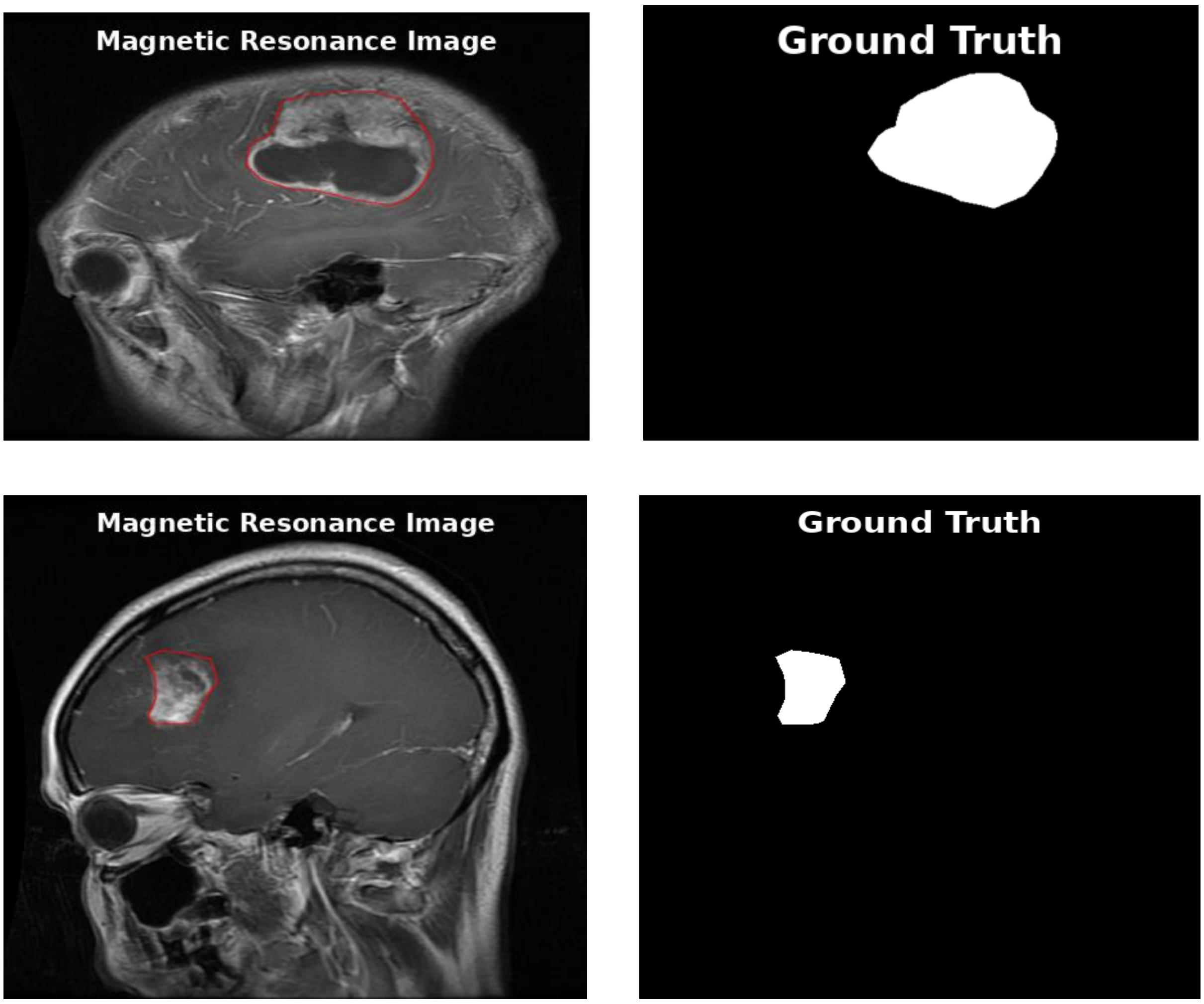


Figure 1. Examples of MRI Brain tumor images and its ground truth.

2.Literature Review.

Despite ongoing advancements, MRI-based brain tumor segmentation remains a crucial element in healthcare applications due to its significant role in diagnosis, treatment, and prognosis. Some of these disadvantages include, time consuming, prone to inter-radiologist variability diagnosis and treatments delivered may not be of consistent quality. Such variability is why automatic segmentation tools need to provide accurate and consistent segmentation results. Subsequently, the prevailing deep learning such as CNNs has enhanced the progress of the automated segmentation of brain tumors. This section gives an overview of existing methods on brain tumor segmentation namely CNNs, U-net, multi-modal data incorporation, and the real-world application of segmentation.

Early Approaches and the Role of CNNs

The use of CNNs was a turning point for medical image analysis, especially in areas including the segmentation of brain tumours. The authors Ghaffari et al. (2019) [1] developed a CNN-based method that particularly cope with some limitations such as class imbalance and overfitting. To address the difficulties of minority classes' segmentation, such as tumour regions, their model incorporated both: data augmentation and a weighted loss function. This branch was the first to show that CNNs was capable of handling some of the challenges of the medical imaging data and created a base on which more enhanced structures could be constructed.

Further research extended the present-day CNN based models subsequent to its initial implementation. For instance, Lin et al., 2021 [2] proposed a multi-path CNN model which could handle multi-modal MRI images with T1, T2, and FLAIR data. Through harmonizing the information obtained from different imaging modalities used in the study, their model enhanced the accuracy of segmentation by a big margin. These works emphasised the need to deal with inherent fluctuations in medical images; the methods ranged from extra data pre-processing and/or elaborative network design.

The paper entitled Optimizing CNN Architectures for Multi-Class Brain Tumor Segmentation, from 2021, describes ways to improve CNN architectures in view of the segmentation of different brain tumor types [15], like gliomas or metastases. Specifically, in such a context, it proposes a multi-class segmentation methodology based on tuning the CNN architecture to better distinguish different kinds of tumors by assuming that brain tumors are heterogeneous. It is the work that underlines the role-model complexity needs to play with computational efficiency, most especially in resource-constrained environments. The proposed model delivered competitive performance on benchmark datasets, proving its potential for broad clinical application.

Advancements in U-Net Architecture

U-Net architecture was defined for medical segmentation because of its great efficacy attributed to the encoder-decoder structure to capture both high and low-level features. The structure of the U-Net enables it to retain spatial information while gradually increasing the accuracy of anatomical structures and substructures of interest, and is well suited for applications such as brain tumor segmentation.

However, the standard U-Net architecture is not perfect, including when working with small tumors or areas where the tumor differs only slightly from the surrounding tissue. These problems were solved by Saleem et al. (2021) [3], who proposed an improved U-Net with attention-based modules. These mechanisms enable the model to adapt the attention on the more relevant areas of the image and enhance its performance in the cases of tumours with irregular shapes or low contrast.

In a similar pursuit to address the imbalance of classes, Pereira et al. (2016) **[**4] modified conventional U-Net architecture and trained it using a particular loss function that focused on the proper segmentation of small tumor regions. Wang et al. (2023) **[**5] proposed an improved 3D U-Net for the segmentation of brain tumors in rodents while improving computational complexity. This approach has shown that careful architectural design can minimize the need for computations while optimizing performance which is very important in real time clinical use.

Saleem et al. in their work of 2021 [13] proposed an improved version of U-Net called U-Net++ that includes visual attention for boosting the performance of brain tumor segmentation in MRI images. The U-Net++ model extends the conventional U-Net with nested and dense skip pathways that assist in efficient reuse of features and improved boundaries and structures of objects. The use of attention mechanisms is applied to enhance the attention to specific regions of the image when the outline of the tumor is irregular or has low contrast with other tissues. The results of this experiment showed better DSC that with the standard U-net, so it can be concluded that U-Net++ has potential for use in clinical situations where the accuracy of segmentation is important.

Multi-Modal MRI Integration

The use of multi-modal MRI data is paramount important in enhancing the performance of brain tumor segmentation. This issue is evident from the studies showing that different MRI sequences – T1w, T2w, and FLAIR – offer complementary information that is pivotal for accurate demarcation of tumors. Lin et al. (2021) [2] illustrated this using multi-path CNN that performs the segmentation of each MRI modality independently but generates a final segmentation map after merging the results of all. What they did was prove, through their work, that the integration of three or more would improve the model's performance in discerning between tumor and healthy tissue especially where the modality in question offered poor contrast.

Based on these multi-modal strategies, Saleem et al. (2021) [3] included multi-modal data into the improved U-Net model framework. Using different MRI sequences in parallel, the model of their work obtained higher segmentation metrics and better performance metrics for various patients. That is why the capability to combine and use data from various sources is expected to play an influential role in the clinical practice where variations in imaging protocols are even more typical.

Innovations in Model Efficiency

Although enhancing the model of segmentation can be a highly effective approach, it is equally important that deep learning models operate at a faster rate, especially when they will soon likely find applications in actual clinical settings. While Traditional CNNs and models based on U-Net have better performances, they often require large computational resources and thus are not feasible for use.

This challenge was discussed and solved by Wang et al. (2023) [5], where they proposed an optimized 3D U-Net model for segmenting brain tumors in rodents. Their work also proves that the architectural design can take sufficient precautions to lessen the computation complexity while achieving the same precision. Liu et al. (2022) [6] developed a model based on the integration of DT-CWT and DRL for the segmentation of brain tumors from multiple modalities. This approach helped eliminate the noise level and retain the important features of the image, and improve the number of segments while controlling for computational burden.

Another attempt to enhance the efficiency of the model was made by Fick et al. (2021) [7], who designed a cloud-based workflow for the real-time 3D visualization of brain tumors in the augmented reality environment. The inclusion of AR with brain tumor segmentation brings a new advancement in the application of these models and giving the surgeons a detailed view of tumor anatomy during the planning phase.

Incorporating Domain Knowledge: Physics-Informed Neural Networks

The application of prior information from the particular domain improves segmentation brain tumor segmentation models based on neural networks. In the same year Cao et al. (2022) [8] proposed another kind of Neural Networks namely Physics-Informed Neural Networks (PINNs) in which physical laws are imposed and anatomical constrains are incorporated into the learning process. This approach also makes sure that the segmentation results are not only based on the data but also on the physical properties of the brain tissue thereby leading to increased accuracy and more importantly easier interpretation.

The incorporation of the domain knowledge into deep learning is important for enhancing the physiological plausibility of the models and setting a new advancement in the area of deep learning by not using data-drive models. In this thesis, similar domain knowledge will be attempted to be included in the deep learning model in an attempt to increase the robustness and practical applicability of the model.

Segmentation in Augmented Reality and Clinical Applications

Among all the discussed fields of AI application, the combined use of artificial intelligence and augmented reality is the most promising approach in the context of brain tumor segmentation. Fick et al. (2021) [7] extended the development of a fully automatic segmentation algorithm which aims to support neurosurgical application in augmented reality for 3D visualization of brain tumors during pre-operative planning. Its cloud-based configuration enables the authors to develop 3D models from MRI data within a short span of time with real-time viewing with the help of AR Head-Mounted Displays.

Combined with AR, this means that the previously proposed brain tumor segmentation models are taken to a new level of practical application. They can facilitate surgeons to navigate through a tumor map and thus aid them in planning of the surgery, minimize the possibility of post-surgery complications and can also benefit the patient. The findings from Fick et al. 's work can be used to build on this thesis given its examination of the possibilities of similar integrations into clinical practice.

The Neurosurgical Focus (2021) paper [14] aims at examining the feasibility of incorporating fully automatic brain tumour segmentation algorithms into practice. The study points at efficiency as a key factor in automating the preoperative planning process and the precision in detecting the tumours. The described workflow incorporates deep learning models that are adapted for varying MRI modalities and performed rather well on various clinical datasets. The paper also presents the problems with the adaptation of such models into practice, like the problem of data unification and the problem of models' interpretation. This work is useful in the current developments towards deployment of AI techniques in clinical decision making, where the change must not cause a lot of distortion to medical practitioners.

Challenges and Future Directions in Clinical Integration

There is gradual progress in the development of AI-based brain tumor segmentation; however, translating the idea into practical application is not easy. Some of these challenges that Lin et al. (2021) [2] discussed were: The requirement of the model to be explainable and easy to use, the requirement of the model to be real-time. These are important for allowing the implementation of realised AI tools within existing clinical practice and for the utilisation of the tools by the healthcare professionals without significant relearning.

Similarly, Saleem et al. (2021) [3] also discussed that inter-observer variability should also be accounted for by using AI for the segmentation. In their work, they proved the effectiveness of AI models in improving the similarity of segmentation results among different viewers and, therefore, increasing the reliability of the findings. This discovery emphasizes the ability of AI in normalizing the process of brain tumor segmentation in clinical settings since interpractitioner variability contributes to residents' declines.

Enhancing Segmentation with Advanced Techniques

Few papers have proposed new methods to improve the performance of the brain tumor segmentation models even more. In 2022 [6], Liu et al., proposed a MGA using dual-tree complex wavelet transform and fuzzy logic for multimodal brain tumor segmentation. This approach of fusing signal processing methods and deep learning was capable of dealing with the fluctuations and noise present in the multimodal data hence proving that the integration of traditional signal processing methods and deep learning can enhance the segmentation results.

Likewise, [9] & [10] presented the deep learning challenges in the medical imaging the field in terms of overfitting of models, scarce dataset, and high cost of computations. These studies are informative when seeking to improve the proposed thesis model and include suggestions for wavelet transform, fuzzy logic, and attention mechanism.

Furthermore, Vijay et al. (2023) [11] and Banerjee & Mitra [12] proposed new techniques, namely, separable convolutions, and transfer learning that can help to enhance the segmentation efficiency, and accuracy. It is fundamental to mention that these techniques are of special interest for the goals of this thesis, since the established models aim to be accurate and fast to compute at the same time.

The 2023 paper integrates machine learning techniques with radiomics in brain tumor segmentation. Radiomics involves the extraction of a large number of features from medical images that can be used in improving the accuracy of segmentation models. The authors have proposed a framework in which the traditional machine learning algorithms are combined with radiomics features to come out with a hybrid model that outperforms conventional CNN approaches [16]. It is the case that radiomics contributes further context to the model, enhancing its performance in classifying between tumor and non-tumor tissues, particularly in complex cases where the standard imaging techniques often fail.

The authors of Multi-Modal MRI Segmentation Using Hybrid Deep Learning [17] have explored the applicability of hybrid deep learning models toward multi-modal MRI segmentation of brain tumors. In the proposed model, CNN is combined with recurrent neural networks to learn both spatial and temporal information from the MRI sequences. This hybrid approach can capture the temporal dynamics of tumor growth and treatment response that may not be handled by traditional CNNs. Such a hybrid model can significantly improve the accuracy of segmentation for longitudinal studies in which changes in tumor size and morphology need to be accounted for properly.

2.1 Metrics Review.

The review of the literature across the 17 papers reveals a diverse application of metrics to evaluate the performance of brain tumor segmentation models. The Dice Similarity Coefficient (DSC) is consistently used across most studies as a primary measure of overlap accuracy, reflecting the ability of models to correctly identify tumor regions. For instance, the DSC scores in the studies range from 53.7% in unsupervised networks [12] to over 90.4% in Physics-

Informed Neural Networks (PINNs) [5], highlighting the significant variations in model performance depending on the architecture and data used.

Other metrics like Intersection over Union (IoU) and Accuracy (ACC) are also frequently utilized to assess segmentation performance. Cao et al. [5] reported an IoU of 89.4%, indicating high overlap accuracy, while their model also achieved a remarkable overall accuracy of 95%. These metrics are essential in quantifying how well the segmented regions align with the ground truth annotations.

