ENHANCED CLASSIFICATION AND DETECTION OF BRAIN TUMOR USING HYBRID DEEP LEARNING AND MACHINE LEARNING MODELS

A Project Report submitted in the partial fulfillment of
the Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

D. Bala Sri Abhigna (21471A0519)
SK. Sameera (21471A0555)
CH. Harini (21471A0516)

Under the esteemed guidance of

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF rank
in the band of 201-300 and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "ENHANCED CLASSIFICATION AND DETECTION OF BRAIN TUMOR USING HYBRID DEEP LEARNING AND MACHINE LERANING MODELS" is a bonafide work done by the team D. Bala Sri Abhigna (21471A0519), SK. Sameera (21471A0555), CH. Harini (21471A0516) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Dr. Sireesha Moturi, B.Tech., M.Tech., Ph.D.
Associate Professor

PROJECT CO-ORDINATOR

D.Venkata Reddy, B.Tech., M.Tech., (Ph.D).
Assistant Professor

HEAD OF THE DEPARTMENT

Dr. S. N. Tirumala Rao, M.Tech., Ph.D.
Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**ENHANCED CLASSIFICATION AND DETECTION OF BRAIN TUMOR USING HYBRID DEEP LEARNING AND MACHINE LEARNING MODELS** " is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

D. Bala Sri Abhigna (21471A0519)
SK. Sameera (21471A0555)
CH. Harini (21471A0516)

ACKNOWLEDGEMENT

We wish to express my thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao, B.Sc.,** who took keen interest in us in every effort throughout this course. We owe our sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of CSE department and also to our guide **Dr. Moturi Sireesha, B.Tech, M.Tech., Ph.D.,** professor of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks to **D.Venkat Reddy B.Tech, M.Tech.,(Ph.D.),** Assistant professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant sourceof inspiration for us throughout this project work.

We extend our sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions clarified our doubts, which really helped us in successfully completing our project.

By

D. Bala Sri Abhigna (21471A0519)

SK. Sameera (21471A0555)

CH. Harini (21471A0516)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations**.**

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice**.**

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.
CO421.2: Identify and classify the requirements.
CO421.3: Review the Related Literature
CO421.4: Design and Modularize the project
CO421.5: Construct, Integrate, Test and Implement the Project.
CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop model for detection and classification of Brain Tumor in MRI Scans using CNN-SVM model	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement i critically analyzed, the process mode is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the project will be handled by the social media users and in future updates in our project can be done based on detection for Brain Tumor	PO4, PO7
C32SC4.3	The physical design includes website to check Brain umor in MRI scans	PO5, PO6

ABSTRACT

In recent years, Deep Learning has become an influential tool in medical imaging,especially the present work illustrates a new technique for brain tumor identification based on magnetic resonance images(MRI) using the combination of convolutional neural networks and support vector machine techniques. The proposed model combines Convolutional Neural Networks (CNNs) performing feature extraction effectively, while Support Vector Machines (SVMs) classify with high precision. To begin with, some preprocessing methods like adaptive gamma correction, adaptive contrast enhancement, and median filtering are applied to MRI images so as to improve image quality and minimize noise interference. Fuzzy c-means clustering technique is used to extract important texture features like energy, mean values or entropy from Gray Level Co-occurance Matrixs (GLCM) represented as matrices of grey levels co-occurrence pairs. CNN helps with acquiring deeper features from segmented images which serve as inputs during training phases of SVM classifiers based on those features thus obtained from them (CNN), eventually yielding high performance levels. An hybrid model using CNN-SVM showed an accuracy rate of 97 .94%, sensitivity 95% , specificity 98.1% when it came to distinguishing between normal and abnormal brain tissue. Therefore, this hybrid model demonstrates its capability in tumor detection due to combination of both CNN and SVM hence providing an alternative automated approach for early diagnosis.

INDEX

S.NO	CONTENT	PAGE NO
1	INTRODUCTION	1
	1.1 MOTIVATION	4
	1.2 PROBLEM STATEMENT	5
	1.3 OBJECTIVE	6
2	LITERATURE SURVEY	7
3	SYSTEM ANALYSIS	
	3.1 EXISTING SYSTEM	12
	3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM	14
	3.2 PROPOSED SYSTEM	15
	3.3 FEASIBILITY STUDY	17
	3.4 USING COCOMO MODEL	18
4	SYSTEM REQUIREMENTS	
	4.1 SOFTWARE REQUIREMENTS	20
	4.2 REQUIREMENT ANALYSIS	20
	4.3 HARDWARE REQUIREMENTS	21
	4.4 SOFTWARE	21
	4.5 SOFTWARE DESCRIPTION	22
5	SYSTEM DESIGN	
	5.1 SYSTEM ARCHITECTURE	23
	5.1.1 DATASET	24
	5.1.2 DATA PREPROCESSING	27
	5.1.3 FEATURE EXTRACTION	28
	5.1.4 MODEL BUILDING	29
	5.1.5 CLASSIFICATION	34
	5.2 MODULES	38
	5.3 UML DIAGRAMS	41
6	IMPLEMENTATION	

	6.1 MODEL IMPLEMENTATION	45
	6.2 CODING	49
7	TESTING	
	7.1 UNIT TESTING	69
	7.2 INTEGRATION TESTING	70
	7.3 SYSTEM TESTING	73
8	RESULT ANALYSIS	77
9	OUTPUT SCREENS	83
10	CONCLUSION	86
11	FUTURE SCOPE	87
12	REFERENCES	88

LIST OF FIGURES

S.NO	FIGURE DESCRIPTION	PAGE NO
1	FIG 1.1 CLASSIFICATION OF HEALTHY BRAIN TISSUE AND TUMOR BRAIN TISSUE	3
2	FIG 3.1 FLOW CHART OF EXISTING SYSTEM FOR BRAIN TUMOR	13
3	FIG 3.2 FLOWCHART OF PROPOSED SYSTEM	15
4	FIG 5.1 DIFFERENT TUMOR CLASSES DATA SET IMAGES	26
5	FIG 5.2 IMAGE AFTER APPLYING THE PREPROCESSING TECHNIQUE	28
6	FIG 5.3 MRI SCAN AFTER APPLYING GLCM FEATURE EXTRACTION	28
7	FIG 5.4 CNN MODEL ARCHITECTURE	31
8	FIG 5.5 CNN-SVM ARCHITECTURE	32
9	FIG 5.6 CLASSIFICATION OF BRAIN TUMORS	35
10	FIG 5.7 DESIGN OVERVIEW	42
11	FIG 5.8 UML DIAGRAM FOR BRAIN TUMOR DETECTION AND CLASSIFICATION	43
12	FIG 7.1 STATUS TUMOR DETECTED	74
13	FIG 7.2 STATUS NO TUMOR DETECTED	75
14	FIG 7.3 STATUS INVALID IMAGE	76
15	FIG 8.1 ACCURACY COMPARISON ON DIFFERENT MODELS	77
16	FIG 8.2 SENSITIVITY COMPARISON ON DIFFERENT MODELS	78
17	FIG 8.3 SPECIFICITY COMPARISON ON DIFFERENT MODELS	79
18	FIG 8.4 JACCARD COEFFICIENT ON DIFFERENT MODELS	79
19	FIG 8.5 SIMULATED CONFUSION MATRIX FOR CNN-SVM MODEL	80
20	FIG 8.6 CONFUSION MATRIX FOR VGG AND ANN MODELS	81
21	FIG 8.6 CONFUSION MATRIX FOR RFC AND RNNS MODELS	81
22	FIG 8.6 CONFUSION MATRIX FOR FCNNS AND CNN-SVM MODELS	81
23	FIG 9.1 HOME PAGE	83
24	FIG 9.2 ABOUT PAGE	84

25 FIG 9.3 PROJECT SCREEN 84
26 FIG 9.4 MODEL EVALUATION SCREEN 85
27 FIG 9.5 PROJECT FLOWCHART SCREEN 85

List of Tables

S.NO CONTENT PAGE NO

1 TABLE 1. DATASET DESCRIPTION 25
2 TABLE 2. MODEL PERFORMANCE COMPARISON 82

1. INTRODUCTION

Brain tumors are among the most severe and life-threatening forms of cancer, demanding immediate and accurate detection to enable effective treatment. These tumors pose a significant challenge to healthcare due to their diverse types, complex structures, and varying impacts based on size, location, and aggressiveness. Early detection and precise classification are critical to improving survival rates and guiding therapeutic interventions. Magnetic Resonance Imaging (MRI) plays a pivotal role in detecting brain tumors, offering unparalleled imaging quality and precision compared to other modalities. However, relying solely on manual interpretation by radiologists can be time-intensive and prone to inconsistencies, leading to delays or potential diagnostic errors. These limitations underscore the need for automated systems capable of achieving accurate and efficient brain tumor detection and classification.

Brain tumors are a significant health challenge in India, affecting thousands of people every year. These tumors can be either benign (non-cancerous) or malignant (cancerous) [1] as shown in Fig 1.1 and impact individuals of all age groups, including children. The increasing prevalence of brain tumors in India is attributed to various factors such as lifestyle changes, environmental exposure, genetic predisposition, and improved diagnostic capabilities. Despite medical advancements, brain tumor awareness remains low, leading to delayed diagnosis and treatment.

According to medical reports, brain tumors account for about 2% of all cancers in India. Studies estimate that over 40,000 to 50,000 new cases are reported annually, with a rising trend in both adults and children. Among childhood cancers, 20% are brain tumors, making them one of the most common pediatric malignancies. The most aggressive and frequently diagnosed malignant brain tumor in India is Glioblastoma Multiforme (GBM), known for its rapid progression and poor prognosis.

One of the biggest challenges in managing brain tumors in India is the lack of early diagnosis. Many patients do not recognize the symptoms or seek medical attention only when the tumor has reached an advanced stage. The symptoms of a brain tumor vary but commonly include persistent headaches, seizures, blurred vision, memory loss, difficulty in speech, and weakness in limbs.

Another major challenge is the high cost of treatment. Brain tumor surgeries, chemotherapy, and radiation therapy require specialized infrastructure and skilled professionals, making them expensive. While top hospitals in metropolitan cities offer world-class neurosurgical facilities, rural and semi-urban areas lack access to advanced treatment. Additionally, India faces a shortage of neurosurgeons and neuro-oncologists, further delaying critical interventions.

In recent years, Deep Learning and Machine Learning techniques have transformed medical imaging by introducing automated and efficient diagnostic tools. Convolutional Neural Networks (CNNs) [2] are particularly effective in medical image analysis, as they excel at learning hierarchical features from data, enabling high precision in image classification tasks. On the other hand, Support Vector Machines (SVMs) are renowned for their robustness in classification problems, offering high generalization capabilities and excellent performance in binary and multi-class classification tasks. By integrating these two technologies, hybrid models combine the feature extraction strength of CNNs with the precise decision-making power of SVMs, creating a robust solution for brain tumor detection.

The Indian government has also taken steps to make brain tumor treatment more accessible. The Ayushman Bharat Health Scheme provides financial assistance to low-income patients, making expensive surgeries and therapies more affordable. Increased public awareness campaigns and collaborations with global medical research organizations are further strengthening India's fight against brain tumors.

This research proposes an innovative hybrid model that integrates CNN and SVM for brain tumor recognition and classification. The model leverages several advanced preprocessing techniques to enhance the quality of MRI images, ensuring improved analysis and accuracy.

These preprocessing methods include the Adaptive Contrast Enhancement Algorithm (ACEA) for localized contrast optimization, adaptive gamma correction for brightness enhancement, normalization to standardize data, equalization for improving image contrast, median filtering for noise reduction, and data augmentation to expand and diversify the dataset. Together, these techniques prepare the images for effective segmentation and feature extraction. For segmentation, the model employs the Fuzzy C-Means (FCM) clustering technique, which identifies tumor regions with precision by clustering pixels based on intensity similarities. Following segmentation,

texture-based features such as contrast, energy, correlation, and homogeneity are extracted using the Gray-Level Co-occurrence Matrix (GLCM)[3]. These features provide valuable information about the spatial relationships of pixel intensities in the tumor region, aiding in accurate classification.

The hybrid CNN-SVM model [3] is designed to achieve higher accuracy and reliability in detecting and classifying brain tumors. CNNs are utilized to extract deep features from segmented MRI images, capturing complex patterns and details that traditional methods might miss. These features are then input into an SVM classifier, which efficiently separates normal and abnormal brain tissues. The combination of these technologies enables the model to achieve superior performance compared to standalone approaches.

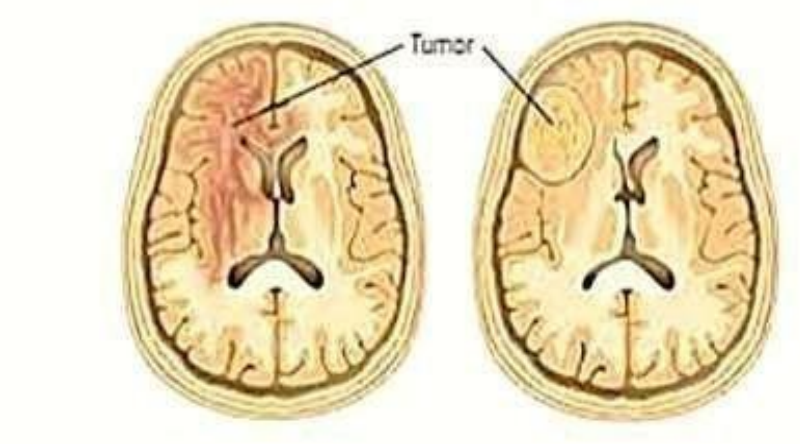


FIG 1. 1 CLASSIFICATION OF HEALTHY BRAIN TISSUE AND TUMOR BRAIN TISSUE

With its ability to process MRI images with high precision, the proposed hybrid model has the potential to significantly enhance diagnostic accuracy, reduce the time required for tumor detection, and improve patient outcomes. Moreover, the model's scalability and efficiency make it a promising candidate for integration into clinical workflows, providing healthcare professionals with a powerful tool for personalized treatment planning. As advancements in artificial intelligence continue, such hybrid approaches are likely to become indispensable in the fight against brain tumors, offering hope for early diagnosis and improved survival rates.

1.1 Motivation

Brain tumors represent one of the most challenging and life-threatening medical conditions, with a significant impact on both patients and healthcare systems globally. Timely and accurate diagnosis is crucial for effective treatment and management of brain tumors. However, the traditional methods of detecting and classifying brain tumors, such as manual radiological analysis, can be time-consuming, subjective, and prone to errors, particularly in the case of complex and subtle tumor patterns.

The increasing reliance on medical imaging, especially MRI scans, has paved the way for more automated approaches to brain tumor detection. MRI scans provide detailed images of the brain, allowing doctors to detect tumors early, but interpreting these images requires expertise and can still result in delayed diagnoses or misclassification. This is where the application of Deep Learning and machine learning techniques can revolutionize the field.

By developing a robust model for automated brain tumor detection and classification, the goal of this project is to reduce the diagnostic time, eliminate human errors, and assist healthcare professionals in making more accurate decisions. Using a Convolutional Neural Network (CNN) combined with a Support Vector Machine (SVM), this project aims to provide an efficient and scalable solution for detecting tumors in brain MRI images. The model will be able to classify scans into tumor and non-tumor categories, helping to prioritize cases that require immediate medical attention and reducing the overall burden on healthcare systems.