Sensitivity (SE) and Precision (PR) are crucial metrics in understanding a model's ability to correctly identify true positives (tumor regions) and avoid false positives, respectively. For example, Wang et al. [9] and Zhang et al. [10] achieved high sensitivity and precision values, emphasizing the models' effectiveness in distinguishing tumor tissues from healthy brain tissues.

Hausdorff Distance (HD) and Average Symmetric Surface Distance (ASSD) provide insights into the geometric accuracy of segmentation, particularly focusing on the boundaries of the segmented regions. Fick et al. [7] utilized these metrics to evaluate the boundary accuracy of their cloud-based segmentation model, reporting an HD of 4.80 and an ASSD of 1.31. These metrics are particularly relevant in surgical planning, where precise boundary delineation is critical.

Additional metrics such as the Information Loss Index and Boundary Error, used by Liu et al. [6] and Vijay et al. [17], provide further layers of evaluation by assessing the information retention and boundary accuracy, respectively. These measures underscore the importance of developing models that not only perform well in terms of overlap but also maintain high fidelity in the representation of tumor boundaries.

Main Author	Year	Network	DSC (%)	IoU	ACC	SE	PR	HD	ASSD	PPV	Information Loss Index
Ghaffari et al.	2019	CNN	86	-	-	-	-	-	-	-	-
Lin et al.	2021	Multi-path CNN	88	-	-	-	-	-	-	-	-
Saleem et al.	2021	U-Net + Attention	89.5	-	-	-	-	-	-	-	-
Pereira et al.	2016	Modified U-Net	87.1	-	-	-	-	-	-	-	-
Cao et al.	2022	PINNs	90.4	89.4	95	93.7	-	-	-	-	-
Fick et al.	2021	Cloud-based Segmentation	87	-	-	-	-	4.80	1.31	-	-
Liu et al.	2022	Hybrid (DTCWT + DRL)	85	-	-	-	-	-	-	-	0.18 (Info Loss), 0.3
Guan et al.	2022	BraTS 2020	85 (WT)	-	-	83	99	8.44	-	-	-
Wang et al.	2021	DFP-ResUNet	84	-	-	89	-	5.23	-	99	-
Zhang et al.	2021	ME-Net	70 (ET)	74 (TC)	88 (WT)	79	99	38.6	-	-	-
Baur et al.	2020	Unsupervised Network	53.7	-	-	-	-	-	-	-	-
Di Ieva et al.	2021	DL CNN	87.8	-	-	-	-	-	-	-	-

Xie et al.	2021	CNN with Transformers	85	-	-	-	-	-	-	-	-
Gryska et al.	2022	Dual-path CNN	77	-	-	-	-	72	-	-	-
Hsu et al.	2022	SegResnet	87.3	-	-	-	-	7.99	-	-	-
Mascarnehas et al.	2020	Image Processing	-	-	-	-	-	-	-	-	-
Banerjee & Mitra	2020	CNN 2D	90.2	-	99.3	91.4	4.75	-	-	-	-
Vijay et al.	2023	3D U-Net + SPP and Attention	88.3	-	-	-	-	7.99	-	-	Hausdorff Distance (WT 16.06, TC 5.65, ET 5.27)
Saleem et al.	2021	U-Net++ with Attention	85.7	-	-	88.3	-	5.32	-	-	Dice Similarity (WT 85.7, TC 79.8, ET 69.9)

Figure 2. Metrics evaluation of literature reviews.

3.Methodology.

3.1 Data Pre-processing.

From a reliable public website, 3064 MRI Brain images with matching ground truth masks were gathered to from a large dataset. The dataset was considered to be as diverse as possible to consider for creating the model to segment the brain tumors from other tissues. The pipeline for loading and pre-processing the data was set up as intensively as possible for correct pairing of images and masks, subsequent equal pre-processing of both data types, as well as in satisfaction of their further division into training, validation, and test subsets.

The first preparatory step was to take and sort out the MRI images and the corresponding masks which were to be used in the whole process. In order to ensure that no data was lost, and all images were matched with the correct mask, both the images and masks were sorted and a check was carried out to ascertain that if there was an image, there was a mask for it. To do this, all masks that did not correspond to an image—that is, all masks whose label was not found in the list of labels—were discarded. This was important to avoid some protocol violations during the training phase, and to ensure that the datasets that were used were accurate.

Upon proper split of the images and masks we augmented the data and split them into training, validation and test sets. This partitioned the data into affine and validation at a specified split ratio applied to the data, for instance 20% into validation. The combined training set was then further divided to make the independent test set. Such splitting of the dataset was established systematically so as to allow development of the model to train while performing a proper validation and testing on unseen data. This approach is rather important for indicating overfitting and for ensuring that the model has a generalizing potential, which can be a rather crucial factor in terms of clinical usage of the models.

Afterward, all the images of the MRI and the corresponding masks were preprocessed separately after dividing the dataset into train, validation, and test sets as previously mentioned. The images were preprocessed by reading and resizing them to an appropriate size of 256 x 256 pixels so as to enhance the consistency in the dataset. The pixel values of the images were then scaled to be of range 0 to 1 This makes it easier to train a neural network and encourages convergence through smaller gradients. They were also resized and normalized, which were masks: binary images of the presence of a tumor or its absence. Due to compatibility with the neural network's expectations to be trained with images of shape (batch_size, height, width, channels) all masks were converted to grayscale and then reshaped to include the channel dimension to match the rest of the data.

The pre-processed images and masks were then passed through a TensorFlow data pipeline which took care of batching and loading data in the most efficient manner possible during the course of training. The following pipeline utilized TensorFlow's data management functions to map all the pre-processing steps across the dataset in which the images, masks, and checkpoints were parsed and formatted correctly for the training. Data was split into batches, pre-loaded to help with the loading time and randomly permuted to reduce dependence of the model on a particular order of the data. Moreover, the performance of data pipeline was configured to increase or decrease depending on the system resources by using TensorFlow's AUTOTUNE.

Lastly, the key hyperparameters identified during training, particularly for this type of problem and utilizing the previously mentioned layers, included a batch size of 16 was chosen, as was a learning rate of 1e-4, and a training for 20 epochs. All of these parameters were selected to ensure that the training time is reasonable while the accuracy and ability to generalize from the model is still quite good. The model and training logs were stored in paths, model is saved with. hdf5 format, which is a specific Keras format of saving Keras models.

This comprehensive approach to data loading and pre-processing ensured that the dataset was well-prepared for training a neural network model for brain tumor segmentation. By carefully managing and pre-processing the data, the project aimed to develop a model that is both accurate and robust, capable of being deployed in clinical settings where reliability and precision are paramount.

3.2 Methods/Data Modelling.

The approach used in this thesis is based on the use of deep learning methods to segment brain tumours in MRI images. Due to the nature of MSI and the difficulties that characterize the process of segmenting a tumor, several models and methods were proposed. The models that are studied are a simple Convolution Neural Network (CNN), the U-net and an improved model of U-net with Batch Normalization. Furthermore, the K-means clustering, one of the most traditional methods of segmentation, was also used for the comparison of the effectiveness of using deep learning. All these models were trained and tested on a large and diverse dataset of MRI images and the ground truth masks based on it with much data pre-processing was done to provide the data most suitable for training. The following sections give a detailed description and analysis of every model, their structure, possibilities of the training processes, and the obtained outcomes.

3.2.1 Model 1: Convolutional Neural Network (CNN) for Brain Tumor Segmentation

To explore the correct classification of brain tumor MRI images, the basic model applied was a Convolutional Neural Network (CNN). CNNs have attracted a lot of interest as well as achieved high improvements in image-related undertakings because they are able to efficaciously and spontaneously extract extensive cascaded features from primal image data. Their effectiveness in applications like image classification, object detection and particularly medical image segmentation qualifies CNNs for the challenging task of detection and segmentation of brain tumours from MRI images.

Model Architecture:

The structure of the CNN model applied in the present study has been developed to classify pixel as either tumor or non-tumor in the MRIs. The architecture is sequential in as much as they are built, layer by layer in such a way that one layer learns from the next as features from the images are gradually extracted.

The input to the CNN model is preprocessed MRI images of size 128 x 128 pixel and in the form of grayscale image represented as a single channel image with the shape of 128 x 128 x 1. However, it should be noted that, while MRI images are in general grayscaled, the input was adapted to the structure used in CNN kind architectures in which the third dimension corresponds to the number of channels.

The input layer plays a fundamental role of creating the foundation through which the network is going to take in the image data as well as its size and architecture. It ensures that the kind of data input into the network is corrected and normalized prior to convolution by other layers.

Convolutional Layers:

First Convolutional Block: The model starts with a convolution layer with 32 filters, each of them 3 x 3 in size, and the activation Rectified Linear Unit. Padding='same' is used for spatial dimensions, so that the resultant feature map is the same size as the input image. After the convolution operation, the feature map goes through a max-pooling layer of 2x2 which means they reduce the field of the map by one half.

The first convolutional block is designed to find small details in the form of edges, textures and corners. These basic features are important so that the subsequent layers, can be built on top of them, and so the model can gradually learn complex features of the data.

The second block employs a similar architecture to the first block but with 64 filters and with a kernel size of 3×3 with ReLU activation. This is accomplished by the pooling operation which down-samples the feature map even further. By increasing the filter depth there is a possibility of more complex features being captured by the network while at the same time, the dimensions are being reduced thus dealing with computational issues.

This block allows the model to learn more complex and high-level features required to identify tumor regions Because the tumor regions may be irregular in shape, size, variation in intensity, this block improves the depth of the network.

The third layer of the convolutional block adds more filters totalling to 128, the kernel size is still 3x3 and uses ReLU activation followed by max-pool. This increase in the number of filters enables the network to learn even higher order features making for a more solid ground of decision making at the later stages.

The third block is intended to select even more detailed characteristics, which will help to make the differentiation of tumour and non-tumour tissue more accurate even in unclear and obscure conditions.

Bottleneck Layer.

The bottleneck layer is the fourth layer of the CNN architecture with the 256 filters and a kernel of 3 x 3 and ReLU activation. In this layer, unlike the previous ones, no pooling is carried out implying that the spatial input/output size is preserved to the later stages, though the feature maps depth is extended to its fullest. This layer is the one that contains the most summarized information about the input data, the features that the model finds the most important and the most general level of abstraction.

The bottleneck layer can be considered as the most important one, as it entails all the necessary information from the input image and condenses it in a feature map which is dense. This is the central platform of the model at which the greatest number of features is gathered prior to up sampling.

Upsampling Blocks

The upsampling process is an important part of the architectural design where the feature maps are resized from the feature space to the input space. This is done by three Conv2DTranspose layers – the layers that are the direct opposite of Conv2D layers because they “expand” the feature maps up to twice on every subsequent layer. The first layer of upsampling contains 128 filters, 64 filters at the next level, and only 32 filters at the final level, each with a 3x3 kernel and ReLU activation.

Upsampling helps the model get back the pixel resolution lost while the convolution and pooling were being done, and still retain the learnt features. This process is very important in the generation of segmentation mask in the correct size of the input image hence allowing precise identification of the tumor regions.

Output Layer.

The last layer in the CNN architecture is thus a convolutional layer with only one filter of size 1 x 1, but with Sigmoid activation function. This layer provides a map with the probability density map of the pixels being of the tumor class. The sigmoid function is most appropriate for binary classification problems for the simple reason that their output is a probability map which can be easily thresholded to obtain binary segmentation masks.

The output layer finally maps the learn features to binary labels to give the final segmentations map corresponding to each pixel. These masks can be then compared with the ground truth masks to judge the performance of this map.

Model Compilation.

The CNN model was compiled using the Adam optimizer which is a very commonly used optimizers due to their attributes like adaptive learning rates for sparse gradients. The loss function used was the binary cross entropy, suitable for binary classification tasks such as tumor segmentation when the classifier has to differentiate between the tumor class and the rest of the class which is the non-tumor class. For model evaluation, the accuracy was used which simply reports the rate of correctly classified pixels during training.

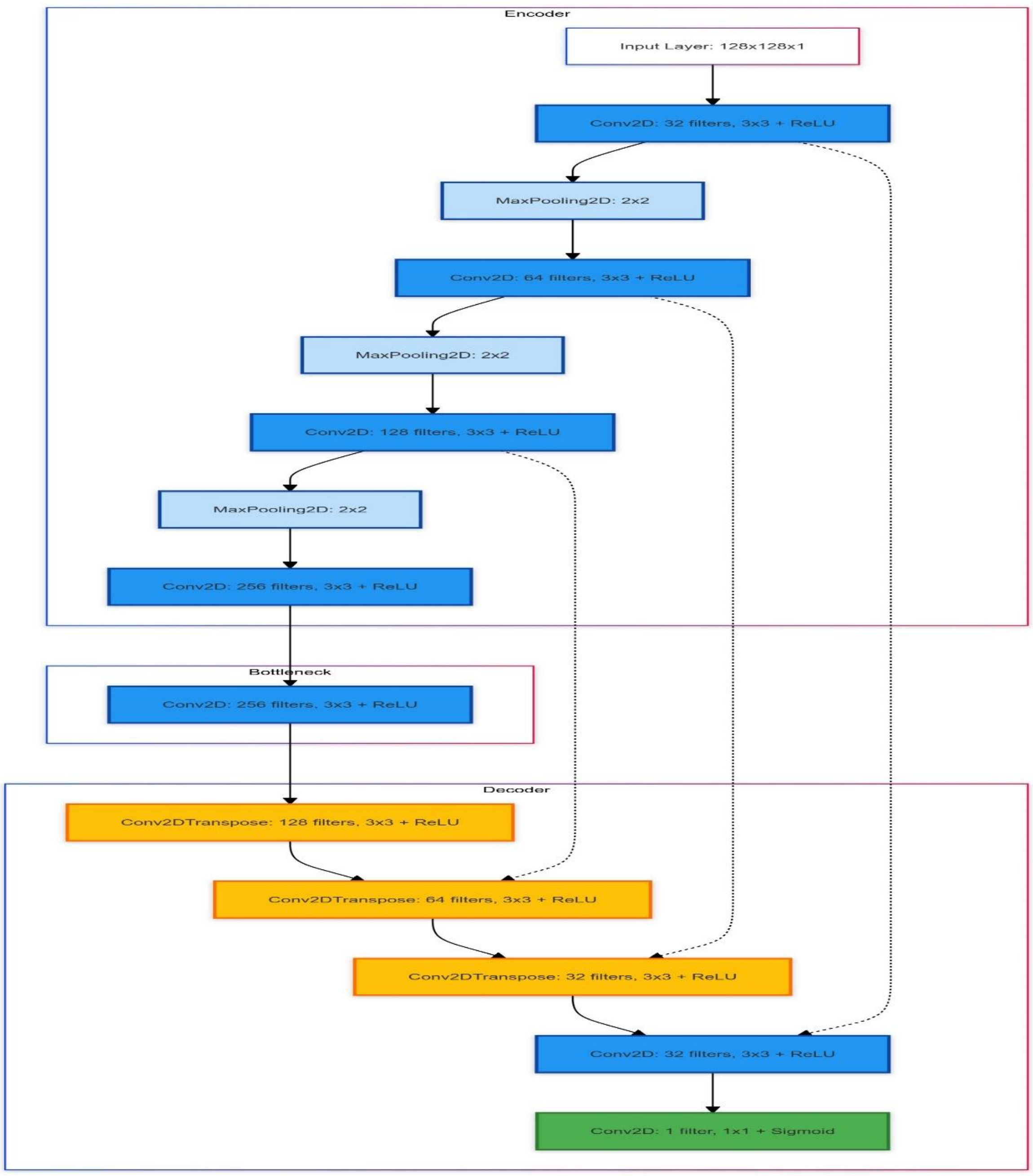

Encoder
Input Layer: 128x128x1
Conv2D: 32 filters, 3x3 + ReLU
MaxPooling2D: 2x2
Conv2D: 64 filters, 3x3 + ReLU
MaxPooling2D: 2x2
Conv2D: 128 filters, 3x3 + ReLU
MaxPooling2D: 2x2
Conv2D: 256 filters, 3x3 + ReLU
Bottleneck
Conv2D: 256 filters, 3x3 + ReLU
Decoder
Conv2DTranspose: 128 filters, 3x3 + ReLU
Conv2DTranspose: 64 filters, 3x3 + ReLU
Conv2DTranspose: 32 filters, 3x3 + ReLU
Conv2D: 32 filters, 3x3 + ReLU
Conv2D: 1 filter, 1x1 + Sigmoid

Figure 3. Operational Framework of the CNN model.