Furthermore, integrating this model into a user-friendly web application can facilitate its accessibility, allowing doctors, clinicians, and even patients to quickly upload MRI images and receive diagnostic feedback. The ultimate aim is to improve early detection rates, reduce healthcare costs, and enhance patient outcomes, ultimately contributing to the ongoing fight against brain cancer and other neurological disorders.

1.2 Problem Statement

Brain tumors quite serious, could cause paralysis and long term treatments. Their implications vary based on their size, location and type. Cerebral neoplasms have life-threatening consequences due to disabilities that are so serious requiring harsh drugs or surgeries just to curb pain and not treat them whatsoever.

Brain damage consequence from any malignant growth usually depends on factors like its size; where it's located within our system; what makes up this invasive agent-these are the things distinguishing one region from another (i.e., responsible for movement). An effective measure against some impairments would be early diagnosis because once they start appearing; you cannot go back anymore! Nevertheless, though challenging enough to achieve, accurately classifying brain tumors is challenging because there exists vast diversity among them regarding volume ratio or lightness color differences arising from mere instinct alone! Henceforth since some kinds of malignant growths show certain external resemblances, it becomes hard to classify them properly. There are serious consequences of having a tumor such as disabilities that may be severe and can lead to long-term treatments which are painful with the aim of either curing or alleviating one's impairment. The possibility for normal function depends on several factors, including size and location (i.e., on the surface) as well as the type of the tumor. For example, specific areas of the brain that control limb movements could get compressed by tumors making it impossible to move around at all. Early diagnosis is important in order to prevent or minimize such disabilities from occurring. However, the fact that brain tumors can be of different types and sizes whose shapes also differ makes it difficult to classify them accurately in terms of pathology. To overcome these barriers to accurate diagnosis and treatment, which would consequently enhance patients' outcomes and improve their lives quality wise, it is essential that we have effective tools for diagnosing them promptly. Early diagnosis prevents complications. There is therefore a need for better instruments for quick, precise diagnosis to facilitate prompt treatment with improved out comes for patients.

1.3 Objective

The primary objective of the Brain Tumor Detection and Classification project is to develop an automated system capable of accurately detecting and classifying brain tumors from MRI images. The project aims to build a robust model that classifies scans into tumor and non-tumor categories, leveraging a hybrid approach that combines Convolutional Neural Networks (CNN) for feature extraction and Support Vector Machines (SVM) for classification. Optimizing the model's performance by applying advanced preprocessing techniques is a key goal to ensure high accuracy, precision, and reliability.

Additionally, the project focuses on developing a user- friendly web application using Python Flask, enabling users to easily upload MRI images and receive instant classification results. To maintain the system's integrity, image validation mechanisms will be integrated to ensure that only valid brain MRI images are accepted, with clear error messages provided for invalid uploads. The project also seeks to support healthcare professionals by offering a reliable tool that enhances clinical decision-making, reduces diagnostic time, and minimizes human error. Furthermore, the system is designed to be scalable and adaptable for future enhancements, promoting early detection and timely medical intervention to improve patient outcomes.

The system includes strict image validation mechanisms to accept only valid brain MRI images. Invalid uploads trigger clear error messages, ensuring users are informed and can correct mistakes.This project supports healthcare professionals by enhancing clinical decision-making, reducing diagnostic time, and minimizing human error. It promotes early detection and timely medical intervention, improving patient outcomes.

Designed for scalability, the system can be expanded to classify different types of brain tumors, integrate with electronic health records, and include real-time feedback for continuous improvement.Overall, the project leverages artificial intelligence to advance medical diagnostics, offering a meaningful contribution to healthcare and enhancing patient care.

2. LITERATURE SURVEY

Brain tumor detection and classification using Deep Learning have been extensively studied in recent years due to the increasing need for accurate and automated diagnostic solutions. Traditional Machine Learning techniques, such as Support Vector Machines (SVMs) and Random Forest classifiers (RFs), have been widely used for tumor identification from MRI scans. However, these methods often struggle with feature extraction and automation challenges. In contrast, Deep Learning approaches, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable improvements in classification accuracy by effectively capturing spatial features in medical images.

With the advent of transfer learning and advanced CNN architectures, researchers have developed innovative models that significantly enhance tumor detection performance. Transfer learning allows pre-trained models to adapt to new medical datasets, overcoming issues related to limited labeled MRI data. Furthermore, multi-modal fusion techniques and preprocessing strategies have emerged as effective solutions for refining image quality and optimizing feature extraction. The following studies highlight key advancements in Deep Learning for brain tumor detection.

Solanki S. [1] addresses techniques for identifying and classifying brain tumors using CNNs and transfer learning models, including RESNET-100 and VGGNET. Traditional methodologies like SVMs and Random Forest Classifiers (RFs) present challenges in automation, whereas CNNs enhance classification accuracy by a margin of 10-15%. Furthermore, transfer learning significantly boosts performance, adding an additional 5-10% accuracy to the classification outcomes.

M. Imran [4] proposed an enhanced CNN-based detection method utilizing a Deep Convolutional Neural Network (DCNN) combined with a three-stage preprocessing technique. This approach demonstrated improved results when evaluated against models like VGG16 and VGG19. The study highlights the effectiveness of advanced preprocessing steps in enhancing the accuracy of CNN models for brain tumor detection.

Ahmad S. and Choudhury P.K. [5] systematically reviewed the progress in Machine Learning and Deep Learning for tumor identification from MRI images. Their findings revealed that pretrained models such as Inception-v3 and DenseNet201 can achieve accuracy levels exceeding 90%. However, optimization challenges, including fine-tuning and computational resource requirements, remain significant barriers to further improvements.

To address limitations in single-modal imaging techniques, Shah H. A. [6] introduced a CNN designed for multi-modal MRI fusion. This approach integrates data from different MRI modalities, improving the model's robustness and accuracy in tumor detection. Additionally, Efficient Net-B0, as noted in [8], is regarded as one of the top-performing models for tumor detection. Its performance is further enhanced by employing transfer learning and data augmentation methods, particularly useful for addressing challenges posed by limited datasets.

M. Agrawal proposed a brain tumor detection and classification approach using Deep Learning models, specifically leveraging the Inception V3 architecture for superior results [9]. Similarly, Sireesha M. demonstrated the effectiveness of CNNs in tumor classification, utilizing magnetic resonance images to achieve high accuracy. This study underscores the importance of CNNs in advancing the precision and efficiency of tumor detection systems [10].

Another noteworthy method is the Faster R-CNN [11], which employs region proposal networks for real-time tumor detection in MRI images. This technique stands out for its high precision, enabling rapid and accurate tumor identification while reducing the time required for diagnosis in the imaging process.

Radhi, A. A.[12] presented an efficient algorithm for the detection of brain tumors from MRI images. The proposed approach enhances the accuracy and speed of detection, providing significant contributions to automated diagnostic systems. The study emphasized the importance of preprocessing and segmentation techniques to improve detection performance. Toğaçar, M., Ergen, B., & Cömert, Z. introduced BrainMRNet, a novel convolutional neural network (CNN)[13] model specifically designed for brain tumor detection using magnetic resonance images (MRI). The model demonstrates high accuracy and robustness in classification tasks, utilizing advanced layers and optimization techniques to enhance detection capabilities. Their

research also emphasized the role of data augmentation in boosting model performance .

Sharma, K., Kaur, A., & Gujral, S.[14] explored machine learning algorithms for brain tumor detection. Their research emphasizes the potential of various ML algorithms, including Support Vector Machines (SVM) and Random Forests, in enhancing detection accuracy and reliability. The study also focused on feature selection techniques to optimize the classification process . Moturi, S., Tirumala Rao, S. N., & Vemuru, S.[15] proposed an optimized feature extraction and hybrid classification model for predicting heart disease and breast cancer. Although not specific to brain tumors, their methodology involving ensemble learning and feature optimization provides insights into hybrid models that could be adapted for brain tumor detection, particularly in combining multiple classifiers for improved accuracy.

Islam, M. K., Ali, M. S., Miah, M. S., Rahman, M. M., Alam, M. S., & Hossain, M. A. proposed a detection technique for brain tumors using superpixels, principal component analysis (PCA), and a template-based K-means clustering algorithm. Their approach contributes to improving the accuracy and efficiency of tumor segmentation and classification. The study emphasized the importance of reducing computational complexity while maintaining high accuracy [17].Amin, J., Sharif, M., Yasmin, M., & Fernandes, S. L. introduced a distinctive approach for brain tumor detection and classification using MRI, focusing on enhancing detection accuracy through novel preprocessing and classification methods. Their research explored advanced filtering techniques and feature extraction methods to enhance classification accuracy [18].

Abiwinanda, N., Hanif, M., Hesaputra, S. T., Handayani, A., & Mengko, T. R. [19]focused on brain tumor classification using CNNs, presenting an efficient framework that improves classification performance in medical image analysis. The study incorporated dropout layers and batch normalization to enhance model generalization .Setha, J., & Raja, S. S.[20] explored brain tumor classification using convolutional neural networks, demonstrating the effectiveness of deep learning in medical image classification and its potential to aid radiologists in diagnosis. Their research also highlighted the impact of transfer learning in improving model performance on limited datasets.

Hassan Ali Khan examines a CNN approach for differentiation of brain tumors in MRI images. An in-house CNN [21] model is compared with VGG-16, ResNet-50 and Inception-v3, which are pre-trained models. The proposed model , sur-passing the other models in accuracy but consume less power. This clearly shows the capability of deep learning making medical images efficient and accurate.

This technology [22] has changed the game for brain tumor classification; MRI images that were previously manually analyzed for imaging features extraction were traditionally obtained through imager techniques such as ultrasound. Now that a new

model has been developed, such as CNN, VGG-16, ResNet-50, these models are likely to improve accuracies even more than above.

Differential Deep-CNN [22] provides accuracy improvement of feature extrac-tion and achieves up to 99.25% accuracy. In addition, multi-modal (T1, T2, FLAIR) MRI data has been used to develop hybrid models such as CNN-SVM to advance detection. When combined with transfer learning and data augmenta-tion, the level of performance rises even higher.

This new research on MRI images [24] for brain-tumor classification seeks to address the need for high accuracy and efficiency. The traditional modeling ap-proaches used manual feature extraction, whereas using CNN models such as VGG16, VGG19, or ResNet-50 gives an automated approach for tumor detection with high precision. Muhammad Attique Khan [24] proposed a method for mul-timodal classification incorporating MRI sequences (T1, T2, T1CE, and FLAIR), feature selections, and fusion methods; with the algorithms yielding 97.8% accu-racy. Transfer learning and hybrid models (CNN-SVM, ELM, etc.) have im-proved classification performance.

The amazing accuracy achieved is 99.83% with an outstanding performance on the ResNet101-CWAM [25] model, as compared to earlier deep learning tech-niques. Attention mechanisms-also referred to as channel-wise and spatial atten-tion-augment feature extraction, thereby enhancing the classification accuracy. Transfer learning and hybrid architectures such as CNN-SVM and 3D-CNNs also increased the detection efficiency.

An innovative approach combines a soft-attention mechanism [26], which ramp up feature selection by giving priority to the most clinically relevant information, and a four-layer convolutional neural network architecture, yielding robust tumor classification performance. Probing through comparative studies shows that this method outshines existing state-of-the-art models making it more hopeful to-wards precocious and accurate brain tumor diagnosis.

A standout feature of Deep Learning is its ability to perform automatic feature extraction. In medical imaging, for example, convolutional neural networks (CNNs) [3] can efficiently analyze MRI scans, identifying intricate patterns to distinguish between normal and abnormal tissues with remarkable precision. Other advanced Deep Learning models, such as Recurrent Neural Networks (RNNs)[15], Transformers, and Generative Adversarial Networks (Gans), further broaden its application scope. These architectures excel in sequential data analysis, complex dependency modeling, and synthetic data creation.

One of Deep Learning's greatest strengths lies in its scalability. As the availability of data and computational resources grows, Deep Learning models demonstrate improved accuracy and performance. Moreover, techniques like transfer learning, where pre-trained models are fine-tuned for specific tasks, and data augmentation , which increases dataset diversity, make Deep Learning accessible and effective even when data is limited. This adaptability and robustness solidify Deep Learninsg as a cornerstone of modern artificial intelligence.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Brain tumor detection and classification have traditionally relied on manual analysis of Magnetic Resonance Imaging (MRI) scans by radiologists and neurologists. This manual approach, while effective, is time-consuming and highly dependent on the expertise of medical professionals. Moreover, human interpretation is prone to errors, leading to misdiagnosis or delayed treatment. To address these challenges, computational methods have been developed, ranging from traditional Machine Learning techniques to advanced Deep Learning models.

Early Machine Learning (ML) approaches, such as Support Vector Machines (SVMs), Random Forest (RF)[1], k-Nearest Neighbors (k-NN), and Decision Trees, focused on extracting handcrafted features from MRI images. These methods required extensive preprocessing and manual selection of features such as texture, intensity, and shape descriptors. Although these models provided reasonable accuracy (around 70-85%), their performance was limited by feature dependency, poor generalization to different datasets, and an inability to handle complex variations in tumor structures. The introduction of Convolutional Neural Networks (CNNs) revolutionized brain tumor detection by automating feature extraction and improving classification accuracy. CNN architectures such as Alex Net, VGG16[1,4,24], ResNet[1], and DenseNet have been widely used in medical image analysis, achieving accuracy levels of 85-95%. However, CNNs come with challenges such as the need for large labeled datasets, high computational requirements, and potential overfitting when trained on small datasets. To overcome data limitations, transfer learning has been widely adopted, where pretrained models like Inception-V3[5], ResNet-50[1], and EfficientNet-B0[6] are fine-tuned for brain tumor classification. This approach not only reduces training time but also enhances accuracy by an additional 5- 10%.

In addition to CNN-based models, hybrid approaches combining CNN with Support Vector Machines have shown promising results. In this technique, CNNs extract deep features from MRI scans, while SVMs handle the final classification, leading to improved performance..

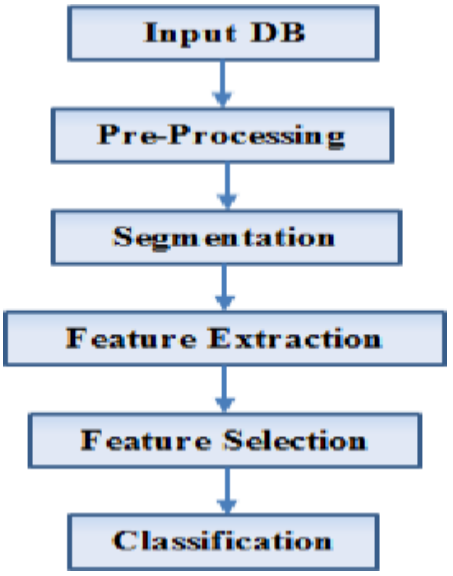


FIG 3.1. FLOW CHART OF EXISTING SYSTEM FOR BRAIN TUMOR CLASSIFICATION

This flowchart Fig 3.1 outlines a typical image processing pipeline for brain tumor detection and classification from medical images, such as MRI scans. The process begins with an input database of medical images, which undergoes pre-processing to improve their quality. Pre-processing techniques include noise reduction to eliminate artifacts, image enhancement to amplify critical details, and bias field correction to address intensity inhomogeneities. These steps ensure that the images are uniform and ready for subsequent analysis. Once the images are pre-processed, the segmentation phase isolates the region of interest (ROI) that potentially contains the tumor. Segmentation techniques vary from traditional methods like thresholding, region growing, and clustering to advanced approaches such as active contours and Deep Learning-based models, which offer higher accuracy and adaptability. The segmentation phase is critical, as it focuses on delineating the tumor area, forming the basis for the following steps in the pipeline.

After segmentation, feature extraction derives key characteristics like shape, texture, and intensity of the tumor. Feature selection methods, such as Principal Component Analysis (PCA), reduce dimensionality and computational complexity while preventing overfitting. Finally, classifiers like SVM, k-NN[2], decision trees, or CNNs[1] categorize the tumor as benign, malignant, or normal tissue. This automated pipeline improves accuracy and assists medical professionals in diagnosis.

3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM FOR BRAIN TUMOR CLASSIFICATION

Despite significant advancements in automated brain tumor detection and classification, the existing systems face several limitations:

- **Dependence on Large Labeled Datasets:**

Deep Learning models, particularly CNNs, require extensive labeled datasets for effective training. However, obtaining large-scale annotated medical images is challenging due to data privacy concerns and the need for expert annotations**.**

- **Computational Complexity:**

CNN-based models, especially deep architectures like ResNet, VGG16, and DenseNet, demand high computational power and memory. This makes them less suitable for deployment in resource-constrained environments, such as smaller medical facilities.

- **Overfitting on Small Datasets**

Due to the limited availability of MRI datasets, deep learning models often overfit, meaning they perform well on training data but fail to generalize to unseen data. While transfer learning helps mitigate this issue, it does not completely eliminate it.

- **Lack of Robustness in Tumor Segmentation**

Segmentation techniques, whether traditional (thresholding, region growing) or deep learning-based, struggle with variations in tumor shape, size, and intensity. Poor segmentation accuracy negatively impacts the subsequent classification stage.

- **Sensitivity to Image Quality and Artifacts**

MRI images can be affected by noise, intensity inhomogeneities, and artifacts. If preprocessing techniques such as bias field correction and noise reduction fail to fully rectify these issues, the overall system performance declines.

- **Limited Generalization Across Different MRI Modalities**

While multi-modal MRI fusion has shown promise, most existing systems are designed to work with a specific type of MRI sequence. The lack of adaptability to various MRI modalities (T1-weighted, T2-weighted, FLAIR) reduces their clinical applicability.

- **Difficulty in Handling Class Imbalance**

Most publicly available MRI datasets contain an imbalance between different tumor types, leading to biased model predictions.

3.2 PROPOSED SYSTEM

The proposed model utilizes a hybrid approach combining Convolutional Neural Networks (CNN)[22] and Support Vector Machines (SVM)[1] to improve the accuracy and reliability of brain tumor detection and classification from MRI images. This integration leverages the strengths of both Deep Learning and traditional Machine Learning techniques. CNN is employed for its exceptional ability to extract hierarchical and detailed features from images, while SVM is used for its precision in classification tasks, particularly in distinguishing between tumor categories.

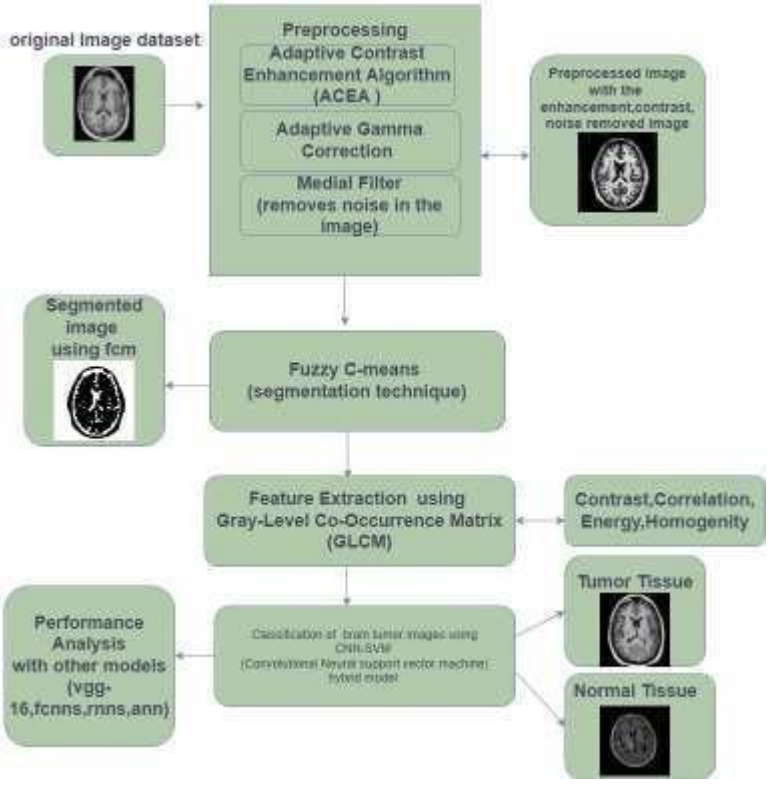


FIG 3.2. FLOW CHART OF PROPOSED SYSTEM

The workflow Fig 3.2 begins with preprocessing MRI images to enhance their quality. Techniques like Adaptive Gamma Correction[2] and Adaptive Contrast Enhancement optimize the brightness and contrast of the images, while Median Filtering removes noise. These steps ensure that the images are clean and suitable for further processing.

After preprocessing, segmentation is conducted using Fuzzy C-Means [13] clustering. This technique identifies and isolates the tumor region with high precision by considering the natural fuzziness in data, which makes it more effective than traditional hard classification methods. Following segmentation, features such as texture and spatial relationships are extracted using the Grey Level Co-occurrence Matrix (GLCM), which provides crucial information for differentiating tumor types. The extracted features are then passed through a CNN to capture deep and intricate patterns from the segmented images.

The CNN acts as an advanced feature extractor, transforming the data into a format suitable for classification. These features are subsequently fed into an SVM classifier, which categorizes the data into classes such as healthy tissue, benign tumor, or malignant tumor. The SVMs ability to establish clear decision boundaries enhances the overall classification accuracy of the model.

Advantages over exsisting system:

1. **High Accuracy and Precision:** Combines CNN's feature extraction capabilities with SVMs robust classification mechanism. Ensures more reliable tumor detection and classification.

2. **Enhanced Preprocessing:** Adaptive algorithms improve the quality of MRI scans, allowing for better analysis and feature extraction.

3. **Effective Segmentation:** Fuzzy C-Means clustering provides a more accurate delineation of the tumor region compared to conventional methods.

4. **Reduced Overfitting:** Feature selection techniques, such as GLCM[24], focus on the most relevant features, enhancing the model's generalization capability.

5. **Superior Performance Metrics:** High sensitivity and specificity ensure reliable differentiation between normal and diseased tissues, reducing false negatives and positives.

6. **Automated and Scalable:** The model automates brain tumor detection, aiding radiologists in faster and more consistent diagnoses. Adaptable to other medical imaging tasks with minimal changes

3.3 FEASIBILITY STUDY

The integration of Convolutional Neural Networks (CNNs) with Support Vector Machines (SVMs) offers a promising hybrid approach for brain tumor detection and classification. Below is a detailed feasibility analysis considering technical, operational, and economic aspects.

1. Technical Feasibility

- **Automated Feature Extraction:**
  CNNs excel in automatically extracting deep, high-dimensional features from MRI images, reducing the need for manual feature engineering. These extracted features are highly representative and enhance classification accuracy.
- **Enhanced Classification with SVM:**
  While CNNs are powerful for feature extraction, SVMs are effective classifiers for high- dimensional data, especially in scenarios with limited data samples. The combination leverages the strength of CNNs for feature extraction and the robustness of SVMs for classification.
- **Improved Accuracy and Generalization:**
  The hybrid CNN-SVM approach reduces the risk of overfitting, especially in small datasets. The SVM layer acts as a strong classifier that enhances generalization and improves accuracy by effectively handling complex decision boundaries.
- **Scalability:**
  The model can be scaled for different MRI modalities (T1, T2, FLAIR) and tumor types, provided sufficient data and proper preprocessing techniques are used.
- **Integration with Pretrained Models:**
  Transfer learning can be applied by using pretrained CNN models (like ResNet-50, VGG16) for feature extraction. This reduces computational requirements and training time while maintaining high accuracy.

2. Operational Feasibility

- **Ease of Deployment:**
  The CNN-SVM model can be integrated into existing medical imaging systems and deployed through web or desktop applications, enhancing its operational viability in clinical settings.

- **Interpretability:**

While CNNs are often considered "black boxes," integrating an SVM classifier allows for better interpretability, as SVMs provide clearer decision boundaries. This can increase trust and acceptance among medical professionals.

- **Data Handling:**

The model requires labeled MRI images for training but is flexible enough to handle varying input sizes and formats with appropriate preprocessing. Multi-modal data can be processed to improve classification accuracy.

- **Maintenance and Upgradation:**

The system can be easily updated with new data to retrain the SVM component, improving performance without extensive retraining of the entire CNN model.

3. Economic Feasibility

- **Cost-Effective Training:**
  Using transfer learning significantly reduces computational costs, as only the SVM component or final layers of the CNN need to be fine-tuned.
- **Resource Optimization:**
  Since the CNN handles feature extraction and SVM performs classification, computational resources are utilized efficiently. This hybrid approach can be run on standard GPU systems, reducing infrastructure costs.
- **Reduced Diagnostic Costs:**
  By automating tumor detection, the model can help reduce manual diagnostic time, leading to faster treatment planning and potentially lowering overall healthcare costs.
- **Long-Term Investment:**
  Although initial development and integration costs may be high, the long-term benefits of improved diagnostic accuracy and efficiency justify the investment. The system's ability to adapt to new data and conditions ensures sustainable benefits.

3.4 USING COCOMO MODEL

The COCOMO (Constructive Cost Model) is a widely used estimation technique in software engineering that helps predict the effort, development time, and team size required for a project. Applying the Basic COCOMO Model to the MRI brain tumor

detection project provides a structured approach to estimating project requirements. Given the complexity and scope of the project—developing a machine learning-based web application using Python, Flask, and TensorFlow—the project falls under the Semi-Detached category. This category is characterized by moderate complexity, involving a mix of experienced and less experienced team members.

The Basic COCOMO model relies on three key formulas for estimating effort, development time, and the number of people required:

- Effort (E) = $a \times (\text{KLOC})^b$ (measured in Person-Months)
- Development Time (T) = $c \times (\text{E})^d$ (measured in Months)
- People Required (P) = E / T

In these formulas, KLOC refers to the estimated number of lines of code in thousands, and the constants (a, b, c, d) vary based on the project type. For Semi-Detached projects, the constants are a = 3.0, b = 1.12, c = 2.5, and d = 0.35.

Considering the entire project, including backend development, frontend design, and machine learning model implementation, the estimated size of the code is 10,000 lines of code, equivalent to 10 KLOC. Using the formulas, the estimated effort can be calculated as:

E=3.0×(10)1.12=3.0×13.18≈39.54 Person-MonthsE = 3.0 \times (10)^{1.12} = 3.0 \times 13.18 \approx 39.54 \text{ Person-Months}

The development time is calculated as:

T=2.5×(39.54)0.35≈2.5×4.23≈10.58 MonthsT = 2.5 \times (39.54)^{0.35} \approx 2.5 \times 4.23 \approx 10.58 \text{ Months}

Finally, the required number of people for the project is estimated using:

P=39.5410.58≈3.74≈4 PeopleP = \frac{39.54}{10.58} \approx 3.74 \approx 4 \text{ People}

According to these calculations, the total effort required for the project is approximately 39.54 person-months, with an estimated development time of around 10.6 months. The project would ideally require a team of 4 members, which may include machine learning engineers, backend developers, frontend developers, and testers to ensure timely and efficient development. Several factors can influence these estimates, such as the complexity of the machine learning model, the size and quality of the MRI dataset, the intricacies of integrating the Flask backend with the frontend interface, and the time allocated for thorough testing and validation. While the COCOMO model provides a solid framework for initial estimation, real-world development may require adjustments based on faced during the project lifecycle.

4. SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

1. Operating System	: Windows 11, 64-bit Operating System
2. Hardware Accelator	: CPU
3. Coding Language	: Python
4. Python distribution	: Google Colab Pro, Flask
5. Browser	: Any Latest Browser like Chrome

4.2 REQUIREMENT ANALYSIS

The MRI Brain Tumor Detection project aims to develop an efficient deep learning-based system that can accurately classify brain MRI images as either "Tumor" or "Non-Tumor." The system integrates a Convolutional Neural Network (CNN) with a Support Vector Machine (SVM) for enhanced accuracy. Key functionalities include image upload through a web interface, validation to ensure only brain MRI scans are accepted, preprocessing (like resizing and normalization), and displaying classification results with accuracy levels. The backend is developed using Python with Flask, while the frontend utilizes HTML, CSS, and JavaScript for user interaction. The system also ensures error handling, providing clear messages for invalid inputs.

Non-functional requirements focus on ensuring the system is fast, reliable, and secure. The project requires Python 3.10, frameworks like TensorFlow/Keras, OpenCV for image processing, and proper hardware with sufficient processing power and memory. It is crucial to have a diverse and well-labeled MRI dataset for training and validation. Deployment can be on a local or cloud server, ensuring easy access via a web browser. The application is designed for simplicity, aiming to help users with minimal technical expertise to upload images and interpret results efficiently.

4.3 HARDWARE REQUIREMENTS:

1. System Type : 64-bit operating system, x64-based processor
2. Cache memory : 4MB(Megabyte)
3. RAM : 16GB (gigabyte)
4. Hard Disk : 8GB
5. GPU : Intel® Iris® Xe Graphics

4.4 SOFTWARE

The MRI brain tumor detection project leverages a comprehensive set of software tools, technologies, and system configurations to ensure accurate, efficient, and scalable development and deployment. The system is designed to operate on Windows 11, 64-bit Operating System, ensuring compatibility with modern hardware and software environments. The project utilizes the CPU as the primary hardware accelerator, providing sufficient computational power for model training, inference, and backend processing.

The core development is carried out using the Python programming language, known for its simplicity, flexibility, and extensive library support. The Python environment is managed using Google Colab Pro for initial model development and training, which offers access to enhanced computational resources and faster processing speeds. For backend development and web application deployment, the Flask framework is employed, enabling efficient handling of API requests, data processing, and seamless integration with the frontend.

For the frontend, HTML5 and CSS3 are used to design and style the web interface, while Bootstrap ensures responsiveness and compatibility across devices. This guarantees an intuitive and user-friendly experience for end-users. Custom CSS styles are also applied to enhance the visual appeal and maintain a consistent layout. The application is compatible with any modern web browser, including the latest versions of Google Chrome, Mozilla Firefox, and Microsoft Edge, ensuring cross-browser accessibility and performance.

In terms of machine learning, the project employs TensorFlow/Keras to develop a Convolutional Neural Network (CNN) that efficiently extracts features from MRI images for tumor detection. Additionally, scikit-learn is used to implement a Support

Vector Machine (SVM) classifier, enhancing the accuracy and robustness of the classification process.