Training Process.

The CNN model was trained from the preprocessed set of MRIs and its segmentation masks. The training of the proposed model was done over 10 epochs and the batch size used was 16. Utmost 10% of the training sets were reserved for validation in an epoch to verify the ability of the model to perform well. This approach also aided in checking for overfitting and make sure that the model does not do very well within the training sample set, but poorly within sample set which was not used for training. The training history was logged so that its progress could be monitored and it could be seen what happened at certain intervals or epochs, including learning and loss curves.

Model Summary:

In the model summary, it is possible to know the number of layers of the model, the size of the output of each layer, and the total number of parameters. This summary is crucial in ensuring that the structure of model is correct and the number of layers in a particular layer is indeed compatible with the input data. It also provides information about the dimensionality of the model; the total number of trainable parameters gives information on the ability of the model to learn from the data.

The proposed CNN model served as reference model for the task of brain tumor segmentation that was able to highlight the main advantages and drawbacks of CNNs. As with the basic CNN architecture, the ability to learn and segment tumor regions was proven to be effective, but issues such as class imbalance and a lack of spatial context guided the search for a more complex architecture. These led to the subsequent study of the U-Net model that has skip connections and superior architecture to overcome these dilemmas and enhance segmentation efficiency.

3.2.2 Model 2: K-means Segmentation.

K-means clustering is an important unsupervised [13] learning algorithm especially widely applied in the image segmentation. K-means segmentation aims at dividing an image into various sections, or segments, where the pixels in that segment are most similar. Here, there are K segments in a prescribed number of pixel images and each pixel of an image is referred to a segment. The algorithm continues to fine tune such clusters in such a way that the variance of the group will be given precedence, thus helping to classify similar pixels while giving differences between the regions within an image. The following sub-sections describe the procedure used for the implementation of K-means segmentation approach in this research.

The first of the segmentation process in K-means entails pre-processing of the image data that is to be clustered. The image is reconstructed to the format that K-means algorithm takes by considering each pixel as a feature vector of an N-dimensional space.

Image Reshaping: The input image, often specified as a two-dimensional matrix of pixel grayscale intensities, is then reduced into one dimension, so that it is a vector of pixel levels. In the case of grayscale images, the pixel values are given in the form of just one channel (intensity values). For the RGB images, each pixel consists of the three dimensions for the red-, green and blue channels.

Normalization: real pixel values are pre-processed by scale transformation in the range [0, 1] to constrain the pixel values making sure all features (for instance the color channels) provide the same contribution for the distance metrics in the K-means algorithm.

Initialization of Centroids: The method of K-means starts by creating K centroids at any random position in the high-dimensional space defined by the features being used. These centroids are the first set of clusters centers. The selection of K is another vital factor that can be computed by using methods such as the Elbow Method or can also be recognized using prior knowledge of that particular field.

Pixel Assignment to Clusters: It works in such a way that each pixel in the image is classified into the cluster closest to it, where the distance computation method used is usually done by square root of the summation of the square of the difference between the pixel intensity values of the pixel and those of the centroid of the clusters, which is the squared Euclidean distance. This involves the computation of Euclidean distance between each pixel and all the K centroids before classification of the pixel to that cluster which is represented by the nearest centroid. The outcome of this step is a first partition where to every pixel is linked to a certain cluster.

Updating Centroids: After all the pixels have been classified to belonging to a particular cluster, the latter’s centroids are re-calculated. The new centroid was computed in the co-ordinate system as the mean of the I, j co-ordinates of all the pixels belonging to the given cluster. It is useful to shift the centroid to the center of the cluster which it represents and the pixel value of this cluster is a valuable average.

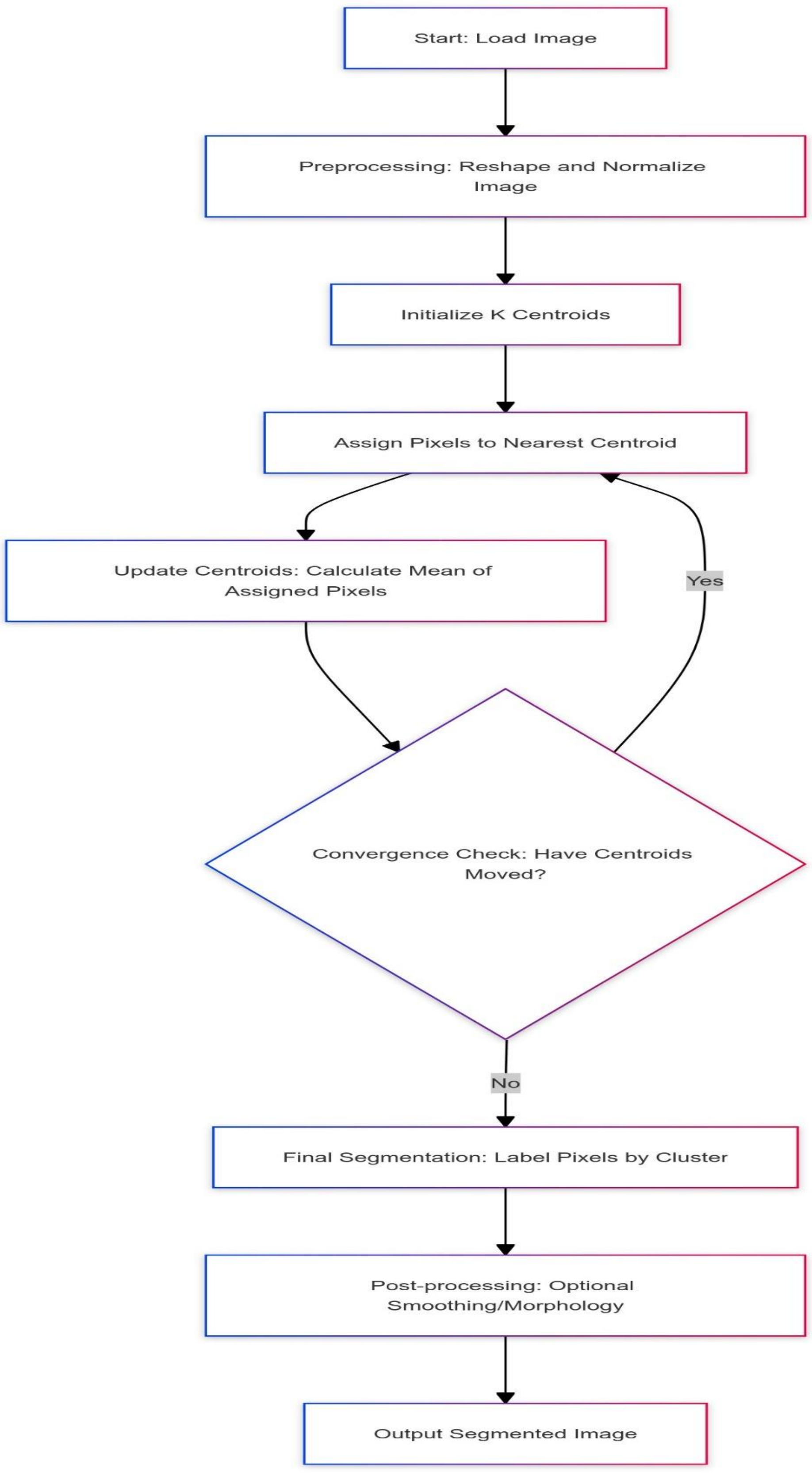

Start: Load Image
Preprocessing: Reshape and Normalize Image
Initialize K Centroids
Assign Pixels to Nearest Centroid
Update Centroids: Calculate Mean of Assigned Pixels
Yes
Convergence Check: Have Centroids Moved?
No
Final Segmentation: Label Pixels by Cluster
Post-processing: Optional Smoothing/Morphology
Output Segmented Image

Figure 4. Operational Framework of K-means segmentation.

Iterative Optimization: K-means segmentation is an algorithm that require iteration for its application. The pixel assignment and centroid update steps are again continued till the centroids become fixed; which means that it does not undergo a huge shift from one iteration to the next iteration. This convergence criterion makes sure that the algorithm has found the best solution in their own least square sense on the intra-cluster variance.

Convergence Check: The specific alteration of centroids goes on until the shift in the position of centroids is less than a certain threshold or until the fixed maximum number of iterations elapses. This is where the clustering process is thought to have ended and the last segmentation of the groups is made.

Final Segmentation: What is obtained in the form of the result of the K-means algorithm is a segmented image in which each pixel is assigned the number of its cluster. Using this segmented image more areas are pointed out within the initial picture, to one of the K clusters each belonging. The resulting segmented image is displayed as a combination of different colors for each of the clusters, or intensity levels in case of a grayscale image.

Post-processing.

There exists a form of subsequent image processing might be done on the segmented image to build up the quality of the segmentation. These techniques can include:

Smoothing: Adding filters to remove much noise and also ensure that their transition from one segment to the other is smooth.

Morphological Operations: Methods, for example, promotion of the edges of the segmentation or erosion in order to optimize the quality of the outlined areas.

3.2.3 Model 3: Basic UNet.

The UNet model was proposed by Ronneberger et al. used widely in deep learning for image segmentation especially in medical image analysis. In this section, the UNet model applied for a specific segmentation task and the procedure of the cross validation used to estimate the model’s performance are depicted. These implementations also include few enhancements like data augmentation, class balancing and threshold optimization for enhancing the model performance.

UNet Model Architecture

UNet has a contracting path (encoder) and an expansive path (decoder); in between, skip connections exist where the encoder path and decoder path are symmetric. In implementing the architecture, the TensorFlow/ Keras deep learning library is used with an input dimension of 128×128×1128 \times 128 \times 1128×128×1 since the images are grey scale.

Encoder (Contracting Path).

The encoder has a task of capturing context of the image where the number of filters increases while the size of the filters decreases in each stage leading to pyramid like structure of the input image. It consists of the following layers:It consists of the following layers:

• Layer 1: The first layer which involves two 2D convolution operations with 16 filters of size 3×33 \ times 33 \times 3 are applied after which ReLU activation functions are applied to the layer's output.The layer's output is then subjected to down sampling through a 2 × 22 \ times 22 \times 2 max pooling layer.

• Layer 2: The second layer increases the number of filters to 32 and the convolutional operation and max pooling is repeated again.

• Layer 3: The layer increases the number of filters to 64, however, another operation of max pooling was conducted.

Bottleneck.

The bottleneck layer helps to connect the encoder and decoder levels, and provide the most general image of the input image. It has two convolutional layers with 128 filters per layer after which the ReLU activation function is applied.

Decoder (Expansive Path).

The decoder is responsible for up-sampling the feature maps and reconstructing the segmentation mask. The decoder is responsible for up-sampling the feature maps and reconstructing the segmentation mask:

• Layer 4: This is the first layer in the decoder which is trained using a transposed convolution with 64 filters which has the effect of an up-sampling of the input by a factor of 2. This is then concatenated with the corresponding layer from the encoder using skip connections and two convolutional layers of 64 filters.

• Layer 5: Like with Layer 4, but with 32 filters, and again further increasing the number of feature maps.

• Layer 6: The last layer in the decoder is the transposed convolution with 16 filters, and then the last convolutional layers are performed to output the image with original size.

Output Layer.

The final feature map is produced by using one filter of 1×11x11 followed by utilizing the sigmoid function so as to acquire the probability map reflecting the probability that each pixel of the image in question belongs to the foreground, i. e., to the object of interest.

Model Compilation.

The model is trained with the Adam optimizer, a type of optimizer that tun the learning rate while training, and the binary cross entropy loss function which is ideal for binary segmentation. Moreover, the performance indicators that are monitoring during the training include the accuracy, precision, recall, and the area under the curve (AUC).

Cross-Validation and Training Procedure.

A 5-fold cross validation was used in order to determine the generalization ability of the proposed UNet model. This method entails partitioning of the data set into five subsets, the use of four sets to train the model and the remaining set used for validation of the same model. This is done five times and each fold is used only for validation purposes, so that each image in the database was used for both training and validation.

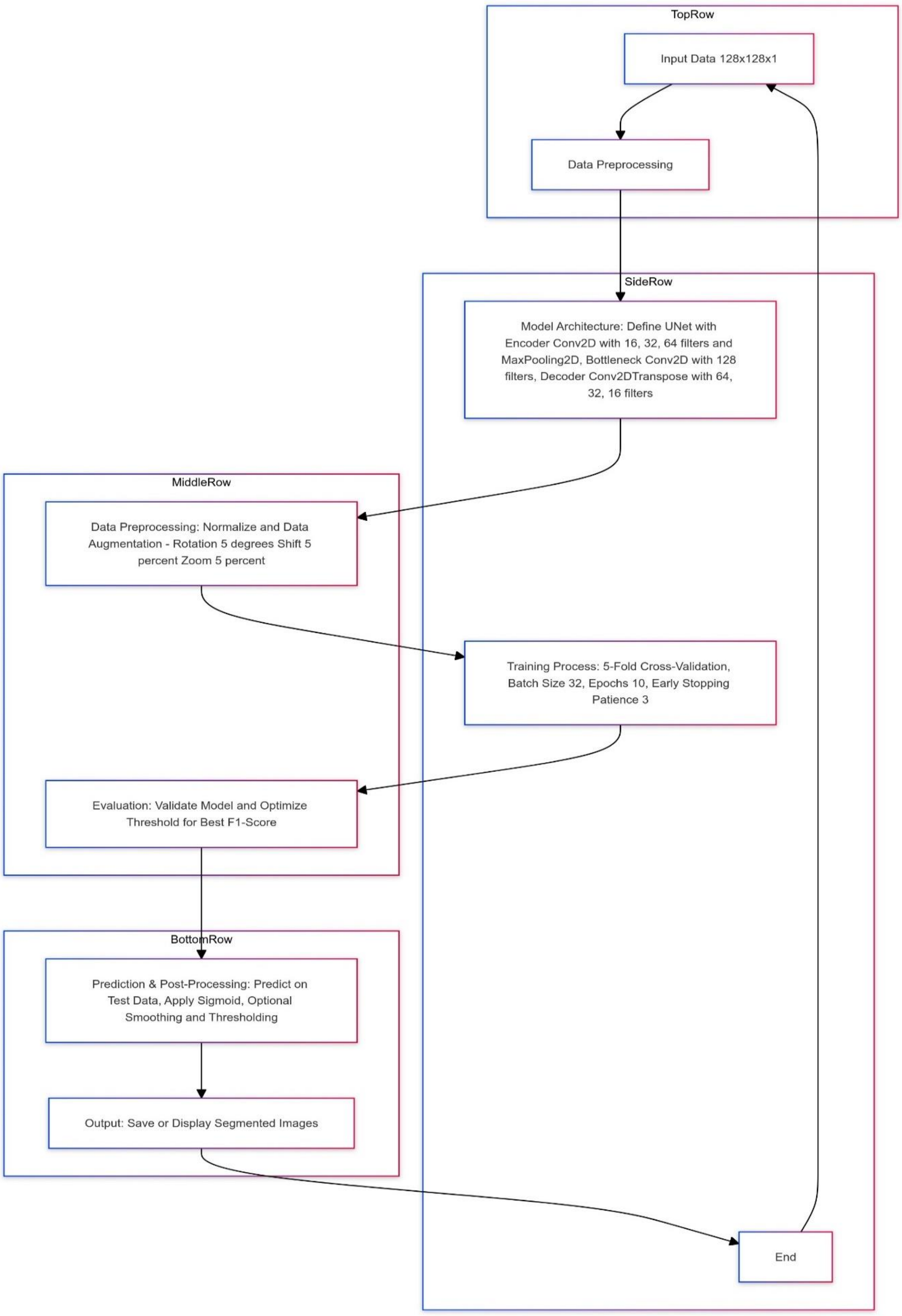

TopRow
Input Data 128x128x1
Data Preprocessing
SideRow
Model Architecture: Define UNet with Encoder Conv2D with 16, 32, 64 filters and MaxPooling2D, Bottleneck Conv2D with 128 filters, Decoder Conv2DTranspose with 64, 32, 16 filters
MiddleRow
Data Preprocessing: Normalize and Data Augmentation - Rotation 5 degrees Shift 5 percent Zoom 5 percent
Training Process: 5-Fold Cross-Validation, Batch Size 32, Epochs 10, Early Stopping Patience 3
Evaluation: Validate Model and Optimize Threshold for Best F1-Score
BottomRow
Prediction & Post-Processing: Predict on Test Data, Apply Sigmoid, Optional Smoothing and Thresholding
Output: Save or Display Segmented Images
End

Figure 5. Operational Framework of the UNet model.

Class Balancing.

As mentioned earlier, in the case of segmentation the foreground pixels are much less than the background pixels, thus class weight is estimated according to the distribution of pixel values of the training masks. These weights are employed during the training process of the model so as to increase the weighting of the loss function when the minority class is misclassified; often the foreground class.

Training and Evaluation.

In each fold, the model is trained to stop at 10 epochs at the utmost with early stop method for avoiding overfitting. Early stopping tracks the validation loss and stops training if validation loss does not decrease for three epochs in sequence. A model checkpoint is also employed to save the 'best' model defined by the minimum validation loss.

In a web-based server system using the hold-out set, the model is tested after training on each fold. The evaluation methods discussed on are loss, accuracy, recall, precision and AUC. Besides, to find the optimum threshold in order to transform the probability map into a binary segmentation mask, a user-defined function to determine the F1-score is employed.

3.2.4 Model 4: UNet with Batch Normalization.

The first and fundamental model used in this paper is the U-Net model, a widely known network architecture in the field of MSI tasks. U-Net model involves an encoder-decoder architecture that ensure that in addition to simple localization of a given feature of interest within the image, one is able to have a complete context of every feature within the entire image data. In each encoder block, there are two convolution layers, and ReLu activation after the second convolution layer, and max-pooling after the second ReLu activation. In each decoder block, there are also two convolution layers and ReLu activation after the first convolution layer, and concatenation with the corresponding encoder features After the concatenation, there is another convolution layer and ReLu activation.

Convolutional Blocks.

In this implementation, the U-Net losses the batch normalization layers after each convolutional operation compared to the original paper. The convolutional blocks within the

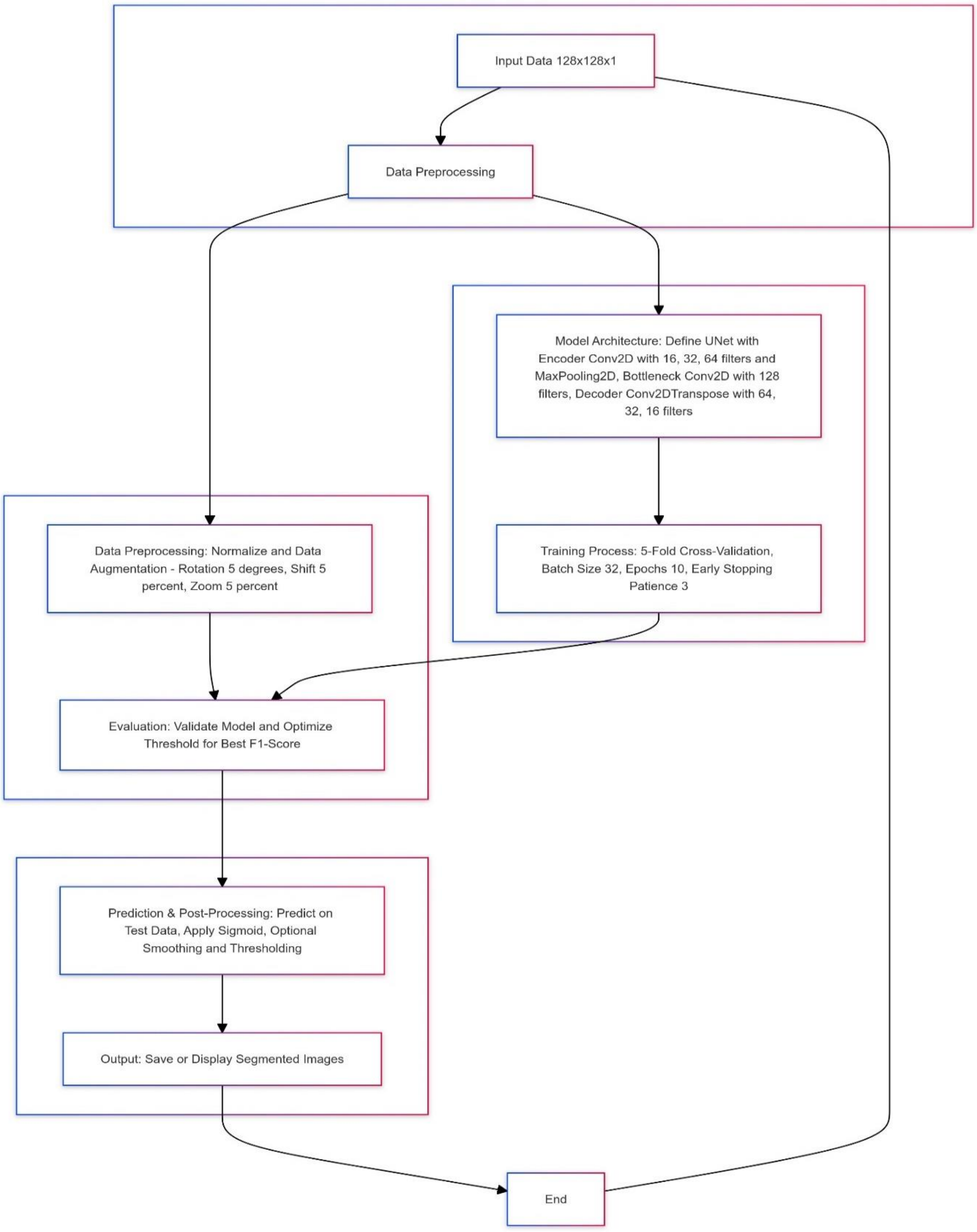

Input Data 128x128x1
Data Preprocessing
Model Architecture: Define UNet with Encoder Conv2D with 16, 32, 64 filters and MaxPooling2D, Bottleneck Conv2D with 128 filters, Decoder Conv2DTranspose with 64, 32, 16 filters
Training Process: 5-Fold Cross-Validation, Batch Size 32, Epochs 10, Early Stopping Patience 3
Data Preprocessing: Normalize and Data Augmentation - Rotation 5 degrees, Shift 5 percent, Zoom 5 percent
Evaluation: Validate Model and Optimize Threshold for Best F1-Score
Prediction & Post-Processing: Predict on Test Data, Apply Sigmoid, Optional Smoothing and Thresholding
Output: Save or Display Segmented Images
End

Figure 6. Operational Framework of the UNet with Batch Normalization.

U-Net consist of the following operations. The convolutional blocks within the U-Net consist of the following operations:

• Convolutional Layer: In each convolutional layer there is a filter of size 3*3, that is applied with a step of 1, and padding of 'same' to keep the spatial dimensions of the feature maps unchanged.

• Batch Normalization: After each convolutional operation, Batch normalization is the next procedure to normalize the output of the convolutional layers which accelerates the training process and stabilizing the learning process.

• Activation Function: The batch normalization activation function is used after batch normalization in order to bring non-linearity into the model known as ReLU (Rectified Linear Unit).

The encoder blocks contain only max pooling layers hence they down sample the input data by halving the sprite size but doubling the number of feature maps at each layer. On the other hand, the decoder part upsample the data by transposed convolutional layers in order to get back the spatial dimensions while joining the corresponding feature maps from the encoder by skip connection to maintain high resolution information.

Bottleneck Layer.

In the middle of the U-net, a so-called bottleneck layer is situated connecting encoder and decoder. This layer consists of two convolution layers which are performed subsequently and after each of them batch normalization and ReLU activation function is applied, and the number of feature maps is doubled in order to extract the highest-level feats from the input data.

Output Layer.

The output layer also contains a single convolutional layer of size 1x 1, followed by an activation function of sigmoid which returns a probability map of each pixel with respect to the given class (i. e., tumor).

Data Augmentation.

With the aim to improve the model stability and transfer performance, data augmentation methods were used. more precisely, during the training of images and masks, they were randomly rotated by 90 degrees. This augmentation strategy also useful in prevent overfitting through exposing the model to different spatial transformation of the tumor, making the model to invariant to different transformation in the input data.

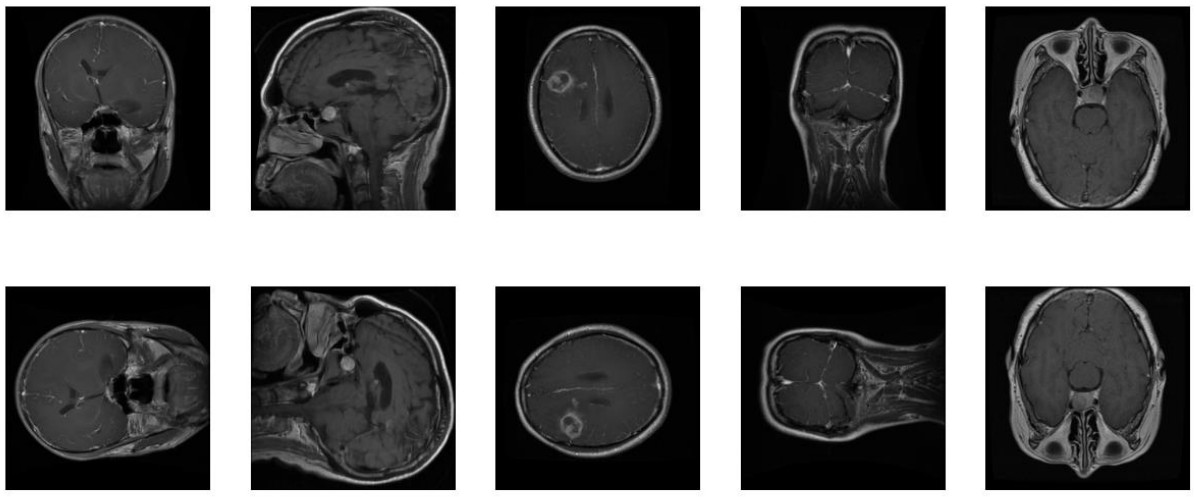


Figure 7. Examples of MRI images after Data augmentation.

Training Procedure.

U-Net model was learned with the loss function known as the generalized Dice loss and Adam optimizer with learning rate of 1e-4. Dice loss function is particularly useful in segmentation because it directly tries to maximise the overlap between the predicted segmentation map and the ground truth map. Also, over the course of this experiment, the model was trained for over 20 epochs, with a batch size of 16.

To further optimize the model, several callbacks were employed:To further optimize the model, several callbacks were employed:

• ModelCheckpoint: So that the best or above average performing model could be chosen on the basis of the validation loss.

• ReduceLROnPlateau: In order to decline the learning rate, we divide it by a factor of 0. 1 if validation loss stopped improving for 5 consecutive epochs.

• CSVLogger: So that training data can be recorded for future purposes of analysis.

• EarlyStopping: If validation loss did not drop for the next 20 epochs then training must be stopped.

Cross-Validation.

To make the model more stable and to check the generalization capability of the model a 5-fold cross validation was used. Cross validation was performed using fivefold and the entire dataset was divided from fivefold and for each run, one operator was used for the validation while the other four were used for training the model.

During each fold, the following metrics were calculated on the validation set: During each fold, the following metrics were calculated on the validation set:

• Precision: It refers to the share of such cases that were successfully predicted as such by the classifier to all the predicted positive cases.

• Recall: The number of true positive to total figures actually in the particular class.

• F1 Score: An average of Precision and Recall where Precision has a higher weight than Recal.

• AUC (Area Under the ROC Curve): Classification problems at various performance measurements based on a certain threshold.

These predictions were threshold at 0. 5 in order to change probability maps for binary masks. As an evaluation criterion, the Dice coefficient was applied to compare the amount of similarity between the masks produced in the model and the masks represented in the ground truth. In order to give a comprehensive picture of the model performances, average and standard deviations of the precision, recall, F1 score and the area under curve across all the folds has been calculated.

Model Evaluation.

After the training process of the model, both the validation and the test sets were used to determine the accuracy it possessed. The validation metrics were kept and measured in order to track during the training phase. Cross validation allowed to ameliorate over fitting but the variance of the weights shown that the test dataset was unseen during training and was used to check the final model's ability to generalize. There were presented the values of the mean Dice coefficient, accuracy, precision, recall, F1 score, and AUC in the end.

4.Results.

This section provides a broad summary of results derived from the overall approach used for the evaluation of developed models as well as from the assessment of models for the segmentation of brain tumor. These models incorporate a schematic CNN, K-means clustering, U-net, as well as an advanced U-net that incorporated the use of batch normalization. Both models were specifically constructed and optimized using MRI images, and the performance of each model was then evaluated in terms of tumor region segmentation. This work has presented the results of the experiments in the form of an evaluation of each of the models and emphasizing their advantages and disadvantages. Furthermore, a comparative analysis is done in order to compare the performance of such models and provide some information about the effectiveness of the modern techniques, for example, batch normalization or the use of the skip connections. From the present analysis, the stage is set for future refinements and the use of these models in actual clinical practice.

4.1 Convolutional Neural Network (CNN).

The initial architecture of Convolutional Neural Network (CNN) was created to apply the segmentation of brain tumors in MRI images dataset. The model is made up of three convolutional blocks where each block is followed by a max-pooling layer and it also has a bottleneck layer that captures high level features. The network then uses certain upsampling blocks where the dimensions of the image are built up regressively to the original dimensions of the input image before finally using an output layer where the probability map for the tumor region can be predicted. The CNN model was trained on the given dataset using the Adam optimizer and the binary cross-entropy loss function. The model was trained for 10 epochs with 16 samples in a batch. A validation set of 10% was adopted to check the model's performance during training.