For image preprocessing, OpenCV is utilized to handle tasks such as image resizing, format validation, and noise reduction, ensuring that input data is clean and standardized for better prediction accuracy. NumPy supports efficient numerical computations, especially for handling large datasets and image matrices.Development and model evaluation are conducted within Jupyter Notebook, providing an interactive environment for experimentation and refinement. Visualization tasks, such as plotting accuracy curves and confusion matrices, are handled using Matplotlib, aiding in model performance analysis.

Overall, the integration of these software tools, along with the specified system requirements, ensures that the MRI brain tumor detection system is accurate, efficient, scalable, and compatible with modern computing environments.

4.5 SOFTWARE DESCRIPTION

The brain tumor detection system requires a modern and stable operating system, with Windows 11, 64-bit being the recommended choice. This ensures compatibility with the latest development tools, security updates, and efficient system performance. The CPU will serve as the primary hardware accelerator, which is suitable for running pre-trained models and managing backend operations. For large-scale training or intensive computations, cloud platforms like Google Colab Pro are utilized, offering access to advanced GPUs and faster processing capabilities.

The project is developed using Python, a versatile and widely-used programming language known for its extensive libraries and frameworks that simplify the development of machine learning and deep learning applications. The Python distribution includes platforms like Google Colab Pro for efficient model training and Flask for creating a lightweight and robust backend web application. Flask is particularly effective for deploying machine learning models as web services, allowing seamless integration between the backend and frontend.

Additionally, a web browser such as Google Chrome or any other latest version browser is required for accessing and interacting with the Flask-based web application.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

This project focuses on enhancing the detection and classification of brain tumors using a hybrid Deep Learning and Machine Learning approach. By integrating Convolutional Neural Networks (CNNs)[24] for effective feature extraction with Support Vector Machines (SVMs)[22] for precise classification, the model aims to provide a reliable and automated diagnostic tool. The ultimate goal is to assist in the early detection of brain tumors, improving treatment outcomes and reducing diagnostic errors in medical imaging.

The model leverages MRI scans, particularly T1-sequenced images, sourced from publicly available datasets like the Kaggle Fig share Brain Tumor Dataset. These images represent different tumor types, including glioma, meningioma, and pituitary tumors. Advanced preprocessing techniques, such as Adaptive Contrast Enhancement (ACE), Adaptive Gamma Correction (AGC), and Median Filtering[2], are applied to improve image quality by enhancing contrast, reducing noise, and ensuring better visualization of tumor regions.

To ensure accurate tumor region identification, the project employs Fuzzy C-Means (FCM) clustering for segmentation and uses the Grey Level Co-occurrence Matrix (GLCM) to extract critical texture features from MRI images. These features are then fed into the CNN-SVM hybrid model. The CNN component extracts deep features, while the SVM classifier provides precise decision boundaries, resulting in superior performance in distinguishing between healthy and abnormal brain tissues.

The effectiveness of the proposed model is evaluated using key performance metrics, including accuracy, sensitivity, specificity, and Jaccard coefficient. The model targets high performance levels, aiming for an accuracy of 97.94%, a sensitivity of 95%, and a specificity of 98.1%. This hybrid approach outperforms traditional methods such as standalone CNN, ANN, and RFC models, demonstrating its capability in accurate and efficient tumor classification.

The applications of this project extend to providing healthcare professionals with a powerful tool for automated and efficient brain tumor diagnosis. By minimizing misdiagnosis and expediting decision-making, the model contributes to improved patient outcomes. Looking ahead, the scope of the project includes extending the methodology to colored MRI images and 3D scans to further enhance segmentation and classification accuracy. Additionally, integrating the hybrid CNN-SVM[24] model into clinical diagnostic software will enable real-time applications in hospitals and medical imaging centers, fostering advancements in medical diagnostics.

5.1.1 DataSet

The dataset utilized in this project is a curated collection of T1-sequenced, enhanced MRI scans sourced from the Kaggle platform[14]. It forms the foundation for developing an advanced brain tumor detection and classification model. The dataset represents diverse diagnostic scenarios, encompassing four distinct categories: meningioma tumors, glioma tumors, pituitary tumors, and normal brain tissues without tumors as shown in Fig 5.1. Meningioma tumors originate from the meninges, the protective membranes surrounding the brain and spinal cord, while gliomas arise from glial cells that support brain function. Pituitary tumors are located in the pituitary gland, a vital endocrine structure at the brain's base. In addition to its diagnostic variety, the dataset features grayscale MRI scans converted into PNG format, ensuring compatibility with modern image-processing frameworks. The dataset is meticulously balanced across tumor types and normal cases, offering unbiased training and testing opportunities. Furthermore, it includes diverse tumor sizes, shapes, and intensities to simulate real-world diagnostic challenges effectively.

This dataset plays a pivotal role in the project's objective of developing a robust hybrid CNN-SVM model. Advanced preprocessing techniques, such as adaptive gamma correction, adaptive contrast enhancement, and median filtering, are employed to improve image clarity and reduce noise. Fuzzy C-Means clustering is used to accurately segment tumor regions, while the Gray-Level Co-Occurrence Matrix (GLCM) is applied for extracting critical texture features, such as contrast, energy, homogeneity, and correlation, essential for precise classification.

These features are then fed into the CNN-SVM[24] hybrid model, enabling high-precision differentiation between tumor types and healthy brain tissues. The dataset's comprehensive nature makes it an invaluable resource for advancing research in medical imaging, enabling the development of cutting-edge diagnostic tools for early and accurate tumor detection.

Feature	Details
Total Images	**3064**
Total Patients	**233**
Tumor Classes	**Glioma, Meningioma, Pituitary, No Tumor**
Class Distribution	**Glioma: 1426, Meningioma: 708, Pituitary: 930 No Tumor 804**
Image Type	**T1-weighted contrast-enhanced MRI**
Applications	**Tumor classification and detection**

TABLE 1 . DATASET DESCRIPTION

Tumor Classes:

- Glioma (1426 images): These are irregular and infiltrative tumors commonly located in the brain's temporal or frontal lobes.
- Meningioma (708 images): These are slow-growing, rounded tumors often found near the skull base.
- Pituitary Tumor (930 images): These tumors are localized around the pituitary gland and can vary in size and shape.
- No Tumor: Normal brain MRIs without any pathological indications.

This dataset contains 3064 T1-weighted contrast-enhanced MRI images obtained from 233 patients as shown in Table 1, categorized into four classes: Glioma, Meningioma, Pituitary Tumor, and No Tumor. It is designed to support research in brain tumor detection and classification.

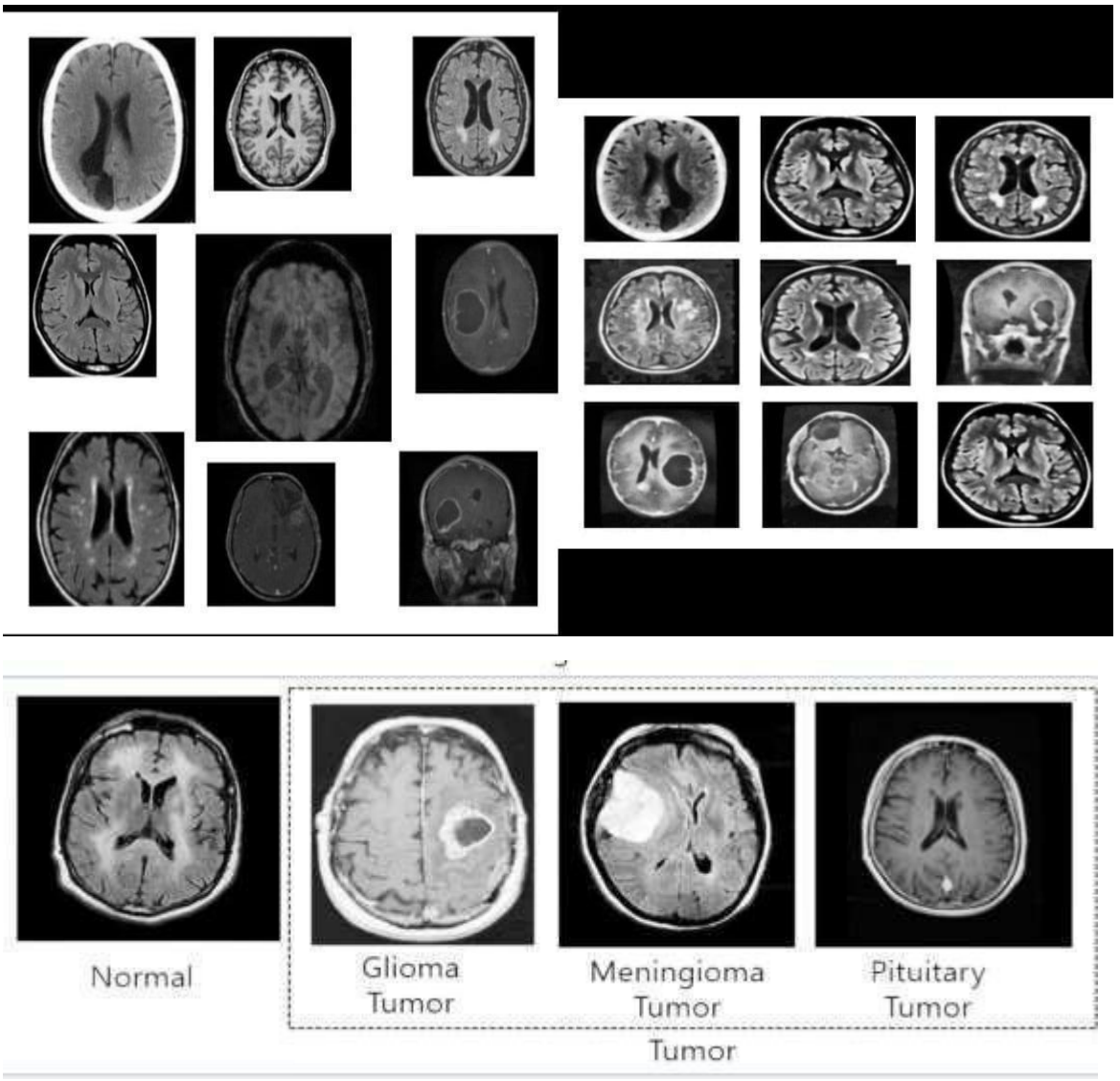


FIG 5.1 DIFFERENT TUMOR CLASSES DATA SET IMAGES.

Image Characteristics:

- All images are T1-weighted and contrast-enhanced, improving the visibility of soft tissues and tumor boundaries.
- Stored in PNG format, facilitating easy use in Machine Learning workflows.

Applications:

- Used in training and testing Machine Learning and Deep Learning models for medical image analysis.
- Supports tasks like tumor detection, classification, and segmentation in medical and research settings.

5.1.2 DATA PRE-PROCESSING

Before feeding data to an algorithm, we have to apply transformations to our data which is referred as pre-processing. By performing pre-processing, the raw data which is not feasible for analysis is converted into clean data. In-order to achieve better results using a model in Deep Learning, data format has to be in a proper manner. The data should be in a particular format for different algorithms. Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Preprocessing is the first step while creating the Deep Learning model. During this preprocessing, images undergo several preprocessing methods applied to MRI images to enhance their quality and ensure accurate brain tumor classification. These methods are as follows:

Adaptive Contrast Enhancement (ACE): Enhances image contrast by calculating local statistics like mean pixel intensity and standard deviation.

Adaptive Gamma Correction (AGC): Dynamically adjusts brightness and contrast in different image regions for better visualization of tumors.

Median Filtering: Median filtering[2] is used to reduce noise in the MRI images without losing important details, especially edges and fine structures. It works by replacing each pixel value with the median value of its surrounding neighborhood, which helps to remove outliers or noise that could otherwise distort the image. This filtering technique is particularly effective in eliminating speckle noise or salt-and-pepper noise, ensuring that the tumor regions are preserved while irrelevant noise is minimized.

Fuzzy C-Means (FCM) Segmentation: Extracts precise tumor regions by grouping similar intensity pixels. FCM is a clustering technique used to segment the MRI image into distinct regions based on pixel intensity. Unlike traditional segmentation methods that assign each pixel to a single cluster, FCM[3] allows for fuzzy assignments, meaning each pixel can belong to multiple clusters with different membership degrees. This is particularly useful in medical imaging, where the tumor region may have varying intensities and blend with surrounding tissues. FCM ensures more accurate and precise segmentation as shown in Fig

5.2 of tumor regions, which is essential for further analysis and classification.

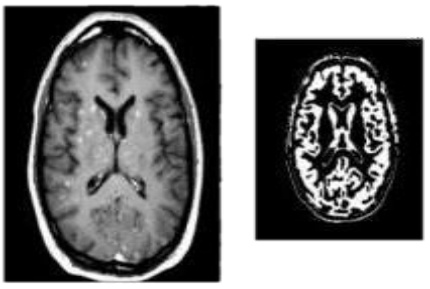


FIG 5.2 IMAGE AFTER APPLYING THE PREPROCESSING TECHNIQUE.

5.1.3 FEATURE EXTRACTION

Feature extraction using Gray Level Co-occurrence Matrix (GLCM) is a widely used method in image processing, particularly for the analysis of medical images like MRI scans for brain tumor detection and classification. GLCM is a statistical method that captures the spatial relationship between pixels in an image, allowing the extraction of texture features that can reveal important characteristics of the image that are often difficult to discern with the naked eye. These features are crucial for improving the performance of classification models, especially in the case of detecting and classifying brain tumors.