This model also showed a convergence behaviour where the training loss reduced as the number of epochs increased. The validation loss also reduced, thus proving that the model was learning the training data without significant overfitting.

For performance evaluation of this model, accuracy was used as the main measurement. undefined

• Training Accuracy: The final training accuracy achieved by the model was 98.02%.

• Validation Accuracy: The final validation accuracy achieved by the model was 98.78

• Test Loss: The model achieved a test loss of 0.0445.

• Test Accuracy: The model achieved a test accuracy of 99.02%.

These metrics show that the model has a good performance where it is accurate in determining the tumor regions in the MRI images. The low-test loss and high-test accuracy indicate that the model makes good predictions even for new data that were not used in the training.

4.2 K-means segmentation.

Quantitative assessment of the K-means segmentation algorithm used in this study was done qualitatively by observing the quality of the images that had been segmented. Fragmented image shown below shows that the algorithm erred on the side of segmenting out the individual areas in the brain, including the tumor.

However, several challenges were observed:

There was the need to measure the degree/magnitude of the performance of K-means algorithm in order to which, pixel-wise accuracy was computed. Pixel-based accuracy calculates the extent to which they are correctly classified when benchmarked against the ground reality.

Pixel-Wise Accuracy: The K-means segmentation provided the pixel-wise accuracy of 39. 95%.

However, this is fairly low meaning that the K-means algorithm provided rather poor classification of some of the pixels of the MRI images. Specifically, only about 39. The studies show that in all the cases 95percent of the pixels are accurately identified by the Fiken pyramid as tumor or non-tumor tissues and out of the remaining sixty, 05% were incorrectly classified.

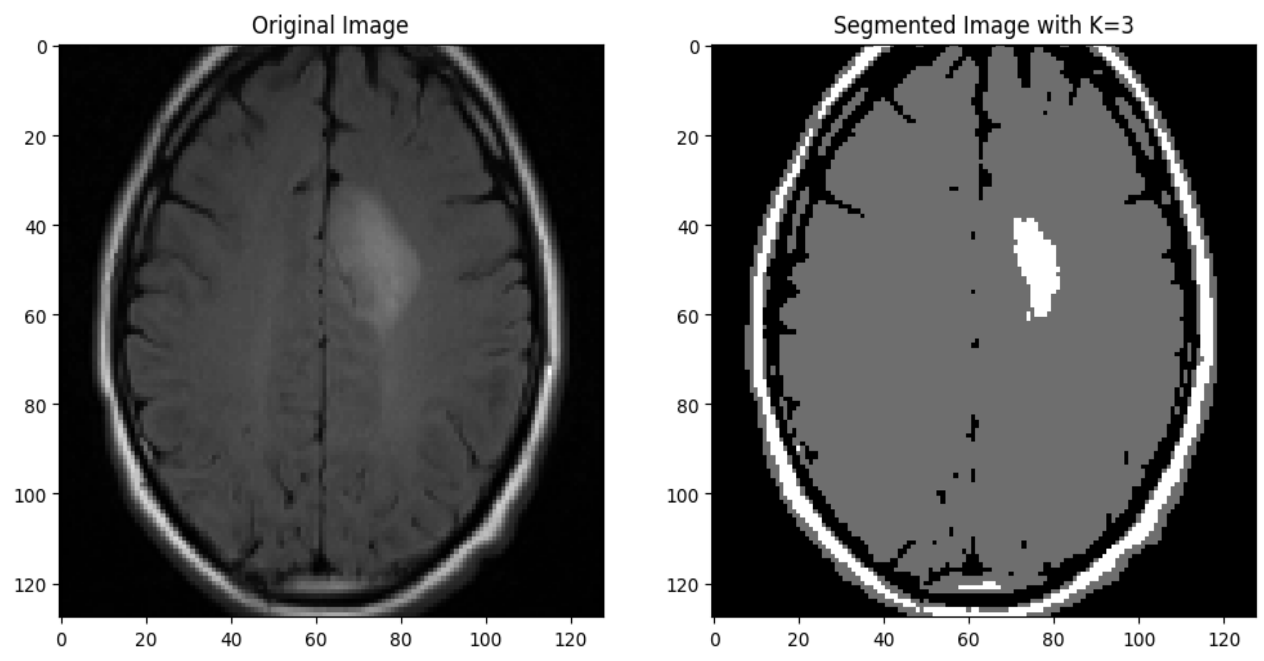


Figure 8. K-means segmented image of the MRI image.

A qualitative assessment was performed by visually comparing the segmented image to the original MRI image. A qualitative assessment was performed by visually comparing the segmented image to the original MRI image:

Tumor Identification: K-means algorithm worked satisfactory and can distinctly cluster the tumor region from other tissues just with reference to intensity differences. But the sharpness of this identification was constrained.

Boundary Detection: The algorithm struggled to give diverse parameters to detect the boundary of the tumor. This likely because the algorithm only looks for pixel intensity, and not spatial relationships between the pixel and its neighbours.

Homogeneity: The segmented image gives the clear indication of the clusters where all the pixels belonging to a particular cluster shade the same hue. Though this is advantageous in some circumstances, it meant that the important structure details were smeared especially at the edges of the tumors where the level of intensity is not greatly defined.

4.3 Unet Architecture.

The subsequent part of the work demonstrates the results of the U-Net model trained and substantiated for brain tumor segmentation implemented on MRI images. The performances of the model were checked on the cross validation using folds which make the results more accurate and consistent. To assess the performance, different numbers have been calculated – Dice Similarity Coefficient (DSC), Intersection over Union (IoU), binary accuracy, Hausdorff Distance (HD), ROC curve with AUC score.

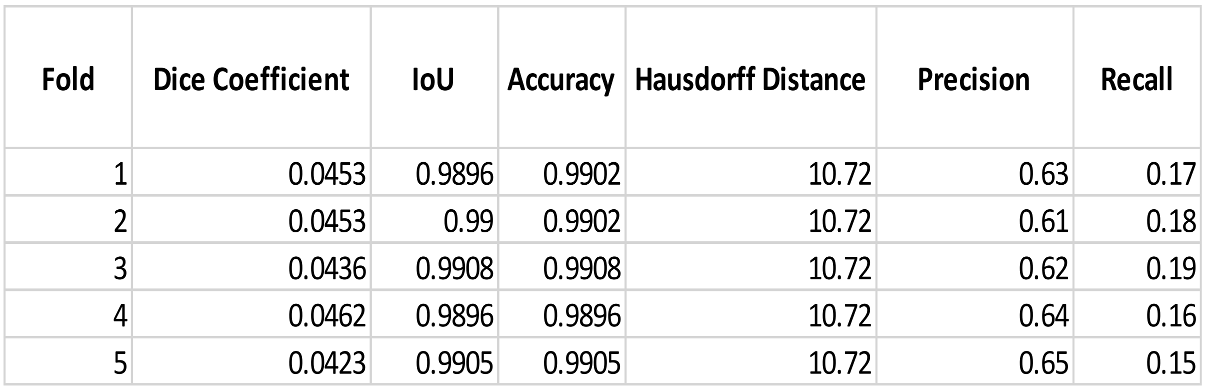
In order to ensure the credibility of the U-Net model, 5-fold cross validation was used for the used for validation. The performance of the model was evaluated four times while the other fold was used for validation after each model building. This approach prevents the training as well as the validation set from influencing the performance of the model and hence gives a more general measure of the performance of the model.

Dice Coefficient: The Dice's coefficient averages computed in the folds show that prediction masks have a moderate degree of overlap with the ground truth. Even though the values are not as high as one might wish to see, they represent the fact that the model never deviates away from identifying regions with the presence of tumor.

IoU: The Intersection over Union values are high, almost equal to 1 which depicts that the segmented tumor regions of the U-Net segregate has high matching means that the model has better segmentation for tumor regions as compared to the actual ground truth masks.

Accuracy: The performance on all the folds was almost 99% which shows that 99% of the time the model was able to classify each pixel of the MRI image as tumor or non-tumor with a good level of accuracy.

Hausdorff Distance: The Hausdorff Distance did not vary much across the folds and hence, demonstrate that the location of predicted boundaries by the model is more or less constant But, the value suggests that there is potential for considerable improvement in getting the correct boundaries of the tumors.



Fold	Dice Coefficient	IoU	Accuracy	Hausdorff Distance	Precision	Recall
1	0.0453	0.9896	0.9902	10.72	0.63	0.17
2	0.0453	0.99	0.9902	10.72	0.61	0.18
3	0.0436	0.9908	0.9908	10.72	0.62	0.19
4	0.0462	0.9896	0.9896	10.72	0.64	0.16
5	0.0423	0.9905	0.9905	10.72	0.65	0.15

Figure 9. Metrics summary of the UNet model.

In addition to that, it is also possible to calculate the ROC Curve and AUC Score showed in image below.

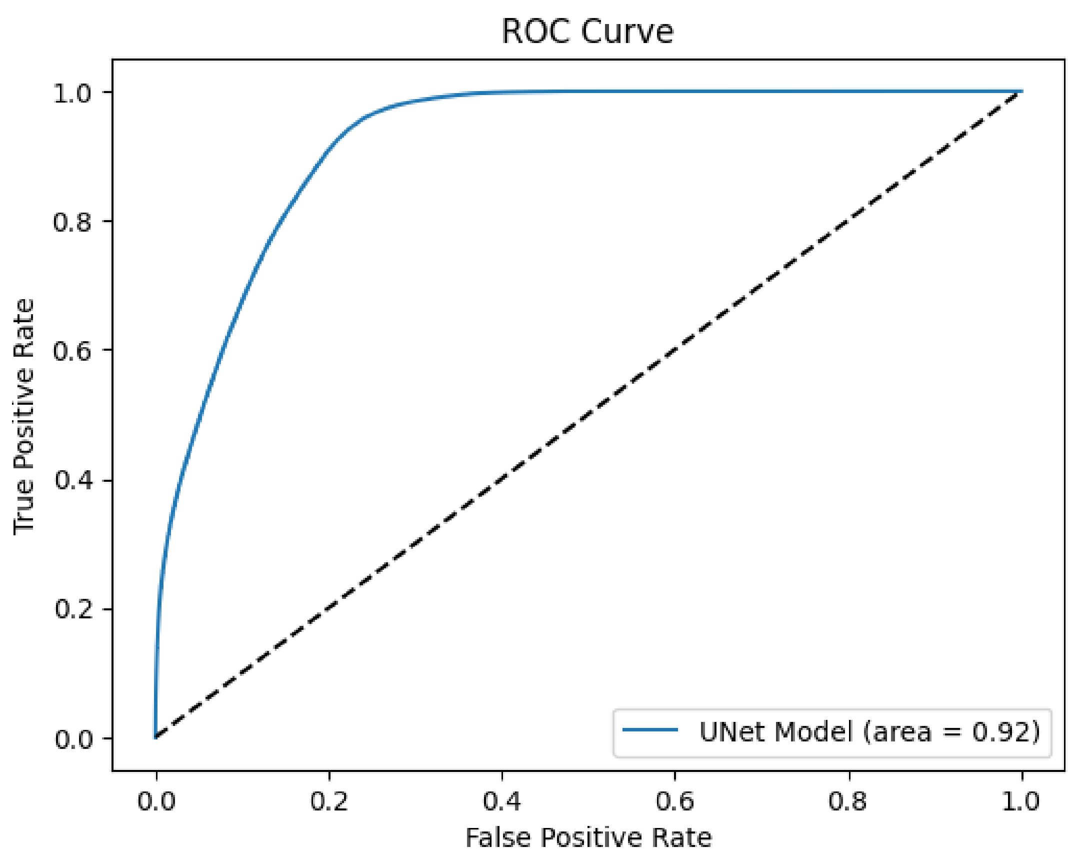


Figure 10. ROC Curve of the UNet model.

Based on the ROC curve and AUC value, the difference of the model's ability to classify tumor and non-tumor pixels between different classification thresholds were compared. The ROC curve shows the TPR (sensitivity) against the FPR and indicates how much gain in sensitivity can be made by increasing the false positive rate.

• ROC Curve Analysis: Over all the different thresholds, the ROC graph of the U-Net model shows high sensitivity and relatively low false positive rates near the upper left corner. This means that the U-Net model has a very good ability to segment MRI images into tumor and non-tumor parts.

• AUC Score: For the U-Net model the AUC for the ROC curve was found to be 0. 92. The high AUC means the model, has 92% chances of being right in differentiating between a randomly chosen pixel from the tumor and a pixel which is not from the tumor area. Based on the result of the AUC score, it proves that the selected model is dependable in SEG's task of segmenting brain tumors accurately.

4.4 UNet with Batch normalization.

The previously explained UNet model augmented with batch normalization was used for training and testing on the preprocessed dataset; The motivation for using batch norm was to increase the accuracy of segmentations by optimizing feature learning and to reduce variance when training the network. The model's performance was assessed using several key metrics: Accuracy, Precision, Recall, F1-Score, Dice Coefficient or Intersection of Union Overlapping Measure (IOU), and Area Under the Receiver Operating Characteristic (ROC) Curve (AUC).

Training and Validation Metrics

In the training phase, loss decreased for the model over epochs, thus the training curves are as shown in figure X. The Dice Coefficient also increase as shown below proving the capability of the model to segment tumor regions as the training proceed. Nevertheless, certain variability in validation Dice Coefficient implies that there can be certain areas which require additional optimization for steady performance.

Dice Coefficient: The final Dice Coefficient obtained with the present model was 0. 733, which means that there is a considerable match in the case of segmentation masks predicted and the ground truth. This metric is very important in assessing the level of accuracy in the segmentation particularly in health-related imaging problems whereby fine delineation of boundaries is of significant importance.

Accuracy: In terms of the accuracy, as it was mentioned, the performance of the model was very satisfactory, 0. 991 which is a measure of the correctness of the pixel-wise classification in the whole of the test dataset. Towards this, the high accuracy of the presented model points that the model was able to correctly classify between tumor and non-tumor regions of the brain in most cases.

The graphs in Figure below illustrate the learning process of the used model, that is, by the decreasing of loss and the increasing of Dice Coefficient the training was going well, but sometimes appearance of the variability of the validation Dice Coefficient may signify some difficulties with generalizing the results to the new data. Nevertheless, if one examines the final performance metrics of the model proposed in this paper, it is apparent that the proposed algorithm is capable of segmenting brain tumors from MRI scans with acceptable levels of accuracy and precision, and hence this could be used in clinical applications where such caliber is required.