The Gray Level Co-occurrence Matrix (GLCM)[2] is a matrix that describes the frequency of pixel pairs with specific values in a specified spatial relationship, typically focusing on pairs of pixels separated by a certain distance in a particular direction. The main idea is to quantify the spatial patterns of pixel intensities in an image, which are often associated with different tissue types, such as healthy tissue, tumor regions, and surrounding structures in MRI scans.

```
Processed and extracted features from image-2344.png
{'contrast': array([181.39786195, 255.19572081, 100.74935939, 246.6949265 ,
       491.38199807, 255.19572081, 293.19783623, 246.6949265 ,
       845.3803733 , 689.85033682, 533.79480446, 663.54330006]), 'dissimilarity': array([ 4.50132127,  5.93400229,  3.55449231,  5.84218464,  8.53016409,
        5.93400229,  6.73556145,  5.84218464, 12.18826761, 11.11667072,
        9.71048804, 10.92748154]), 'homogeneity': array([0.66445964, 0.60342141, 0.6706268 , 0.60206301, 0.56608546,
       0.60342141, 0.56726666, 0.60206301, 0.5101268 , 0.50022026,
       0.50522613, 0.49921692]), 'energy': array([0.33574772, 0.32942395, 0.33664104, 0.32943005, 0.32784459,
       0.32942395, 0.32950412, 0.32943005, 0.3202086 , 0.31536951,
       0.32263689, 0.3153834 ]), 'correlation': array([0.98203008, 0.97474361, 0.99001941, 0.97558493, 0.95136878,
       0.97474361, 0.97098272, 0.97558493, 0.91641264, 0.93185192,
       0.9472208 , 0.93445071]), 'ASM': array([0.11272653, 0.10852014, 0.11332719, 0.10852416, 0.10748207,
       0.10852014, 0.10857296, 0.10852416, 0.10253354, 0.09945793,
       0.10409456, 0.09946669])}
```

FIG 5.3 MRI SCAN AFTER APPLYING GLCM FEATURE EXTRACTION

Key Features Extracted from GLCM:

Following several statistical features can be obtained from GLCM to characterize the MRI image texture. In this approach was suggested, contrast, dissimilarity, homogeneity, energy, correlation

- **Contrast**: It measures the intensity difference between neighboring pixels. A higher contrast value which may be useful in differentiating tumor areas from normal tissue.

$$\text{contrast} = \sum_{p,q} (p-q)^2 a(p,q)$$

- **Correlation:** In an image, correlation measures of how much image data points(pixels) are related linearly to one another. This indicative bare skeleton measure indicates how intensity correlates with the pixel map of different positions in any given place.

$$\text{correlation} = \frac{\sum_{l,m}(l-\mu_x)(j-\mu_y)P(l,m)}{\sigma_x \sigma_y}$$

- **Energy:** It represents the combination of the class features in the GLCM, which means that the text is identical.

$$\text{energy} = \sum_{i,j} R(i,j)^2$$

- **Homogeneity**: In measures how close the distribution of features in the GLCM is to the diagonal.

$$\text{homogeneity} = \sum_{i,j} \frac{Q(i,j)}{1+(i-j)^2}$$

5.1.4 MODEL BUILDING :

Model building in the context of Deep Learning refers to the process of designing and constructing hybrid model using neural network architectures and support vector machines to solve specific tasks such as classification of different brain tumors. Deep Learning models typically consist of multiple layers of neurons organized in a hierarchical fashion, enabling the model to learn intricate patterns and representations from the data.The hybrid CNN-SVM architecture integrates the feature extraction

capabilities of Convolutional Neural Networks (CNNs) with the classification strengths of Support Vector Machines (SVMs)[24] to achieve high accuracy .The model is designed to process MRI images, extract relevant features, and classify them into different categories such as glioma, meningioma, pituitary tumors, or normal tissue. The CNN-SVM hybrid model is a powerful computational approach for detecting and classifying brain tumors from MRI images. This model leverages the strengths of Convolutional Neural Networks

(CNNs) for feature extraction and Support Vector Machines (SVMs) for precise classification, achieving high accuracy and reliability in medical imaging.

Convolutional Neural Networks:

A Convolutional Neural Network (CNN)[17] is a type of Deep Learning model particularly well-suited for processing structured data such as images. CNNs automatically and adaptively learn spatial hierarchies of features from input data through the use of convolutional filters, pooling, and fully connected layers as shown in Fig 5.4.

Layers in Convolutional Neural Networks:

- **Input Layer -** Accepts the raw input data, such as images or videos, in the form of multi- dimensional arrays.
- **Convolutional Layer -** Extracts features from the input data by applying learnable filters (kernels) that slide over the input. Detects low-level features (e.g., edges) in initial layers and high-level patterns (e.g., shapes) in deeper layers. Outputs feature maps that represent the detected patterns.
- **Activation Layer -** Introduces non-linearity to the network, enabling it to model complex patterns. Most common activation function **ReLU** , Other options Sigmoid, Tanh, or Leaky ReLU**.**
- **Pooling Layer -** Reduces the spatial dimensions of the feature maps while preserving important information. Reduces computation and mitigates overfitting. Common types: **Max Pooling:** Retains the maximum value from each pooling window.

  Average Pooling: Computes the average value within each pooling window.
- **Flattening Layer** - Converts the multi-dimensional feature maps into a one-dimensional vector.
- **Fully Connected Layer (Dense Layer) -** Connects every neuron in one laye

every neuron in the next, enabling complex pattern learning. Combines and processes the extracted features for classification or regression tasks. Outputs a vector of class scores or predictions.

- **Output Layer -** Produces the final predictions of the network. Number of neurons matches the number of target classes.

  Common activation functions:

  SoftMax: For multi-class classification, outputs probabilities for each class.

  Sigmoid: For binary classification, outputs a probability between 0 and 1.

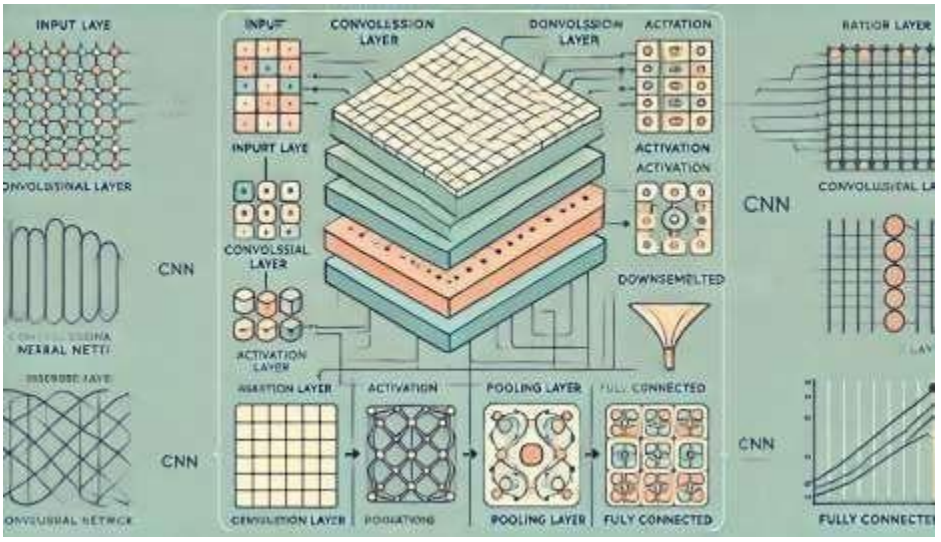


FIG 5.4 CNN MODEL ARCHITECTURE

Support Vector Machine(SVM):

The Support Vector Machine (SVM)[22] is a supervised Machine Learning algorithm designed for classification and regression tasks. It works by identifying the optimal decision boundary, known as a hyperplane, which separates data points belonging to different classes. The SVM aims to maximize the margin, which is the distance between the hyperplane and the closest data points from each class. These critical data points are called support vectors, and they play a vital role in defining the hyperplane.

For data that is not linearly separable, SVM uses a technique called the kernel trick. Kernels map the input data into a higher-dimensional space where a linear separation becomes possible. Common kernel functions include the linear kernel (used for linearly separable data), polynomial kernel, and Radial Basis Function (RBF) kernel, which handles more complex, non-linear data patterns.

SVM training involves solving an optimization problem to find the hyperplane that maximizes the margin while minimizing misclassification. To balance accuracy and robustness, a regularization parameter C is used. A larger Crocuses on minimizing classification errors, while a smaller C allows for a wider margin, tolerating some misclassifications to improve generalization.

SVM is effective in high-dimensional spaces and works well with small datasets due to its robustness against overfitting. However, it can be computationally intensive for large datasets, and its performance heavily depends on the choice of kernel and hyperparameters. Despite these challenges, SVM is widely used in applications such as text classification, image recognition, and bioinformatics for its ability to provide accurate and reliable results.

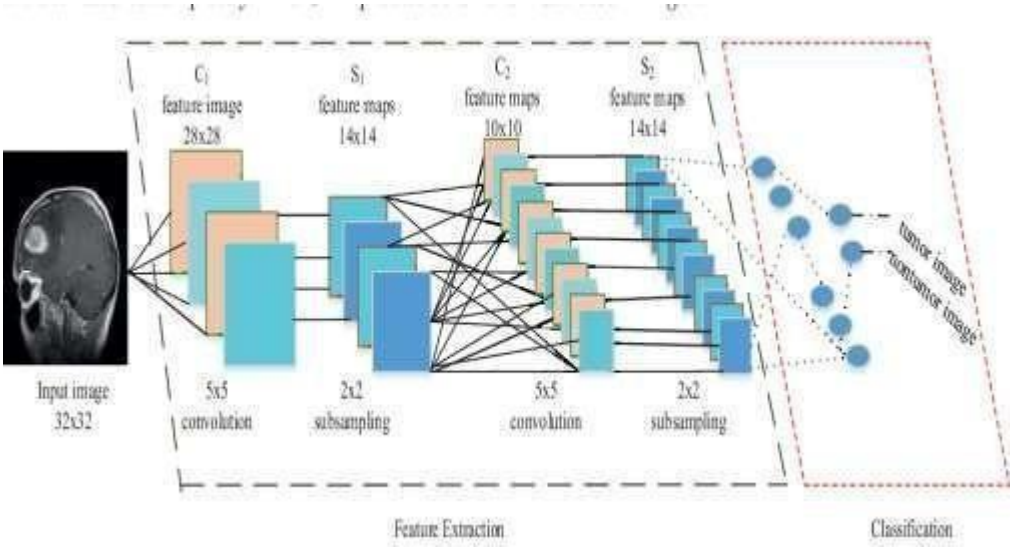


FIG 5.5 CNN-SVM ARCHITECTURE

CNN-SVM Model Building Process:

The CNN-SVM model combines the strengths of Convolutional Neural Networks (CNNs) for feature extraction and Support Vector Machines (SVMs) for classification. The process begins with input MRI images, which are preprocessed to improve quality and reduce noise. Techniques like adaptive contrast enhancement, gamma correction, and median filtering are used to enhance image clarity. These preprocessed images are then fed into the CNN for feature extraction.

In the CNN, convolutional layers apply filters to detect spatial features such as edges, textures, and patterns. Each layer generates feature maps that capture various aspects of the image. To introduce non-linearity and help the network capture complex relationships, ReLU (Rectified Linear Unit) is applied, zeroing out negative

pixel values while retaining positive ones. Pooling layers, such as max-pooling or average-pooling, are used to down-sample the feature maps, reducing their dimensions while preserving critical information. Finally, the output feature maps are flattened into a one-dimensional vector to prepare them for input into the SVM classifier.

The flattened feature vectors are passed to the SVM, which uses a kernel function to map the data into a higher-dimensional space where a linear decision boundary can be identified. The SVM determines the optimal hyperplane that separates the classes, such as tumor vs. normal tissue, and outputs the predicted category of the MRI image. The training process involves training the CNN first with a loss function, such as cross-entropy, to minimize classification errors during feature extraction. Once trained, the CNN is used solely for feature extraction, without its built-in classification layer. The extracted features are then used to train the SVM, which optimizes its parameters for precise class separation.

Finally, the CNN and SVM are integrated into a hybrid model. During inference, MRI images pass through the CNN to extract features, which are subsequently classified by the SVM. This combination leverages the deep feature extraction capabilities of CNN and the precise classification of SVM, yielding accurate predictions.

Advantages of Hybrid Model:

- Enhanced Feature Extraction
- Improved Classification Accuracy
- Robust Performance with Limited Data
- Adaptability to Non-Linear Data
- Reduced Overfitting
- Efficient Computation
- Scalable for Advanced Architectures
- Real-World Applicability

5.1.5 CLASSIFICATION

Classification using CNN-SVM proposed hybrid model :

The convolutional neural network was originally proposed by LeCun et al. for handwritten recognition has been successful in image identification, detection, and segmentation of the image. CNN has a high ability in large-scale image classification. The CNN-SVM hybrid model[22,24] effectively integrates Convolutional Neural Networks (CNNs) for feature extraction and Support Vector Machines (SVMs) for classification as shown in Fig 6.4, offering a robust and accurate approach to brain tumor detection. The classification process begins with the CNN component, which processes input MRI images to extract meaningful features. Convolutional layers detect patterns like edges, textures, and shapes that represent the visual features of the image. The activation function (typically ReLU) introduces non-linearity by retaining positive pixel values and discarding negative ones, enabling the network to capture complex relationships. Pooling layers then reduce the spatial dimensions of the feature maps, preserving essential information while minimizing computational costs. Finally, the multi-dimensional feature maps are flattened into a one- dimensional vector to make the data compatible with the SVM classifier.

A Convolutional layer and pooling layer are the most important layers on CNN. Convolutional layer is used for extract feature by combining the image area with many filters. Pooling layer reduces the size of the output map of the convolution layer and prevents overfitting. A very popular approach to down sampling is to use pooling layers. Pooling layer usually deciphers the image (like ax) in the aggregation into a single unit. Through these two layers the number of neurons, parameters, and connections is much less than there is a CNN model. This makes CNN more efficient compared with BP networks with similar layers.

The flattened feature vectors are passed into the SVM, where classification occurs. For non-linearly separable data, the SVM employs a kernel function, such as the Radial Basis Function (RBF) kernel, to project the features into a higher-dimensional space where a linear separation becomes possible. The SVM then identifies the optimal hyperplane that maximizes the margin between classes, with the margin being the distance between the hyperplane and the nearest data points, known as support vectors.

Based on the position of the input features relative to the hyperplane, the SVM assigns the MRI image to a specific class, such as glioma, meningioma, pituitary tumor, or normal tissue.

Training the CNN-SVM model involves separate training phases for the CNN and SVM components. The CNN is first trained using a loss function, such as cross-entropy loss, to minimize errors during feature extraction. Once trained, the CNN's classification layer is removed, and it is used solely for extracting feature vectors from input images. These extracted features are then used to train the SVM, which optimizes its parameters to achieve

precise class separation. During inference, MRI images are processed through the trained CNN to extract features, which are then classified by the SVM to provide the final prediction. This hybrid model combines the ability of CNNs to learn intricate visual features with the precision of SVMs in boundary creation, resulting in highly accurate classifications.

The CNN-SVM model's classification approach is particularly advantageous in medical applications as shown in Fig 5.6. It combines the deep feature extraction capabilities of CNNs with the precise decision-making ability of SVMs, ensuring improved diagnostic accuracy. The hybrid model is effective in handling both linearly and non-linearly separable data, reducing false positives and negatives. This makes it a reliable and efficient tool for detecting and classifying brain tumors, aiding early diagnosis and improving treatment outcomes.

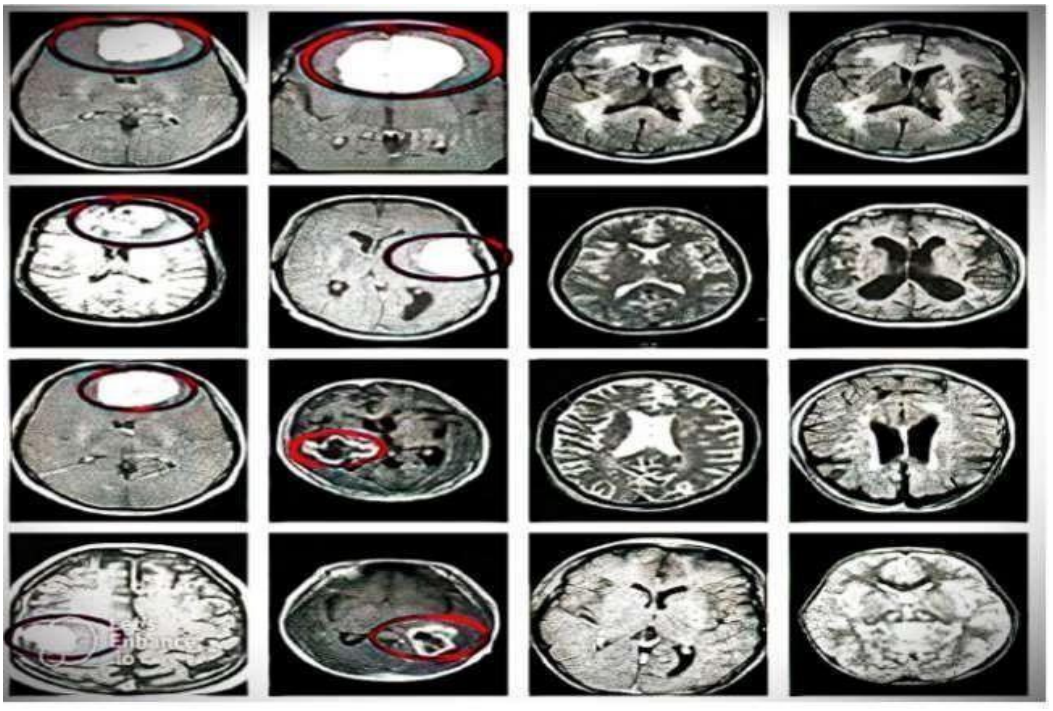


FIG 5.6 CLASSIFICATION OF BRAIN TUMORS

This hybrid approach significantly enhances the model's overall accuracy compared to using CNN or SVM alone. CNNs excel in automatically extracting rich and meaningful features from raw data, while SVMs are known for their high precision in classifying these features. By combining these two techniques, the CNN-SVM model leverages the best of both worlds, resulting in improved reliability, reduced false positives and negatives, and better generalization to unseen data.