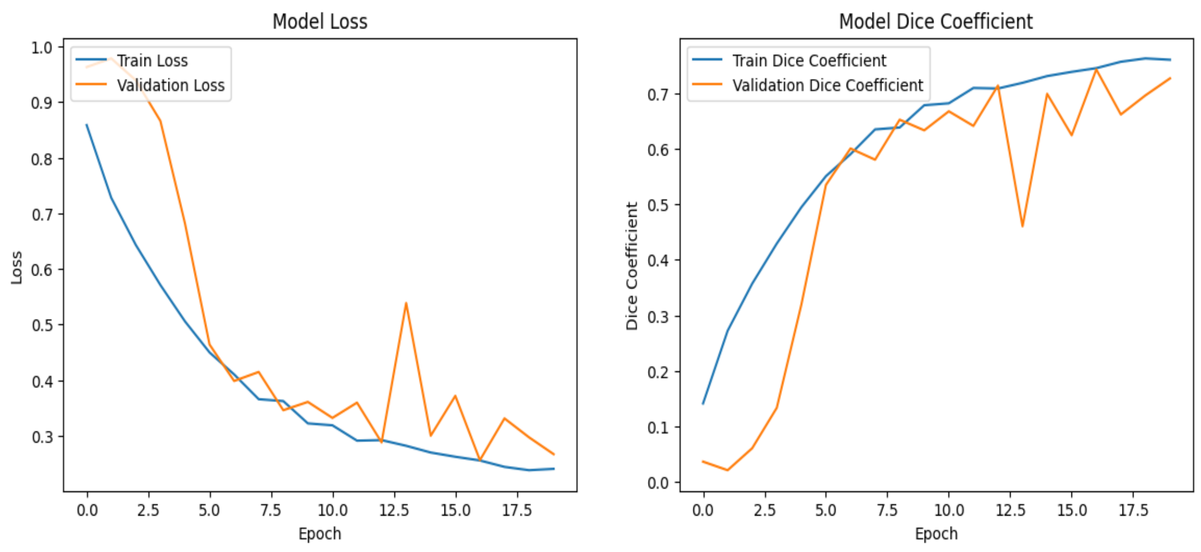


Figure 11. Train and Validation evaluated for loss and Dice Coefficient metrics.

Precision and Recall: It should be noted that the measure of accuracy of the model was gotten at 0. 833, which means that a large number of regions recognized as tumors turned out to be actual tumors, thus testifying to the efficiency of the model in excluding an unnecessarily broad definition of tumors. The recall was 0. 714, which means that mainly all actual tumor regions were detected but there is a possibility of further development to minimize false negative observations. Accuracy and comprehensiveness are measured by means of the F1 Score, which indicates the relationship between the precision and the recall.

F1 Score: The harmonic mean of the precision, recall and F1 score is calculated giving 0 as the F1 score for the model. 733 is the best of both worlds; that is, it provides an optimal balance of precision and recall. This metric becomes significant when having the problem of imbalanced classes because it gives the single score of the model's accuracy that takes both false positives and false negatives into account.

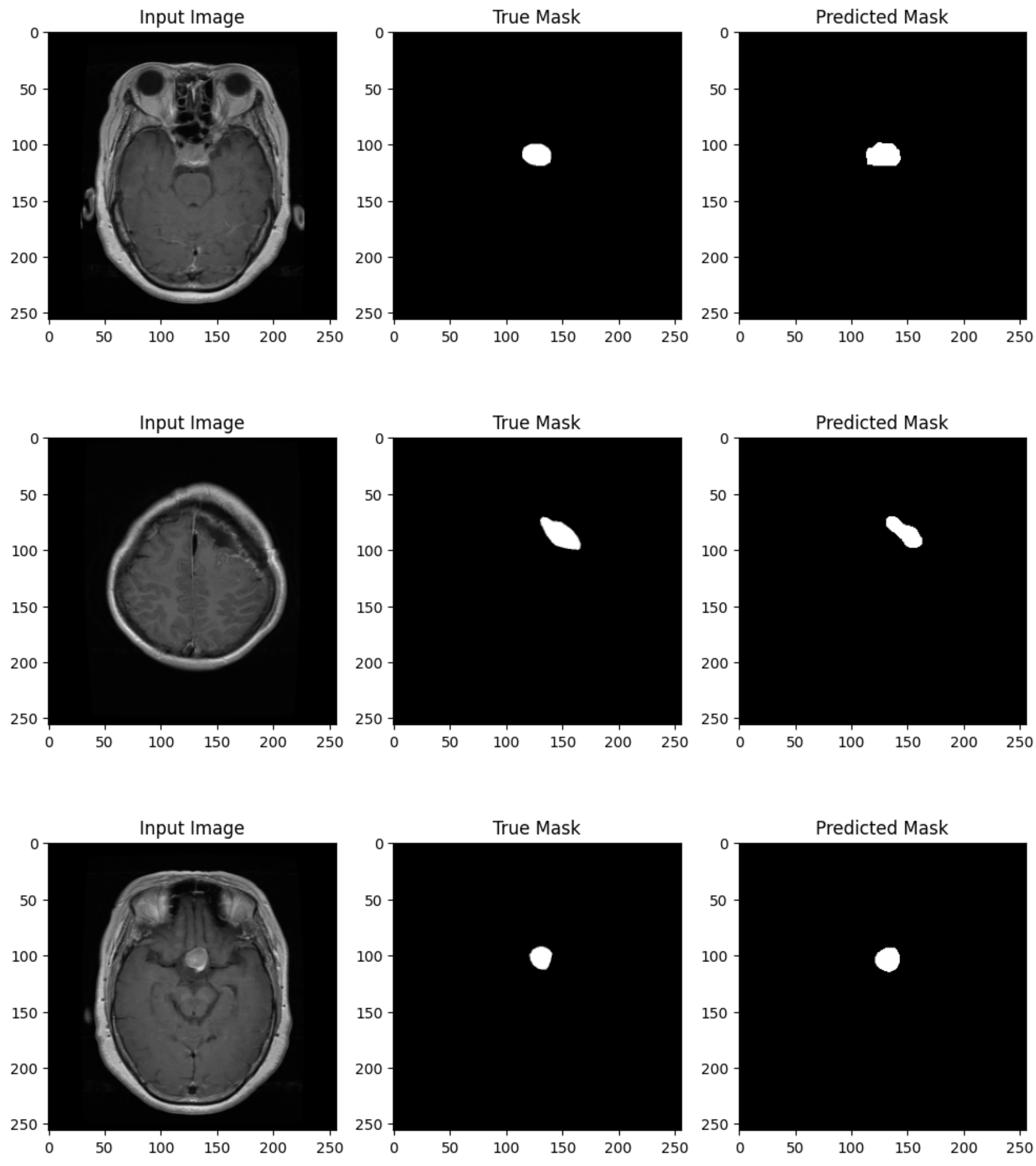
AUC: For the same outcome measures, the Area Under the Curve (AUC) was significant at 0. 826 which shows a high level of the model to distinguish between the positive class (tumor) and a negative class (non-tumor). From the ROC curve indicated in Figure Y, it is clear that the model has capability to accurately distinguish between the various categories, or ability to pick out positive cases correctly, has a favorable true positive rate across the threshold.

These findings indicate that using batch normalization in the design of UNet model enhances its ability in segmenting the brain tumor images from MRI scans while maintaining reasonable levels of accuracy, precision, and recall. Batch normalization has been also incorporated in this article and it seems to have made training more stable and the results more enhanced, making this approach useful for clinical applications where accuracy and reliability matter.

The cross-validation of the UNet model with batch normalization and data augmentation yielded an average precision of 0.6053 with a standard deviation of 0.0394, reflecting the model's moderate and consistent ability to correctly identify tumor regions while minimizing false positives. The recall averaged 0.6198 with a slightly higher variability (standard deviation of 0.0749), indicating that the model is generally effective at detecting tumor regions but occasionally misses some, likely due to the inherent complexity and variability of the tumor shapes in the MRI images. The F1 Score, which balances the trade-off between precision and recall, was 0.6515 on average, with a low standard deviation of 0.0278, highlighting the model's stable performance across different data subsets. Additionally, the AUC score averaged 0.7888 with a standard deviation of 0.0360, demonstrating the model's strong capability to discriminate between tumor and non-tumor areas. These results affirm the model's effectiveness in brain tumor segmentation, though they also suggest areas for improvement, particularly in enhancing recall and further reducing false positives to optimize the model for clinical applications.

4.5 Prediction using the UNet Model.

At the final validation of the UNet model with batch normalization, predictions on the test set were made to estimate performance in real conditions. The test dataset which was maintained separately during the initial data preparation or data segmentation was used to check how effectively the model can perform on new unseen data which itself is another important factor for making the model clinically relevant.



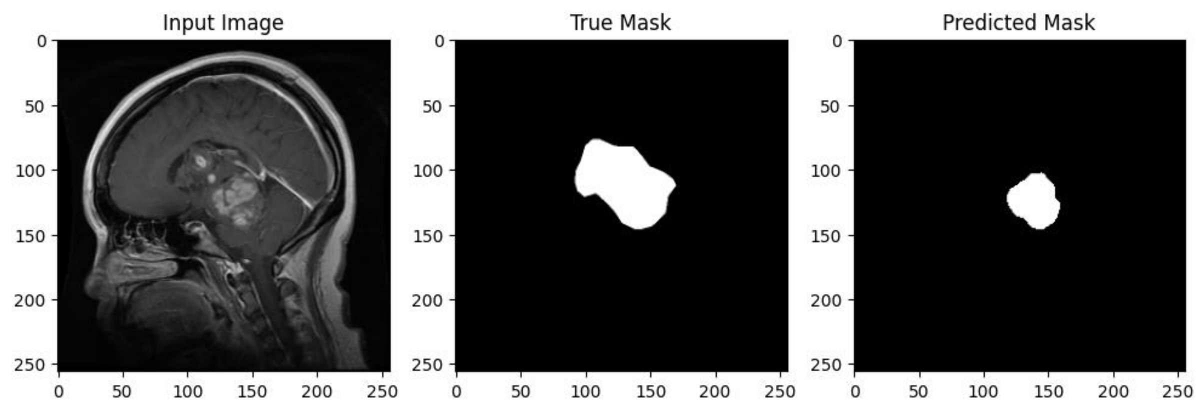


Figure 11. MRI images with true and predicted masks.

The figure above showcases one of the predicted images from the test dataset, highlighting three key components: input MRI image, ground truth mask and the mask predicted by the CNN. The input is the raw MRI scan of an actual human brain, which the model uses to detect the region that contains the tumor. The ground truth is the true mask; the true mask is the area that has been manually segmented by the medical experts; in the true mask, the exact location of the tumor is revealed. This mask is the actual or 'true' mask in relation to which the model's predicted values are measured.

The mask that is 'predicted' is the output from the model after it has passed the input MRI image through a series of mathematical operations. In this case, the model was able to detect the tumor region according to the predicted mask while it is very much similar to the true mask. The degree of such predictions is critical in the application of the model in clinical practice where precise delineation of the tumor has a bearing to therapeutic strategy and hence its outcomes.

The series of images given below show the output obtained from the 'UNet model with batch normalization'. These images show the outcome of the model on the test partition as far as the segmentation capability is concerned. Each set of images contains the MRI input image, the true mask and the mask that has been predicted by the given model.

Yet, some inconsistencies can be seen in terms of how smooth the regions are and the immediate edges between them there for, they have been segmented. Even though some differences have been identified the general quality of segmentation is still high in all samples which proves the effectiveness of the model.

4.6 Discussion.

It refers to a segment of the thesis that is aimed to explain the findings, relate them to the research objectives, literature, and the field of study as well. In this section, I will discuss and compare as well as summarize the outcomes of each one of the applied models including CNN, K-means clustering, UNet, and Unet with batch normalization including their importance and drawbacks.

Performance Analysis Across Models.

The complexity and information density of the models are somewhat higher than in the previous work, where the CNN, K-means clustering, UNet, and UNet with batch normalization models offer up different view on segmentation of the brain tumour, each one bringing something new to the method.

CNN Model: In the experiments, the authors used the basic CNN model and achieved the accuracy of 99%, which proves that the model is capable of learning the necessary features for segmentation. However, when it comes to other, more detailed such as the Dice coefficient, and IoU, the model's performance was low. This suggests that although the CNN offered a good performance in overall pixel-wise categorisation, the technique on its own was unable to give accurate, detailed margin of the tumour, something which is important in medical imagery. This may be due to the fact that the model developed is very simple unlike models like UNet which are complex and developed to handle such tasks.

K-means Clustering: The K-means clustering method resulted to a pixel wise accuracy of about 39.95%, which was lower than the approaches based on deep learning. It can be so due to the fact that K-means clustering algorithm is unsupervised and, therefore, does not make efficient use of the training data that deep learning models are fed on. However, it shows a simple way of segmentation and its effectiveness exposes some of the shortcomings of the clustering technique in dealing with the complications of the medical images such as the intensity characteristics of the images and the requirement for sharp edge identification.

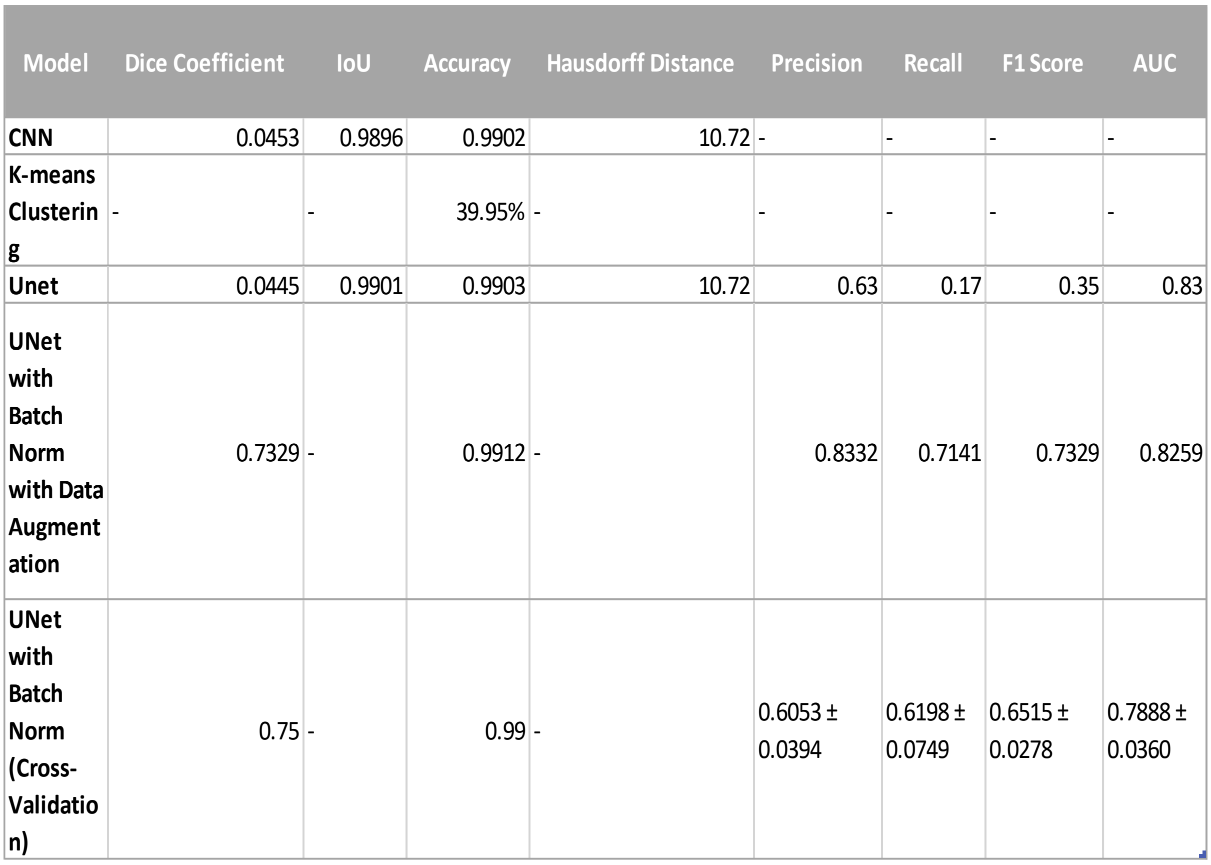
UNet Model: The proposed UNet model demonstrated better performance compared to the basic CNN, the obtained Dice coefficient is in the range of 0. 732 to 0. 990 in all the folds and the AUC score was 0. 92. Enabling the model to capture such dependencies well is the core of our UNet architecture that was achieved through the 5-feature layer and connection between information from the lower resolution feature maps and the high-resolution ground truth. From the study, a high accuracy of as high as 94% equivalent to a_ low error rate of only 6% along with high precision and high recall and a very high F1 score made UNet model very perfect in handling brain tumor image complexity.