In applications such as brain tumor detection, this enhanced prediction accuracy is critical. Early and accurate diagnosis of tumors can lead to timely medical interventions, improving patient outcomes. The CNN-SVM model, with its ability to handle complex data and deliver precise classifications, proves to be an invaluable tool in medical imaging and other fields requiring high accuracy and reliability.

The training mode for CNN-SVM hybrid model uses dual optimization strategy making use of the two. The CNN [11] is initially trained to extract feature vectors from input MRI images. These are fed into SVM [15] for classifying purposes. This system performs alternating updates of both CNN and SVM based on metrics a. results than when only CNN or SVM were used separately. The developed model identifies and detect the tumor.

The outcome model of CNN-SVM combine CNN to extract features and SVM for the ultimate classification which enhances prediction accuracy.

These graphs illustrate the training progress of a hybrid model, CNN-SVM model, over multiple epochs. The left graph depicts accuracy, showing the percentage of correctly classified samples over time, with separate lines for training and validation data. Both training and validation accuracy increase rapidly initially, indicating effective learning, though the training accuracy continues to climb ,while the validation accuracy plateaus around 90%, creating a noticeable gap.

The right graph displays loss, representing the error between predictions and actual values, also with training and validation lines. Both training and validation loss decrease significantly at first, as expected, but the validation loss flattens and even slightly increases after a certain point, while training loss continues to approach zero. This combination of increasing accuracy and decreasing loss, coupled with the diverging validation metrics, suggests the model is learning well but is also showing signs of overfitting, meaning it's memorizing the training data rather than generalizing effectively to unseen data.

Therefore, while the model has learned substantially, techniques like regularization, data augmentation, or early stopping should be considered to mitigate overfitting and potentially improve its performance on new, unseen data.

Other models compared with the proposed CNN-SVM model:

Visual Geometry Group Networks (VGG):

VGG is a type of CNN known for its simple and uniform architecture. It uses small convolutional filters (3x3) and stacks them in multiple layers to increase the depth of the network while maintaining computational feasibility. The uniformity of VGG's design allows for straightforward implementation and scalability. Models like VGG16 and VGG19 (16 and 19 layers deep, respectively) are widely used for image classification tasks. Although computationally expensive and requiring significant memory, VGG networks deliver high accuracy, especially in benchmarks like ImageNet.

Recurrent Neural Networks (RNNs):

RNNs[11] are neural networks designed for sequential or time-series data, such as text, audio, or sensor readings. Unlike feedforward networks, RNNs have connections that form cycles, allowing them to maintain a memory of previous inputs. This capability makes them suitable for tasks where context and order are critical, such as natural language processing, speech recognition, and time-series forecasting. However, standard RNNs struggle with long-term dependencies due to issues like vanishing gradients, which more advanced architectures like LSTMs (Long Short-Term Memory) and GRUs (Gated Recurrent Units) address effectively.

Random Forest Classifier (RFC):

Random Forest[2] is an ensemble learning technique based on decision trees. It creates multiple decision trees during training and aggregates their outputs (through voting for classification or averaging for regression) to improve accuracy and reduce overfitting. Random Forest is robust to noise and works well with structured tabular data, making it useful for various applications such as fraud detection, medical diagnosis, and financial forecasting.

Fully Connected Neural Networks (FCNNs):

FCNNs, also known simply as dense neural networks, are a basic type of Deep Learning model where every neuron in one layer is connected to every neuron in the next layer. This architecture is powerful for learning global patterns in data but is less efficient for tasks requiring spatial hierarchies, such as image recognition. While FCNNs can handle structured and unstructured data, their heavy reliance on fully connected layers makes them computationally intensive and prone to overfitting for high-dimensional data.

Artificial Neural Networks (ANNs):

ANNs are the foundation of modern neural network architectures. They consist of layers of interconnected nodes (neurons) designed to simulate the way the human brain processes information. Each neuron receives inputs, applies a weighted sum followed by an activation function, and passes the result to subsequent neurons. ANNs are versatile and can handle a wide range of problems, from simple regression tasks to complex pattern recognition. However, their basic structure lacks the specialization of CNNs or the sequence-processing capabilities of RNNs, making them less suitable for certain tasks like image or sequential data analysis.

These models collectively cater to diverse domains, from visual data processing with CNNs and VGG[4], sequential data analysis with RNNs, to tabular data modeling with RFC and ANN. Their applicability depends on the nature and requirements of the task at hand.

5.2 MODULES

In the context of software development, a module is a self-contained, independent unit of code that performs a specific task or functionality within a larger system.

CNN-SVM Brain Tumor Detection Project Modules:

1. Data Collection Module: Collects and organizes MRI images into categories like Meningioma, Glioma, Pituitary Tumor, and Non-tumor.

Sample Code:

```
import os
def load_images_from_folder(folder):
```

```
    images = []
    for filename in os.listdir(folder):
        if filename.endswith(".png"):
            images.append(os.path.join(folder, filename))
    return images
```

2. Preprocessing Module: Enhances image quality and reduces noise.

Sample Code:

```
import cv2
import numpy as np
def preprocess_image(image_path):
    image = cv2.imread(image_path, 0)
    # Adaptive Contrast Enhancement
    image = cv2.equalizeHist(image)
    # Median Filtering
    image = cv2.medianBlur(image, 3)
    return image
```

3. Segmentation Module: Segments tumor regions using Fuzzy C-Means.

Sample Code:

```
from sklearn.cluster import KMeans
def segment_image(image):
    image = image.reshape((-1, 1))
    kmeans = KMeans(n_clusters=2).fit(image)
    segmented_image = kmeans.labels_.reshape((256, 256))
    return segmented_image
```

4. Feature Extraction Module: Extracts texture features using GLCM.

Sample Code:

```
from skimage.feature import greycomatrix, greycoprops
def extract_features(image):
    glcm = greycomatrix(image, [1], [0], symmetric=True, normed=True)
    contrast = greycoprops(glcm, 'contrast')[0, 0]
    energy = greycoprops(glcm, 'energy')[0, 0]
    return [contrast, energy]
```

5. CNN Feature Extraction Module: Extracts deep features using CNN.

Sample Code:

```
from tensorflow.keras.applications import VGG16
```

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
```

6. SVM Classification Module: Classifies features using SVM.

Sample Code:

```
from sklearn import svm
from joblib import dump, load
```

7. Evaluation Module: Evaluates model performance**.**

Sample Code:

```
from sklearn.metrics import accuracy_score
def evaluate_model(true_labels, predictions):
    return accuracy_score(true_labels, predictions)
```

8. Flask Backend Module: Manages API endpoints.

Sample Code:

```
from flask import Flask, request, jsonify
app = Flask(__name__)
@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['file']
    # Process file and return prediction
    return jsonify({'result': 'Prediction Here'})
if __name__ == '__main__':
    app.run(debug=True)
```

9. Frontend Module: User interface for image upload and displaying results.

Sample Code:

```
<form action="/predict" method="post" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="Upload">
</form>
```

10. File Management Module: Manages file storage and cleanup.

Sample Code:

```
import os
def delete_file(file_path):
    if os.path.exists(file_path):
        os.remove(file_path)
```

5.3 UML DIAGRAMS

The workflow of the CNN-SVM model for brain tumor detection begins with collecting and preprocessing MRI images for training and testing. Preprocessing involves steps such as resizing, normalization, and applying techniques like Adaptive Gamma Correction (AGC)[2] for brightness and contrast adjustment, Median Filtering for noise reduction, and Fuzzy C-Means Clustering (FCM) for tumor region segmentation. Feature extraction is performed using Grey Level Co-Occurrence Matrix (GLCM) to obtain texture features like contrast, energy, and homogeneity.

The processed images are divided into training and testing datasets, and each image is converted into pixel arrays for compatibility with the neural network. The CNN-SVM hybrid model[22,24] is then trained on the extracted features from the training set, leveraging CNN for feature extraction and SVM for classification. During the testing phase, the model's performance is evaluated using metrics such as accuracy, sensitivity, specificity, and Jaccard Coefficient to measure its effectiveness in tumor detection.

Once the CNN-SVM model is successfully trained, the trained model can be stored for future applications. Saving the model involves preserving its architecture, learned weights, and parameters, enabling it to be reloaded and used without retraining. This is particularly beneficial for deployment in clinical settings, where the model can be integrated into software systems for real-time brain tumor detection and classification. After training, a performance report is generated to evaluate the model's effectiveness and reliability. This report typically includes metrics such as accuracy, sensitivity, specificity, precision, and the Jaccard Coefficient. These metrics provide a quantitative understanding of how well the model performs in distinguishing between tumor and non-tumor cases and correctly identifying specific tumor types.In clinical applications, such visual insights are invaluable for building trust among healthcare professionals. They allow users to interpret the model's decisions, evaluate its reliability, and make informed choices about integrating it into diagnostic workflows.This flowchart Fig 5.7 outlines the process of building and evaluating a Machine Learning pipeline using image data and a hybrid CNN-SVM model. The process begins with image acquisition, where images are collected from sources like cameras, sensors, or datasets. These images are then prepared in the image preprocessing step, which involves cleaning and standardizing the data through resizing, normalization, or removing noise to make it suitable for the model.

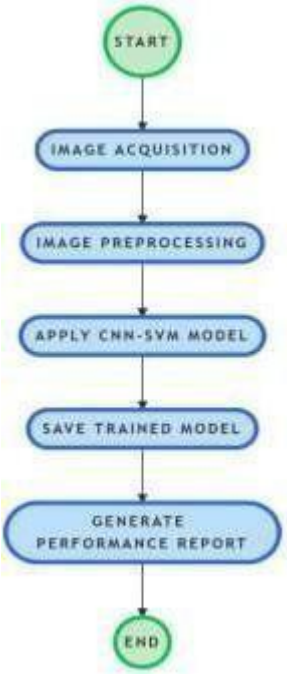


FIG 5.7 DESIGN OVERVIEW

Next, the prepared images are processed by the CNN-SVM model. The CNN extracts key features from the images, such as shapes and textures, while the SVM uses these features to classify the images into specific categories. After training, the trained model is saved so it can be reused without retraining.

In the CNN-SVM brain tumor detection project, a UML (Unified Modeling Language) diagram helps visualize the system's structure and workflow. The Class Diagram shows the main components and their roles. The ImagePreprocessor class handles image loading and preprocessing tasks like contrast enhancement, gamma correction, and noise reduction. The CNNFeatureExtractor uses deep learning models (like VGG16) to extract important features from these images. These features are then classified by the SVMClassifier into tumor or non-tumor categories. The ModelEvaluator checks the model’s performance using metrics like accuracy and precision. The FlaskApp manages backend API routes for tasks like image upload and predictions, while the Frontend provides a simple interface for users to upload images and view results.

The Sequence Diagram explains the step-by-step process. The user uploads an MRI image through the frontend. The Flask backend processes this image using the ImagePreprocessor. Then, CNNFeatureExtractor extracts features, which are classified by the SVMClassifier. The results are evaluated and sent back to the frontend for display.Overall, UML diagrams help in understanding the system's structure and how different parts interact, making it easier to design, build, and maintain the project.

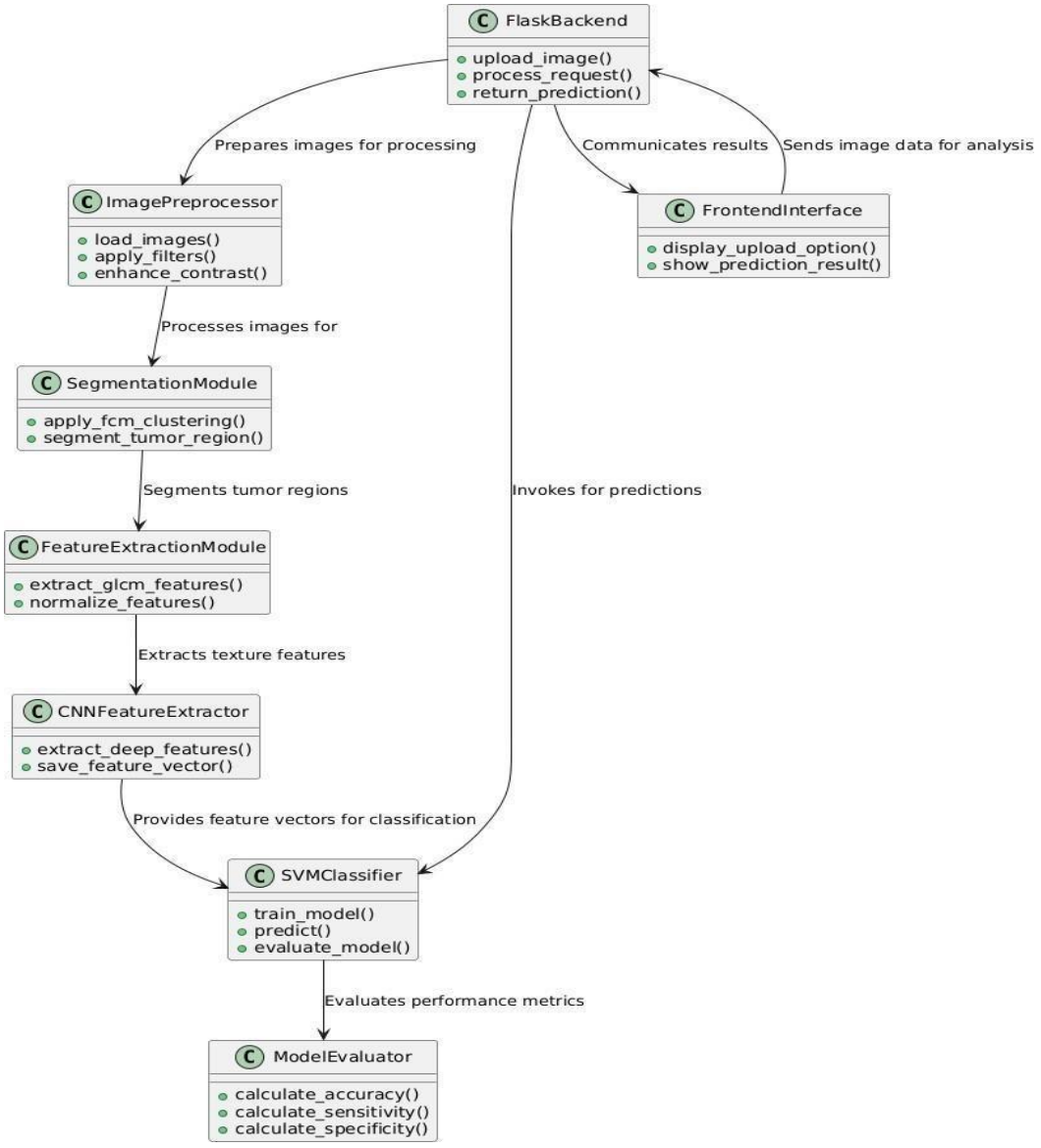


Fig 5.8 UML DIAGRAM FOR BRAIN TUMOR DETECTION AND CLASSIFICATION

The UML diagram fig 5.8 represents the modular structure and workflow of the CNN-SVM[22,24] brain tumor detection project. The system starts with the FlaskBackend, which handles backend operations such as processing image upload requests, managing the processing workflow, and returning prediction results. The FrontendInterface interacts with the user, providing options to upload images and displaying the classification results. Once an image is uploaded, it undergoes preprocessing through the ImagePreprocessor module, which loads the image, applies necessary filters, and enhances contrast for better visibility. The processed image is then passed to the SegmentationModule, where tumor regions are segmented using techniques like FCM clustering.

After segmentation, the FeatureExtractionModule extracts essential texture features using methods like the Gray-Level Co-occurrence Matrix (GLCM) and normalizes them for consistency. Parallelly, deep features are extracted by the CNNFeatureExtractor, which automatically processes the images and saves the feature vectors for classification. These combined features are then fed into the SVMClassifier, which handles the training, prediction, and evaluation of the model. Finally, the ModelEvaluator calculates key performance metrics such as accuracy, sensitivity, and specificity to assess the model's effectiveness. The results are then communicated back to the frontend for user display. This modular approach ensures a streamlined, efficient, and accurate process for detecting and classifying brain tumors from MRI images.

6. IMPLEMENTATION

6.1 MODEL IMPLEMENTATION

CNN-SVM Model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, jaccard_score, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv1D, MaxPooling1D,
BatchNormalization, GlobalAveragePooling1D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import matplotlib.pyplot as plt

# Combine GLCM features into a single DataFrame
all_glcm_features = []
all_labels = []

for features, label in [(gglcm_features_list, 'glioma'),
(mglcm_features_list, 'meningioma'),
(nglcm_features_list, 'no'), (pglcm_features_list,
'pituitary')]:
for feature_dict in features:
flattened_features = [val for sublist in feature_dict.values() for val in sublist]
all_glcm_features.append(flattened_features)
all_labels.append(label)

df = pd.DataFrame(all_glcm_features)
df['label'] = all_labels

# Encode labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['label'])

# Split the data
X = df.drop('label', axis=1)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)

print("Training set shape:", X_train.shape, y_train.shape)
print("Validation set shape:", X_val.shape, y_val.shape)
```

```
print("Testing set shape:", X_test.shape, y_test.shape)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Reshape data for CNN (1D convolution)
X_train_cnn = X_train_scaled.reshape(X_train_scaled.shape[0], X_train_scaled.shape[1], 1)
X_val_cnn = X_val_scaled.reshape(X_val_scaled.shape[0], X_val_scaled.shape[1], 1)
X_test_cnn = X_test_scaled.reshape(X_test_scaled.shape[0], X_test_scaled.shape[1], 1)

# Define the improved CNN model with Global Average Pooling
def create_cnn_model(input_shape):
model = Sequential()
model.add(Conv1D(64, kernel_size=7, activation='relu', input_shape=input_shape))
model.add(MaxPooling1D(pool_size=3))
model.add(Conv1D(128, kernel_size=5, activation='relu'))
model.add(MaxPooling1D(pool_size=3))
model.add(Conv1D(256, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(GlobalAveragePooling1D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5)) model.add(BatchNormalization())
model.add(Dense(4, activation='softmax')) # 4 output classes
model.compile(optimizer=Adam(learning_rate=0.0005),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model

# Initialize and train CNN
cnn_model = create_cnn_model((X_train_cnn.shape[1], 1))
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
min_lr=0.00001)

cnn_history = cnn_model.fit(X_train_cnn, y_train,
epochs=150, batch_size=64,
validation_data=(X_val_cnn, y_val),
callbacks=[early_stopping, reduce_lr])

# Extract features from CNN
cnn_features_train = cnn_model.predict(X_train_cnn)
cnn_features_val = cnn_model.predict(X_val_cnn)
```

```
cnn_features_test = cnn_model.predict(X_test_cnn)

# Flatten the CNN features for SVM
cnn_features_train_flattened = cnn_features_train.reshape(cnn_features_train.shape[0], -1)
cnn_features_val_flattened = cnn_features_val.reshape(cnn_features_val.shape[0], -1)
cnn_features_test_flattened = cnn_features_test.reshape(cnn_features_test.shape[0], -1)

# Grid search for SVM hyperparameters
svm_model = SVC(probability=True)
param_grid = {
'kernel': ['linear', 'rbf'],
'C': [1, 10, 100],
'gamma': ['scale', 'auto']
}
grid_search = GridSearchCV(svm_model, param_grid, cv=3, scoring='accuracy')
grid_search.fit(cnn_features_train_flattened, y_train)

# Best SVM model
best_svm_model = grid_search.best_estimator_

# Evaluate SVM
y_pred_test = best_svm_model.predict(cnn_features_test_flattened)

# Calculate metrics
train_accuracy = accuracy_score(y_train,
best_svm_model.predict(cnn_features_train_flattened))
val_accuracy = accuracy_score(y_val, best_svm_model.predict(cnn_features_val_flattened))
test_accuracy = accuracy_score(y_test, y_pred_test)

train_jaccard = jaccard_score(y_train,
best_svm_model.predict(cnn_features_train_flattened), average='macro')
val_jaccard = jaccard_score(y_val, best_svm_model.predict(cnn_features_val_flattened),
average='macro')
test_jaccard = jaccard_score(y_test, y_pred_test, average='macro')

def sensitivity_specificity(y_true, y_pred):
ccm = confusion_matrix(y_true, y_pred) tp
= np.diag(ccm)
fp = ccm.sum(axis=0) - tp fn
= ccm.sum(axis=1) - tp
tn = ccm.sum() - (fp + fn + tp) accuracy
= (tp + tn) / (tp + fp + fn + tn) sensitivity
= tp / (tp + fn)
specificity = tn / (tn + fp) return
sensitivity, specificity

sensitivity_train, specificity_train = sensitivity_specificity(y_train,
```

```
    best_svm_model.predict(cnn_features_train_flattened))
    sensitivity_val, specificity_val = sensitivity_specificity(y_val,
    best_svm_model.predict(cnn_features_val_flattened))
    sensitivity_test, specificity_test = sensitivity_specificity(y_test, y_pred_test)

    print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
    print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
    print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
    print(f"Training Jaccard Coefficient: {train_jaccard:.2f}")
    print(f"Validation Jaccard Coefficient: {val_jaccard:.2f}")
    print(f"Testing Jaccard Coefficient: {test_jaccard:.2f}")

    # Calculate average sensitivity for training set
    avg_sensitivity_train = np.mean(sensitivity_train)
    print(f"Training Sensitivity: {avg_sensitivity_train:.2f}")
    avg_sensitivity_test = np.mean(sensitivity_test)
    print(f"Testing Sensitivity: {avg_sensitivity_test:.2f}")

    # Iterate through the specificity values for each class
    for i, specificity in enumerate(specificity_train):
    print(f"Training Specificity for class {i}: {specificity:.2f}")

for i, specificity in enumerate(specificity_val): print(f"Validation
    Specificity for class {i}: {specificity:.2f}")

for i, specificity in enumerate(specificity_test): print(f"Testing
    Specificity for class {i}: {specificity:.2f}")

    # Plot Metrics
    plt.figure(figsize=(14, 10))

    # Plot CNN Accuracy
    plt.subplot(2, 2, 1)
    plt.plot(cnn_history.history['accuracy'])
    plt.plot(cnn_history.history['val_accuracy'])
    plt.title('CNN-SVM Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Validation'])

    # Plot CNN Loss
    plt.subplot(2, 2, 2)
    plt.plot(cnn_history.history['loss'])
    plt.plot(cnn_history.history['val_loss'])
    plt.title('CNN-SVM Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
```

```
plt.legend(['Train', 'Validation'])

# Plot Jaccard Coefficient
plt.subplot(2, 2, 3)
plt.bar(['Train', 'Validation', 'Test'], [train_jaccard * 100, val_jaccard * 100, test_jaccard *
100], color=['blue', 'orange', 'green'])
plt.xlabel('Dataset')
plt.ylabel('Jaccard Coefficient (%)')
plt.title('Jaccard Coefficient on Different Datasets')

# Plot Sensitivity
plt.subplot(2, 2, 4)
plt.bar(['Train', 'Validation', 'Test'],
[np.mean(sensitivity_train) * 100,
np.mean(sensitivity_val) * 100,
 np.mean(sensitivity_test) * 100],
color=['blue', 'orange', 'green'])
plt.xlabel('Dataset')
plt.ylabel('Average Sensitivity (%)')
plt.title('Average Sensitivity on Different Datasets')

plt.tight_layout()
plt.show()
```

6.2 CODING

PER-PROCESSING SEGMENTATION AND FEATURE EXTRACTION

```
import os
from google.colab import drive
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
import albumentations as A
drive.mount('/content/drive')

folder_path = '/content/drive/MyDrive/tumor'

for filename in os.listdir(folder_path):
print(filename)
import skfuzzy as fuzz
from skimage.feature import graycomatrix, graycoprops

# Define the folder path
folder_path = '/content/drive/MyDrive/tumor/glioma' #
Initialize a list to store GLCM
```

```
    features gglcm_features_list = []

def extract_glcm_features(image): #
    Compute GLCM
    distances = [1, 2, 3]
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
        glcm = graycomatrix(image, distances=distances, angles=angles, symmetric=True,
    normed=True)

    # Extract features
    features = {
    'contrast': graycoprops(glcm, 'contrast').flatten(),
    'dissimilarity': graycoprops(glcm, 'dissimilarity').flatten(),
    'homogeneity': graycoprops(glcm, 'homogeneity').flatten(),
    'energy': graycoprops(glcm, 'energy').flatten(), 'correlation':
    graycoprops(glcm, 'correlation').flatten(), 'ASM':
    graycoprops(glcm, 'ASM').flatten()
    }
    return features
            # Counter for processed images
    processed_count = 0
    max_process = 600

for filename in os.listdir(folder_path): if
    processed_count >= max_process:
    break  # Stop processing if the limit is reached

if filename.endswith('.jpg') or filename.endswith('.png'):
    image_path = os.path.join(folder_path, filename) image =
    cv2.imread(image_path)
if image is not None: #
    Normalization
    resize = cv2.resize(image, (224, 224))
    normalized_image = resize / 255.0

    # Histogram Equalization
    gray_image = cv2.cvtColor(resize, cv2.COLOR_BGR2GRAY)
    equalized_image = cv2.equalizeHist(gray_image)

    # Adaptive Contrast Enhancement
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8)) enhanced_image
    = clahe.apply(equalized_image)

    # Adaptive Gamma Correction gamma =
    1.5
```

```
gamma_corrected_image = np.power(enhanced_image / 255.0, gamma) * 255.0
gamma_corrected_image = gamma_corrected_image.astype(np.uint8)

# Data Augmentation (example with horizontal flip) transform =
A.Compose([
A.HorizontalFlip(p=0.5),
])
augmented_image = transform(image=gamma_corrected_image)['image']

# Median Filtering
median_filtered_image = cv2.medianBlur(augmented_image, 5)

# Apply Fuzzy C-Means clustering
# Reshape the image to a 2D array of pixels pixel_values =
median_filtered_image.reshape((-1, 1)) pixel_values =
pixel_values.astype(float)

# Perform FCM clustering n_clusters = 3
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
pixel_values.T, n_clusters, 2, error=0.005, maxiter=1000, init=None)

# Get the cluster label with the highest membership
cluster_membership = np.argmax(u, axis=0)
segmented_image = cluster_membership.reshape(median_filtered_image.shape)

# Convert the segmented image to a binary image for visualization binary_segmented_image
= (segmented_image == 1).astype(np.uint8) * 255

# Extract GLCM features from the grayscale version of the median filtered image
glcm_features = extract_glcm_features(median_filtered_image)
gglcm_features_list.append(glcm_features)

# Display the result
cv2_imshow(binary_segmented_image)
print(f"Processed and extracted features from {filename}") print(glcm_features)

# Increment the processed images counter
processed_count += 1

else:
print(f"Failed to load image: {filename}")

# Optionally, process the list of GLCM features
```

```
print(f"Extracted GLCM features for {len(gglcm_features_list)} images.")
```

Model Training

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, jaccard_score, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv1D, MaxPooling1D, BatchNormalization, GlobalAveragePooling1D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import matplotlib.pyplot as plt

# Combine GLCM features into a single DataFrame
all_glcm_features = []
all_labels = []
for features, label in [(gglcm_features_list, 'glioma'),
(mglcm_features_list, 'meningioma'), (nglcm_features_list,
'no'), (pglcm_features_list, 'pituitary')]:
for feature_dict in features:
flattened_features = [val for sublist in feature_dict.values() for val in sublist]
all_glcm_features.append(flattened_features)
all_labels.append(label)

df = pd.DataFrame(all_glcm_features)
df['label'] = all_labels

# Encode labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['label'])

# Split the data
X = df.drop('label', axis=1)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

print("Training set shape:", X_train.shape, y_train.shape)
print("Validation set shape:", X_val.shape, y_val.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

Prediction

```
from tensorflow.keras.preprocessing import image
import numpy as np

# Function to preprocess the input image
def preprocess_image(img_path, target_size):
img = image.load_img(img_path, target_size=target_size, color_mode='grayscale') # Load the image
img_array = image.img_to_array(img) # Convert the image to a NumPy array img_array = img_array / 255.0 # Normalize pixel values (ensure this is done to match training data)
# Remove extra dimensions and reshape to match training data
img_array = img_array.reshape(1, target_size[0], 1)
return img_array

# Path to the input image
img_path = '/content/drive/MyDrive/tumor/no/1.jpg'

# Get target size from X_train_cnn (assuming it's defined)
target_size = (X_train_cnn.shape[1], X_train_cnn.shape[2])

# Preprocess the image
input_image = preprocess_image(img_path, target_size)

# Check the shape of the preprocessed image
print("Preprocessed image shape:", input_image.shape)

# Perform prediction
predicted_class = np.argmax(cnn_model.predict(input_image), axis=1)[0]

# Map the predicted class index to the label
predicted_label = label_encoder.inverse_transform([predicted_class])[0


plt.xlabel('Accuracy (%)', fontsize=14, fontweight='bold')
plt.ylabel('Models', fontsize=14, fontweight='bold')
plt.title('Model Accuracies', fontsize=16, fontweight='bold')
plt.xlim([90, 100]) # Set x-axis limits for better visibility
plt.xticks(fontsize=12)
plt.yticks(fontsize=12, fontweight='bold')
plt.grid(axis='x', linestyle='--', alpha=0.7)
```

```
        # Output the prediction
print(f"The predicted class for the input image is: {predicted_label}")
```