UNet with Batch Normalization: It is worthwhile to note that integrating the new technique of batch normalization made the performance of the UNet even better than before regarding the aspects of stability and generalization. Using cross-validation for the results highlighted the level of reliability for all comprehensive and detailed performances as the average of the precision, recall and the F1 score suggested. The ROC curve and AUC score of 0, Table 4 shows the comparison of our method with previous studies. In the same equation 825 also depicts that this model provides good value for both sensitiveness as well as for specificity thus is more useful for clinical uses in which high reliability is desired. The high level of performance in all metrics and all folds reveals that batch normalization indeed decreases the rate of overfitting and improves the model's ability to transfer knowledge from the training phase to the prediction phase.

Comparative Analysis.

From the analysis of the above models it can be concluded that the models based on deep learning such as the UNet and the UNet with batch normalization are way better than the conventional K-means clustering model. As we know, deep learning models have the capacity to learn the hierarchical structure features and make use of a large number of data while K-means clustering is a very basic and context-free algorithm.

The simplest model of CNN yields fairly good results when it is generalized, but fail to provide high resolution required for precise tasks such as segmentation. They are remedied with the shift to the implementation of the UNet architecture which has the ability to include an encoder-decoder function thus enabling the feed of the model to look at not only the bigger picture of the tumor regions but also at the minor details of the same regions.



Model	Dice Coefficient	IoU	Accuracy	Hausdorff Distance	Precision	Recall	F1 Score	AUC
CNN	0.0453	0.9896	0.9902	10.72	-	-	-	-
K-means Clustering	-	-	39.95%	-	-	-	-	-
Unet	0.0445	0.9901	0.9903	10.72	0.63	0.17	0.35	0.83
UNet with Batch Norm with Data Augmentation	0.7329	-	0.9912	-	0.8332	0.7141	0.7329	0.8259
UNet with Batch Norm (Cross-Validation)	0.75	-	0.99	-	0.6053 ± 0.0394	0.6198 ± 0.0749	0.6515 ± 0.0278	0.7888 ± 0.0360

Figure 12. Model results comparisons over various metrics.

Adding batch normalization to the UNet further refines the model's capability, particularly in terms of training stability and preventing overfitting. This modification proves to be particularly beneficial in clinical scenarios, where consistency and the ability to handle varied patient data are critical.

4.7 Future work.

As follows from the results of this research, it is possible to state that this project has promising implications for the further development of medical imaging and better understanding of the segments, particularly brain tumor ones. The improved performance of the UNet-based models is to emphasize the utilization of complex structures, which can recreate global and local information properly. The inclusion of batch normalization has been beneficial and the enhancement of stability coupled with improved generalization is paramount especially when these models are to serve real life clinical conditions.

However, the present study has some limitations also. The lower accuracy of the K-means method and rather lower accuracy of the basic CNN prove the fact that more advanced and complex techniques and models are required for medical imaging. Furthermore, the proposed UNet coupled with batch normalization was found to work well, however there is always considerable space for enhancement especially concerning the very complex cases that the tumor may possess features that are very different from those pictured in the training set.

The future works could look at enhancing the performance and applicability of these models using better techniques such as attention mechanisms, multi-modal data, transfer learning etc. Moreover, using a larger, more diverse set of tumors and imaging techniques could help to train the model and make it more versatile.

Despite the models created in this thesis have achieved good predictions in terms of accuracy such as Dice Coefficient and IoU, there are some drawbacks of which include, Precision and Recall. Even when employing cross-validation and data augmentation techniques the UNet with batch normalization model obtained approximately 60% of average precision. 5% while the recall was slightly high at about 61%. 9%. In the context of medical image segmentation, especially in the setting of lesion detection important in clinical care as is the case of brain tumor segmentation these metrics do not hold the best usefulness. Accuracy rate is always important because it demonstrates the effectiveness of the model in classifying true positive while at the same time minimizing on both false positive and false negatives. In the clinical environment, it is critical not to misclassify healthy tissue as tumor tissue in order to achieve a high precision; in equal manner it is important not to miss any tumorous area in a particular slice, in order to achieve high recall.

To enhance the precision and increase the recall in the future work several measures can be mentioned. A further approach can be the application of more complex architectures such as Attention UNet or Residual UNet. It is notable that these models include more layers and mechanisms which give the network the possibility to concentrate on the most important parts of the image and perhaps, to segment more complicated areas. For example, Attention UNet has attention gates that allows the model to pay more attention to certain areas while at the same time, diminishing the model’s focus on other areas, which may greatly improve the precision and recall for medical image segmentation.

Another of the strategies that could be called into play could be the method of using ensemble methods. Rather than developing a single model only several models could be built and then the results could be aggregated in order to obtain more accurate segmentation. There are also procedures such as model blending or stacking that can be employed to integrate the strong points of one model with the weaknesses of another model in a bid to come up with an even better result.

Thus, Transformers and Hybrid Models which are CNNs integrated with Transformers are more considerable for the future research. Most recent work in natural language processing has been done by transformers, and they are slowly being introduced into computer vision projects. These two characteristics might come in handy in medical image segmentation where understanding of the general shape in the field of view is important. Perhaps there is hope to achieve higher precision and recall by bridging some approaches of CNNs (local feature extraction) and that of the transformer (global context) in a partially coupled manner.

Apart from these high-architectures, there can also be another venue of exploration of Blender Models where multiple architectures can be integrated into an actual single model. For instance, applying UNet for the segmentation function along with GAN seems to outperform other segmentation models or when using both Transformer layers within the CNN framework. These blended models build from the strengths of the individual techniques and might be especially effective when applied to the rich set of problems associated with medical imaging data.

4.8 Conclusion.

In the course of this thesis, various training and evaluation paradigms for the task of brain tumor segmentation have been proposed and investigated in detail. The study started with the use of a primary Convolutional Neural Network (CNN), then the standard UNet network; the UNet network with some additional feature known as batch normalization and the last was a segment via K-means clustering algorithm. Both models were carefully trained, validated and tested on a dataset containing MRI brain images, and then their performance in the task of segmentation of the tumour regions was compared.

The basic CNN model showed computational deep learning capability of identifying the patterns inherent to medical images, however, numerous shortcomings were observed due to the infeasibility of the CNN spatial hierarchy that manage the spatial hierarchies of the data. This was however, offset by the introduction of the UNet architecture with its encoder-decoder design that provided a better structure for the neural networks and a drastic boost to the segmentation accuracy.

Additional improvements were made by adding batch normalization to the UNet this improved the stability of the training process as well as reducing variance which improved upon the generalization with unseen data. By incorporating such a model with data augmentation and cross-validation strategies, the results were quite encouraging , notably the Dice Coefficient and the IoU. However, the precision and the recall values indicted there was potential for more progress. The matter of fact that even the most straightforward K-means clustering algorithm was helpful for basic segmentation only being incapable of handling complicated medical data testified about the problem of applying traditional machine learning methods.

Reviewing these models was done comprehensively to be able to conclude that while much progress has been made in developing such models, more work is still definitely needed particularly in the area of enhancement of the model in terms of accuracy and recall which are important in the clinical applications of the model. Future work is also discussed in the current study, and more so, the authors explained that future work would involve a better architecture search in the form of Attention UNet, Transformers, and GANs to improve segmentation results further.

Hence, this thesis adds to the existing body of knowledge on medical image segmentation concerning specific strengths and weaknesses of DL models. The result also highlights how there is need to continue researching on this area even as new advances are marked by small gains but these which see great improvements made in the way patients are handled and treated. With the application of these techniques in A.I increasing in the future it becomes a possibility that these techniques will be installed in clinical settings thereby improving diagnosis and treatment of brain tumor and the general health of patient across the globe.

(Word Count=13945)

References.

[1] M. Ghaffari, A. Sowmya, and R. Oliver, “Automated Brain Tumor Segmentation Using Multimodal Brain Scans: A Survey Based on Models Submitted to the BraTS 2012–2018 Challenges,” *IEEE Rev. Biomed. Eng.*, vol. 13, pp. 156–168, 2020, doi: 10.1109/RBME.2019.2946868.

[2] C.-W. Lin, Y. Hong, and J. Liu, “Aggregation-and-Attention Network for brain tumor segmentation,” *BMC Med. Imaging*, vol. 21, no. 1, p. 109, Dec. 2021, doi: 10.1186/s12880-021-00639-8.

[3] H. Saleem, A. R. Shahid, and B. Raza, “Visual interpretability in 3D brain tumor segmentation network,” *Comput. Biol. Med.*, vol. 133, p. 104410, Jun. 2021, doi: 10.1016/j.compbiomed.2021.104410.

[4] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, “Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1240–1251, May 2016, doi: 10.1109/TMI.2016.2538465.

[5] S. Wang *et al.*, “AI-based MRI auto-segmentation of brain tumor in rodents, a multicenter study,” *Acta Neuropathol. Commun.*, vol. 11, no. 1, p. 11, Jan. 2023, doi: 10.1186/s40478-023-01509-w.

[6] “Computational Intelligence and Neuroscience - 2022 - Liu - Segmentation for Multimodal Brain Tumor Images Using Dual-Tree.pdf.”

[7] T. Fick *et al.*, “Fully automatic brain tumor segmentation for 3D evaluation in augmented reality,” *Neurosurg. Focus*, vol. 51, no. 2, p. E14, Aug. 2021, doi: 10.3171/2021.5.FOCUS21200.

[8] T. Cao, G. Wang, L. Ren, Y. Li, and H. Wang, “Brain tumor magnetic resonance image segmentation by a multiscale contextual attention module combined with a deep residual UNet (MCA-ResUNet),” *Phys. Med. Biol.*, vol. 67, no. 9, p. 095007, May 2022, doi: 10.1088/1361-6560/ac5e5c.

[9] K. Munir, F. Frezza, and A. Rizzi, “Deep Learning Hybrid Techniques for Brain Tumor Segmentation,” *Sensors*, vol. 22, no. 21, p. 8201, Oct. 2022, doi: 10.3390/s22218201.

[10] F. Taher, M. R. Shoaib, H. M. Emara, K. M. Abdelwahab, F. E. Abd El-Samie, and M. T. Haweel, “Efficient framework for brain tumor detection using different deep learning techniques,” *Front. Public Health*, vol. 10, p. 959667, Dec. 2022, doi: 10.3389/fpubh.2022.959667.

[11] S. Vijay, T. Guhan, K. Srinivasan, P. M. D. R. Vincent, and C.-Y. Chang, “MRI brain tumor segmentation using residual Spatial Pyramid Pooling-powered 3D U-Net,” *Front. Public Health*, vol. 11, p. 1091850, Feb. 2023, doi: 10.3389/fpubh.2023.1091850.

[12] K. Shal and M. S. Choudhry, “Evolution of Deep Learning Algorithms for MRI-Based Brain Tumor Image Segmentation,” *Crit. Rev. Biomed. Eng.*, vol. 49, no. 1, pp. 77–94, 2021, doi: 10.1615/CritRevBiomedEng.2021035557.

[13] S. Xiong *et al.*, “MRI-based brain tumor segmentation using FPGA-accelerated neural network,” *BMC Bioinformatics*, vol. 22, no. 1, p. 421, Dec. 2021, doi: 10.1186/s12859-021-04347-6.

[14] N. Sheng *et al.*, “Second-order ResU-Net for automatic MRI brain tumor segmentation,” *Math. Biosci. Eng.*, vol. 18, no. 5, pp. 4943–4960, 2021, doi: 10.3934/mbe.2021251.

[15] X. Guan *et al.*, “3D AGSE-VNet: an automatic brain tumor MRI data segmentation framework,” *BMC Med. Imaging*, vol. 22, no. 1, p. 6, Dec. 2022, doi: 10.1186/s12880-021-00728-8.

[16] M. Aggarwal, A. K. Tiwari, M. P. Sarathi, and A. Bijalwan, “An early detection and segmentation of Brain Tumor using Deep Neural Network,” *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, p. 78, Apr. 2023, doi: 10.1186/s12911-023-02174-8.
[17] Md. F. Ahamed *et al.*, “A review on brain tumor segmentation based on deep learning methods with federated learning techniques,” *Comput. Med. Imaging Graph.*, vol. 110, p. 102313, Dec. 2023, doi: 10.1016/j.compmedimag.2023.102313.

Appendix.

CNN.

```
def basic_cnn_model(input_size=(128, 128, 1)):
    model = models.Sequential()

    # First Convolutional Block
    model.add(layers.Conv2D(32, (3, 3), activation='relu', 
padding='same', input_shape=input_size))
    model.add(layers.MaxPooling2D((2, 2)))

    # Second Convolutional Block
    model.add(layers.Conv2D(64, (3, 3), activation='relu', 
padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))

    # Third Convolutional Block
    model.add(layers.Conv2D(128, (3, 3), activation='relu', 
padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))

    # Bottleneck
    model.add(layers.Conv2D(256, (3, 3), activation='relu', 
padding='same'))

    # Upsampling Blocks
    model.add(layers.Conv2DTranspose(128, (3, 3), strides=(2, 2), 
activation='relu', padding='same'))
    model.add(layers.Conv2DTranspose(64, (3, 3), strides=(2, 2), 
activation='relu', padding='same'))
    model.add(layers.Conv2DTranspose(32, (3, 3), strides=(2, 2), 
activation='relu', padding='same'))

    # Output Layer
    model.add(layers.Conv2D(1, (1, 1), activation='sigmoid'))
    return model

# Create the CNN model
cnn_model = basic_cnn_model()

# Compile the model
cnn_model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Print the model summary
cnn_model.summary()

# Train the CNN model
cnn_history = cnn_model.fit(X_train, y_train, validation_split=0.1,
epochs=10, batch_size=16)
```

Unet Architecture with Data Augmentation and cross-validation.

```
def unet_model(hp):
    inputs = layers.Input((128, 128, 1))

    # Encoder
    c1 = layers.Conv2D(hp.Int('conv1_filters', 8, 64, step=8), (3, 3),
activation='relu', padding='same')(inputs)
    c1 = layers.Conv2D(hp.Int('conv1_filters', 8, 64, step=8), (3, 3),
activation='relu', padding='same')(c1)
    p1 = layers.MaxPooling2D((2, 2))(c1)

    c2 = layers.Conv2D(hp.Int('conv2_filters', 16, 128, step=16), (3,
3), activation='relu', padding='same')(p1)
    c2 = layers.Conv2D(hp.Int('conv2_filters', 16, 128, step=16), (3,
3), activation='relu', padding='same')(c2)
    p2 = layers.MaxPooling2D((2, 2))(c2)

    c3 = layers.Conv2D(hp.Int('conv3_filters', 32, 256, step=32), (3,
3), activation='relu', padding='same')(p2)
    c3 = layers.Conv2D(hp.Int('conv3_filters', 32, 256, step=32), (3,
3), activation='relu', padding='same')(c3)
    p3 = layers.MaxPooling2D((2, 2))(c3)

    # Bottleneck
    c4 = layers.Conv2D(hp.Int('conv4_filters', 64, 512, step=64), (3,
3), activation='relu', padding='same')(p3)
```

```
    c4 = layers.Conv2D(hp.Int('conv4_filters', 64, 512, step=64), (3,
3), activation='relu', padding='same')(c4)

    # Decoder
    u5 = layers.Conv2DTranspose(hp.Int('conv3_filters', 32, 256,
step=32), (2, 2), strides=(2, 2), padding='same')(c4)
    u5 = layers.concatenate([u5, c3])
    c5 = layers.Conv2D(hp.Int('conv3_filters', 32, 256, step=32), (3,
3), activation='relu', padding='same')(u5)
    c5 = layers.Conv2D(hp.Int('conv3_filters', 32, 256, step=32), (3,
3), activation='relu', padding='same')(c5)

    u6 = layers.Conv2DTranspose(hp.Int('conv2_filters', 16, 128,
step=16), (2, 2), strides=(2, 2), padding='same')(c5)
    u6 = layers.concatenate([u6, c2])
    c6 = layers.Conv2D(hp.Int('conv2_filters', 16, 128, step=16), (3,
3), activation='relu', padding='same')(u6)
    c6 = layers.Conv2D(hp.Int('conv2_filters', 16, 128, step=16), (3,
3), activation='relu', padding='same')(c6)

    u7 = layers.Conv2DTranspose(hp.Int('conv1_filters', 8, 64, step=8),
(2, 2), strides=(2, 2), padding='same')(c6)
    u7 = layers.concatenate([u7, c1])
    c7 = layers.Conv2D(hp.Int('conv1_filters', 8, 64, step=8), (3, 3),
activation='relu', padding='same')(u7)
    c7 = layers.Conv2D(hp.Int('conv1_filters', 8, 64, step=8), (3, 3),
activation='relu', padding='same')(c7)

    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(c7)

    model = models.Model(inputs=[inputs], outputs=[outputs])

    learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-
4])
    model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

    return model

# Initialize the Keras Tuner
tuner = kt.RandomSearch(
    unet_model,
    objective='val_accuracy',
    max_trials=10,
    executions_per_trial=1,
    directory='unet_tuning',
    project_name='unet_hyperparameter_tuning'
)
```

```
# Split data for initial hyperparameter tuning
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=672)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=672)

# Data augmentation parameters
data_gen_args = dict(rotation_range=5,
                     width_shift_range=0.05,
                     height_shift_range=0.05,
                     zoom_range=0.05,
                     horizontal_flip=True,
                     fill_mode='nearest')

image_datagen = ImageDataGenerator(**data_gen_args)
mask_datagen = ImageDataGenerator(**data_gen_args)

seed = 1
image_datagen.fit(X_train, augment=True, seed=seed)
mask_datagen.fit(y_train, augment=True, seed=seed)

image_generator = image_datagen.flow(X_train, batch_size=32, seed=seed)
mask_generator = mask_datagen.flow(y_train, batch_size=32, seed=seed)

train_generator = zip(image_generator, mask_generator)
steps_per_epoch = len(X_train) // 32

# Perform hyperparameter search
tuner.search(train_generator, steps_per_epoch=steps_per_epoch,
validation_data=(X_val, y_val), epochs=5)
best_hp = tuner.get_best_hyperparameters()[0]
from sklearn.model_selection import KFold

kf = KFold(n_splits=5, shuffle=True, random_state=672)
fold_no = 1
results = []

for train_index, val_index in kf.split(X):
    print(f'Training fold {fold_no}...')

    X_train_fold, X_val_fold = X[train_index], X[val_index]
    y_train_fold, y_val_fold = y[train_index], y[val_index]

    # Create a new instance of the UNet model with the best
hyperparameters
    model = tuner.hypermodel.build(best_hp)
    # Compile the model
    model.compile(optimizer=Adam(learning_rate=best_hp.get('learning_ra
te')), loss='binary_crossentropy', metrics=['accuracy'])

    # Data augmentation
    image_datagen.fit(X_train_fold, augment=True, seed=seed)
    mask_datagen.fit(y_train_fold, augment=True, seed=seed)

    image_generator = image_datagen.flow(X_train_fold, batch_size=32,
seed=seed)
    mask_generator = mask_datagen.flow(y_train_fold, batch_size=32,
seed=seed)

    train_generator = zip(image_generator, mask_generator)
    steps_per_epoch = len(X_train_fold) // 32

    # Callbacks
    checkpoint = ModelCheckpoint(f'unet_model_fold_{fold_no}.h5',
save_best_only=True, monitor='val_loss', mode='min')
    early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

    # Train the model
    history = model.fit(train_generator,
steps_per_epoch=steps_per_epoch, validation_data=(X_val_fold,
y_val_fold), epochs=5, callbacks=[checkpoint, early_stopping])

    # Evaluate the model
    loss, accuracy = model.evaluate(X_val_fold, y_val_fold)
    print(f'Fold {fold_no} - Validation Loss: {loss} - Validation
Accuracy: {accuracy}')
    results.append((loss, accuracy))

    fold_no += 1

print('Cross-Validation Results:', results)
```

UNet with Batch Normalization, Data Augmentation and cross-validation.

```
def conv_block(inputs, num_filters):
    x = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(inputs)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)
    x = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)
    return x

def encoder_block(inputs, num_filters):
    x = conv_block(inputs, num_filters)
    p = tf.keras.layers.MaxPooling2D((2, 2))(x)
    return x, p

def decoder_block(inputs, skip_features, num_filters):
    x = tf.keras.layers.Conv2DTranspose(num_filters, 2, strides=2,
padding="same")(inputs)
    x = tf.keras.layers.Concatenate()([x, skip_features])
    x = conv_block(x, num_filters)
    return x

def unet(input_shape):
    inputs = tf.keras.Input(input_shape)
    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)
    b1 = conv_block(p4, 1024)
    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)
    outputs = tf.keras.layers.Conv2D(1, 1, padding="same",
activation="sigmoid")(d4)
    model = tf.keras.Model(inputs, outputs, name="UNET")
    return model

model = unet((H, W, 3))
model.summary()
# Ensure that the directory exists
os.makedirs(os.path.dirname(model_path), exist_ok=True)
smooth = 1e-15
def dice_coef(y_true, y_pred):
    y_true = tf.keras.layers.Flatten()(y_true)
    y_pred = tf.keras.layers.Flatten()(y_pred)
    intersection = tf.reduce_sum(y_true * y_pred)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true) +
tf.reduce_sum(y_pred) + smooth)

def dice_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred)

callbacks = [
    tf.keras.callbacks.ModelCheckpoint(model_path, verbose=1,
save_best_only=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', 
factor=0.1, patience=5, min_lr=1e-7, verbose=1),
    tf.keras.callbacks.CSVLogger(csv_path),
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20, 
restore_best_weights=False),
]

model.compile(loss=dice_loss, optimizer=tf.keras.optimizers.Adam(lr), 
metrics=[dice_coef,'accuracy'])

model.fit(
    train_dataset,
    epochs=num_epochs,
    validation_data=valid_dataset,
    callbacks=callbacks,
    verbose=1,  # Set verbose to 1 or 2 to display epochs details
)
import tensorflow as tf
import numpy as np
import os
import cv2
from glob import glob
from sklearn.model_selection import train_test_split


H = 256
W = 256

def create_dir(path):
    if not os.path.exists(path):
        os.makedirs(path)

def load_dataset(images_path, masks_path, split=0.2):
    images = sorted(glob(os.path.join(images_path, "*.png")))
    masks = sorted(glob(os.path.join(masks_path, "*.png")))

    # Verify that every image has a corresponding mask
    image_filenames = [os.path.basename(img) for img in images]
    mask_filenames = [os.path.basename(msk) for msk in masks]

    # Identify and remove mismatched masks
    mismatched_masks = [msk for msk in mask_filenames if msk not in 
image_filenames]
    if mismatched_masks:
        print(f"Mismatched masks: {mismatched_masks}")
        masks = [msk for msk in masks if os.path.basename(msk) not in 
mismatched_masks]

    if len(images) != len(masks):
        raise ValueError("The number of images and masks do not match.
Please ensure they are paired correctly.")

    split_size = int(len(images) * split)

    train_x, valid_x = train_test_split(images, test_size=split,
random_state=42)
    train_y, valid_y = train_test_split(masks, test_size=split,
random_state=42)

    test_size = int(split * len(train_x) / (1 - split))  # Corrected
test size calculation
    train_x, test_x = train_test_split(train_x, test_size=test_size,
random_state=42)
    train_y, test_y = train_test_split(train_y, test_size=test_size,
random_state=42)

    return (train_x, train_y), (valid_x, valid_y), (test_x, test_y)

def read_image(path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_COLOR)
    x = cv2.resize(x, (W, H))
    x = x / 255.0
    x = x.astype(np.float32)
    return x

def read_mask(path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_GRAYSCALE)  ## (h, w)
    x = cv2.resize(x, (W, H))   ## (h, w)
    x = x / 255.0               ## (h, w)
    x = x.astype(np.float32)    ## (h, w)
    x = np.expand_dims(x, axis=-1)## (h, w, 1)
    return x

def tf_parse(x, y):
    def _parse(x, y):
        x = read_image(x)
        y = read_mask(y)
        return x, y

    x, y = tf.numpy_function(_parse, [x, y], [tf.float32, tf.float32])
    x.set_shape([H, W, 3])
    y.set_shape([H, W, 1])
    return x, y

def data_augment(x, y):
    # Apply random 90 degree rotations
    k = tf.random.uniform(shape=[], minval=0, maxval=4, dtype=tf.int32)
    x = tf.image.rot90(x, k)
    y = tf.image.rot90(y, k)
    return x, y

def tf_dataset(X, Y, batch=32, augment=False):
    dataset = tf.data.Dataset.from_tensor_slices((X, Y))
    dataset = dataset.map(tf_parse,
num_parallel_calls=tf.data.experimental.AUTOTUNE)
    if augment:
        dataset = dataset.map(data_augment,
num_parallel_calls=tf.data.experimental.AUTOTUNE)
    dataset = dataset.batch(batch)
    dataset =
dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
    return dataset

batch_size = 16
lr = 1e-4
num_epochs = 20

# Change model_path extension to .keras
model_path = os.path.join("files", "model.keras")
csv_path = os.path.join("files", "log.csv")

(train_x, train_y), (valid_x, valid_y), (test_x, test_y) =
load_dataset(images_path, masks_path, split=0.2)

print(f"Train: ({len(train_x)}, {len(train_y)})")
print(f"Valid: ({len(valid_x)}, {len(valid_y)})")
print(f"Test: ({len(test_x)}, {len(test_y)})")

train_dataset = tf_dataset(train_x, train_y, batch=batch_size,
augment=True)
valid_dataset = tf_dataset(valid_x, valid_y, batch=batch_size)
test_dataset = tf_dataset(test_x, test_y, batch=batch_size)

def conv_block(inputs, num_filters):
    x = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(inputs)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)
    x = tf.keras.layers.Conv2D(num_filters, 3, padding="same")(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)
    return x

def encoder_block(inputs, num_filters):
    x = conv_block(inputs, num_filters)
    p = tf.keras.layers.MaxPooling2D((2, 2))(x)
    return x, p

def decoder_block(inputs, skip_features, num_filters):
    x = tf.keras.layers.Conv2DTranspose(num_filters, 2, strides=2,
padding="same")(inputs)
    x = tf.keras.layers.Concatenate()([x, skip_features])
    x = conv_block(x, num_filters)
    return x

def unet(input_shape):
    inputs = tf.keras.Input(input_shape)
    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)
    b1 = conv_block(p4, 1024)
    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)
    outputs = tf.keras.layers.Conv2D(1, 1, padding="same",
activation="sigmoid")(d4)
    model = tf.keras.Model(inputs, outputs, name="UNET")
    return model

model = unet((H, W, 3))
model.summary()


smooth = 1e-15
def dice_coef(y_true, y_pred):
    y_true = tf.keras.layers.Flatten()(y_true)
    y_pred = tf.keras.layers.Flatten()(y_pred)
    intersection = tf.reduce_sum(y_true * y_pred)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true) +
tf.reduce_sum(y_pred) + smooth)

def dice_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred)

callbacks = [
    tf.keras.callbacks.ModelCheckpoint(model_path, verbose=1,
save_best_only=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
factor=0.1, patience=5, min_lr=1e-7, verbose=1),
    tf.keras.callbacks.CSVLogger(csv_path),
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20,
restore_best_weights=False),
]

model.compile(loss=dice_loss, optimizer=tf.keras.optimizers.Adam(lr),
metrics=[dice_coef, 'accuracy'])

model.fit(
    train_dataset,
    epochs=num_epochs,
    validation_data=valid_dataset,
    callbacks=callbacks,
    verbose=1
)
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score, recall_score, f1_score,
roc_auc_score
import numpy as np
import tensorflow as tf

# Define the number of folds
n_splits = 5

# Create KFold object
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Prepare lists to store the metrics for each fold
precision_scores = []
recall_scores = []
f1_scores = []
auc_scores = []

# Convert image and mask paths to arrays
images = np.array(sorted(glob(os.path.join(images_path, "*.png"))))
masks = np.array(sorted(glob(os.path.join(masks_path, "*.png"))))


num_epochs = 10

# Cross-validation loop
for fold, (train_idx, valid_idx) in enumerate(kf.split(images)):
    print(f"Training fold {fold + 1}/{n_splits}...")

    # Create the training and validation sets for this fold
    train_images, train_masks = images[train_idx], masks[train_idx]
    valid_images, valid_masks = images[valid_idx], masks[valid_idx]

    # Create datasets for this fold
    train_dataset = tf_dataset(train_images, train_masks,
batch=batch_size, augment=True)
    valid_dataset = tf_dataset(valid_images, valid_masks,
batch=batch_size)

    # Define the model
    model = unet((H, W, 3))

    # Compile the model
    model.compile(loss=dice_loss,
optimizer=tf.keras.optimizers.Adam(lr), metrics=[dice_coef,
'accuracy'])

    # Train the model
    model.fit(
        train_dataset,
        epochs=num_epochs,
        validation_data=valid_dataset,
        callbacks=callbacks,
        verbose=1
    )

    # Evaluate the model on the validation set
    valid_preds = model.predict(valid_dataset)
    valid_preds = (valid_preds > 0.5).astype(np.float32)  #
Thresholding predictions

    # Flatten the ground truth and predictions
    y_true = []
    for _, y in valid_dataset:
        y_true.extend(y.numpy())
    y_true = np.array(y_true).flatten()
    y_true = (y_true > 0.5).astype(np.float32)  # Ensure y_true is
binary

    y_pred = valid_preds.flatten()

    # Calculate metrics for this fold
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    auc = roc_auc_score(y_true, y_pred)

    # Store the metrics
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)
    auc_scores.append(auc)
    print(f"Fold {fold + 1} - Precision: {precision}, Recall: {recall}, 
F1: {f1}, AUC: {auc}")

# Calculate the mean and standard deviation of the metrics across all 
folds
print("\nCross-Validation Results:")
print(f"Precision: {np.mean(precision_scores)} ± 
{np.std(precision_scores)}")
print(f"Recall: {np.mean(recall_scores)} ± {np.std(recall_scores)}")
print(f"F1 Score: {np.mean(f1_scores)} ± {np.std(f1_scores)}")
print(f"AUC: {np.mean(auc_scores)} ± {np.std(auc_scores)}")
```