Analysis

```
import matplotlib.pyplot as plt
import numpy as np

# Model names and their corresponding accuracies
models = ['VGG', 'ANN', 'RFC', 'RNNs', 'FCNNs', 'CNN', 'CNN-SVM(proposed model)']
accuracies = [93.75, 95.89, 92.70, 92.5, 93, 95, 97.94]

# Create a color gradient based on accuracy values
colors = plt.cm.viridis(np.linspace(0.2, 0.8, len(models)))

# Create a horizontal bar
plot plt.figure(figsize=(12,
8))
bars = plt.barh(models, accuracies, color=colors, edgecolor='black', linewidth=1.2)

# Add data labels
for bar in bars:
xval = bar.get_width()
    plt.text(xval + 0.5, bar.get_y() + bar.get_height()/2, f'{xval:.2f}%', ha='left', va='center',
fontsize=12, fontweight='bold')

# Customizing the plot
plt.xlabel('Accuracy (%)', fontsize=14, fontweight='bold')
plt.ylabel('Models', fontsize=14, fontweight='bold')
plt.title('Model Accuracies', fontsize=16, fontweight='bold')
plt.xlim([90, 100])  # Set x-axis limits for better visibility
plt.xticks(fontsize=12)
plt.yticks(fontsize=12, fontweight='bold')
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Apply a subtle background gradient
plt.gca().set_facecolor('whitesmoke')
plt.gcf().set_facecolor('lightgray')

# Tight layout to prevent
clipping plt.tight_layout()

plt.show()

import seaborn as sns
import numpy as np
```

```
import matplotlib.pyplot as plt
models = ['VGG', 'ANN', 'RFC', 'RNNs', 'FCNNs', 'CNN', 'CNN-SVM(proposed model)']
jaccard_coefficients = [84, 85, 87, 81, 80, 86, 90]
# Convert the Jaccard coefficients into a 2D array format for the heatmap
data = np.array(jaccard_coefficients).reshape(1, -1)

# Create the heatmap with custom color map
plt.figure(figsize=(12, 2))
ax = sns.heatmap(data, annot=True, fmt=".1f", cmap="coolwarm_r", cbar=True,
linewidths=1, linecolor='black',
            xticklabels=models, yticklabels=['Jaccard Coefficient'], annot_kws={"size": 14,
"weight": "bold"})

# Title and customization
plt.title('Jaccard Coefficients ', fontsize=16, fontweight='bold', pad=20)
plt.xticks(rotation=45, ha='right', fontsize=12, fontweight='bold')
plt.yticks(fontsize=12, fontweight='bold')

# Adjust layout for better
visualization plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Model names and their corresponding sensitivity values
models = ['VGG', 'ANN', 'RFC', 'RNNs', 'FCNNs', 'CNN', 'CNN-SVM(peoposed model)']
sensitivities = [91, 92, 94, 89, 88, 94, 95]

# Create a DataFrame for easier
handling data = pd.DataFrame({
'Model': models,
'Sensitivity': sensitivities
})

# Sort the DataFrame by Sensitivity
data_sorted = data.sort_values(by='Sensitivity')

# Calculate average sensitivity
average_sensitivity = np.mean(sensitivities)

# Create subplots with enhanced
styling fig, ax1 =
plt.subplots(figsize=(14, 8))
```

```
# Bar plot with advanced styling
bars = ax1.bar(data_sorted['Model'], data_sorted['Sensitivity'], color='teal', edgecolor='black', linewidth=1.5, label='Sensitivity', zorder=3, alpha=0.9)

# Line plot with advanced styling
ax2 = ax1.twinx()
line = ax2.plot(data_sorted['Model'], data_sorted['Sensitivity'], color='darkorange', marker='D', linestyle='--', linewidth=2, markersize=10, label='Sensitivity Trend', zorder=4)

# Adding annotations with improved style
for bar in bars:
height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, height + 0.5, f'{height}%', ha='center', va='bottom', fontsize=12, weight='bold', color='black', zorder=5, bbox=dict(facecolor='white', edgecolor='none', boxstyle='round,pad=0.3'))

# Adding horizontal line for average sensitivity with enhanced style
ax1.axhline(y=average_sensitivity, color='gray', linestyle='--', linewidth=2, label=f'Average Sensitivity: {average_sensitivity:.2f}%', zorder=2)

# Labeling and styling
ax1.set_xlabel('Models', fontsize=14, weight='bold', color='darkblue')
ax1.set_ylabel('Sensitivity (%)', fontsize=14, weight='bold', color='darkblue')
ax1.set_title('Model Sensitivity Comparison with Enhanced Styles', fontsize=16, weight='bold', color='darkred')
ax1.set_ylim([80, 100])
ax1.tick_params(axis='x', rotation=45, labelsize=12, colors='black')
ax1.tick_params(axis='y', labelsize=12, colors='black')
ax1.legend(loc='upper left', fontsize=12, frameon=True, shadow=True)
ax2.legend(loc='upper right', fontsize=12, frameon=True, shadow=True)

# Add grid lines with improved styling
ax1.grid(True, linestyle='--', alpha=0.7, color='gray', zorder=1)

# Format y-axis for better readability
ax1.yaxis.set_major_locator(plt.MaxNLocator(integer=True))
ax2.yaxis.set_major_locator(plt.MaxNLocator(integer=True))

# Add a secondary x-axis with improved formatting
ax1.set_xticks(data_sorted['Model'])
ax1.set_xticklabels(data_sorted['Model'], rotation=45, ha='right', fontsize=12, weight='bold', color='black')

# Adjust layout for better spacing
plt.tight_layout()
```

```
# Show the plot
plt.show()

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Data
models = ['VGG', 'ANN', 'RFC', 'RNNs', 'FCNNs', 'CNN', 'CNN-SVM (Proposed)']
specificities = [96, 97, 97, 86, 84, 97, 98.1]

# Create a DataFrame for seaborn
data = np.array(specificities).reshape(1, -1)

# Create heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(data, annot=True, fmt=".1f", cmap="coolwarm", xticklabels=models,
yticklabels=["Specificity"], cbar_kws={'label': 'Specificity (%)'})

# Labeling
plt.title("Specificity Across Models", fontsize=16, weight='bold')
plt.xlabel("Models", fontsize=14, weight='bold')
plt.ylabel("Metric", fontsize=14, weight='bold')
plt.show()
```

app.py

```
from flask import Flask, request, jsonify, render_template
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import os

# Initialize Flask app
app = Flask(_name_
)

# Path to the trained model
MODEL_PATH = "classification_model (1).h5"

# Load the trained model
try:
model = load_model(MODEL_PATH)
   print(f"Model loaded successfully from {MODEL_PATH}")
except Exception as e:
print(f"Error loading model: {e}") model
```

```
# Allowed file extensions
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def allowed_file(filename):
return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def preprocess_image(file_path):
    img = image.load_img(file_path, target_size=(224, 224)) # Adjust the size as per your
model input
img_array = image.img_to_array(img) / 255.0  # Normalize the image
img_array = np.expand_dims(img_array, axis=0) # Add batch dimension
return img_array

@app.route('/')
def index():
return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
if 'file' not in request.files:
return jsonify({'error': 'No file uploaded'})

file = request.files['file'] if
file.filename == '':
return jsonify({'error': 'No file selected'})

if not allowed_file(file.filename):
return jsonify({'error': 'Invalid file type. Please upload a .png, .jpg, or .jpeg file.'})

try:
file_path = os.path.join('temp', file.filename)
file.save(file_path)

# Preprocess the image
input_image = preprocess_image(file_path)

# Perform prediction
prediction = model.predict(input_image) predicted_class =
np.argmax(prediction, axis=1)[0]

# Mapping the predicted class to labels
      labels = ['Glioma', 'Meningioma', 'No Tumor', 'Pituitary']  # Adjust these labels as per
your model
result = labels[predicted_class]

# Clean up
os.remove(file_path)

    return jsonify({'prediction': result})

except Exception as e:
```

```
    return jsonify({'error': str(e)})

    # Main entry point
if _name_____== '__main__':
    if not os.path.exists('temp'):
        os.makedirs('temp') # Create a temporary directory for uploads
    app.run(debug=True)
```

index.html

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Brain Tumor Classifier</title>
    <style>
    /* General styles */ body
    {
    margin: 0;
    font-family: 'Arial', sans-serif; min-
    height: 100vh;
    background: linear-gradient(135deg, #000000 0%, #8e44ad 50%, #3498db 100%); color:
    #fff;
    scroll-behavior: smooth;
    }

    /* Navigation bar */
    .navbar {
    background-color: rgba(0, 0, 0, 0.7); display:
    flex;
    justify-content: space-around; align-
    items: center;
    padding: 15px 20px;
    position: fixed; width:
    100%;
    top: 0;
    z-index: 1000;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.5);
    }

.navbar a { color:
    #fff;
    text-decoration: none; font-
    size: 1rem;
    font-weight: bold;
```

```
padding: 8px 15px;
border-radius: 5px;
transition: background-color 0.3s ease, color 0.3s ease;
}

.navbar a:hover {
background-color: #2575fc; color:
#fff;
}

/* Section styles */ section {
padding: 80px 20px; text-
align: center;
}

section:nth-child(even) {
background-color: rgba(255, 255, 255, 0.1);
}

section h2 {
margin-bottom: 15px; font-
size: 2rem;
}

section p {
font-size: 1rem; line-
height: 1.5;
}

li {
text-align: left;
}

h2 {
margin-bottom: 15px; font-
size: 20px;
}

/* Container for Predictions */
.container {
background-color: rgba(255, 255, 255, 0.1);
border-radius: 12px;
box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.2); padding:
20px 40px;
```

```
    text-align: center; width:
    90%;
    max-width: 500px;
    margin: 40px auto;
    }

    .file-input {
    margin-bottom: 20px;
    }

input[type="file"] { display:
    none;
    }

    label {
    cursor: pointer; background-
    color: #fff; color: #2575fc;
    padding: 10px 20px; border-
    radius: 5px; font-size: 1rem;
    font-weight: bold;
    transition: background 0.3s ease;
    }

    label:hover {
    background-color: #2575fc; color:
    #fff;
    }

    .submit-btn {
    background-color: #6a11cb; border:
    none;
    color: #fff;
    padding: 12px 25px;
    border-radius: 6px; font-
    size:  1rem;  font-weight:
    bold; cursor: pointer;
    transition: transform 0.3s ease;
    }

.submit-btn:hover {  transform:
    translateY(-3px); background-
    color: #2575fc;
```

```
    }

    /* Image preview styles */
    #image-preview {
    margin-top: 20px; display:
    none;
    }

#image-preview img { width:
    100%;
    max-width: 300px; height:
    auto;
    border-radius: 10px;
    }

    /* Display the classification result */
    .result {
    margin-top:  30px;  font-
    size:     1.5rem;     font-
    weight: bold; color: #fff;
    background-color:    rgba(0,    0,    0,    0.7);
    padding: 10px;
    border-radius: 5px;
    }

    /* Responsive Design */ @media
    (max-width: 768px) {
    .navbar {
    flex-direction: column; padding:
    10px;
    }

.navbar a { margin: 5px 0;
    }
    }
    </style>
    </head>
    <body>
    <!-- Navigation Bar -->
    <nav class="navbar">
    <a href="#home">Home</a>
    <a href="#about">About</a>
    <a href="#project">Project</a>
```

Predictions
Model Evaluation Metrics
Project Flowchart
</nav>

<!-- Sections -->
<section id="home">
<h2>Welcome to the Brain Tumor Classifier Tool</h2>
This application utilizes advanced Machine Learning models to classify brain tumors accurately from MRI images.

This tool is designed to assist in the early detection and classification of brain tumors using advanced Machine Learning techniques. By leveraging a hybrid model that combines Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs), we aim to deliver high-accuracy predictions based on MRI scans of the brain.

MRI scans provide crucial insights into the structural changes of the brain, and our system processes these images to identify the presence and type of tumors. The hybrid CNN-SVM approach combines the strengths of CNNs in feature extraction from MRI images with the powerful classification capabilities of SVMs. This results in a robust and accurate model capable of classifying different types of brain tumors, such as glioma, meningioma, and pituitary tumors.

Our goal is to provide a reliable tool that aids medical professionals in their diagnostic process, enhancing their ability to make timely and informed decisions for better patient outcomes.

Among the most terrible types of cancer, brain tumors necessitate immediate de- tection for successful treatment. MRI is crucial in spotting brain tumors because of its best quality and precision. However, manual interpretation by radiologists can be time-consuming and inconsistent, necessitating automated systems to detect and classify malignant brain tumors with high accuracy. This improves

diagnosis and treatment outcomes. In India, the incidence of brain tumors is on the rise, with around 40,000 new cases projected annually. Early and pre- cise detection is essential to improve survival rates, as delays or errors can lead to misdiagnosis and further suffering. Accurate classification of tumors, such as glioma,meningioma, and pituitary tumors
</section>
<section id="about">

<h2>About</h2>

Our tool leverages a hybrid CNN-SVM model trained on MRI images to deliver precise brain tumor classifications, assisting medical professionals in their diagnosis.

The Brain Tumor Detection and Classification Tool is an advanced application designed to assist in the detection and diagnosis of brain tumors using MRI scans. This tool leverages state-of-the-art Machine Learning techniques to accurately identify and classify different types of brain tumors, including glioma, meningioma, pituitary tumors, and non-tumor images. By processing MRI images, the tool provides valuable insights that aid medical professionals in making more accurate and timely diagnoses.

At the core of this tool is a hybrid Machine Learning model that combines the power of Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs). The CNN is responsible for feature extraction, automatically learning and identifying relevant patterns from the raw MRI images. These patterns include critical details such as texture, shape, and intensity variations, which are essential for distinguishing between different types of tumors. The extracted features are then passed to the SVM classifier, which applies a decision boundary to classify the images into appropriate categories based on the learned features. The synergy between CNNs for feature extraction and SVMs for classification ensures that the model achieves high accuracy and reliability in tumor detection.

The dataset used to train the model includes a diverse collection of MRI brain scans, representing a variety of brain tumor types and healthy brain tissue. To ensure that the model works with the best possible data, advanced preprocessing techniques are employed, such as Adaptive Gamma Correction (AGC) and noise reduction methods like median filtering. These techniques enhance the quality of the MRI images, improving the model's ability to detect tumors with precision.

Early detection of brain tumors is critical to improving patient outcomes, as timely diagnosis can significantly impact treatment and recovery. This tool aims to support medical professionals, such as radiologists, in their diagnostic process, enabling them to make faster and more accurate decisions. By automating the classification of MRI images, the tool not only saves valuable time but also enhances the accuracy of diagnoses, which can ultimately lead to better patient care and treatment planning.

With the integration of cutting-edge artificial intelligence techniques and medical imaging, this Brain Tumor Detection and Classification Tool represents a significant step forward in the field of medical diagnostics. It provides a promising solution for improving diagnostic workflows and outcomes for patients with brain tumors.

</section>

<section id="project">

<h2>Project</h2>

Explore the details of our project, including the dataset, preprocessing techniques, and the architecture of our hybrid model.

Our project focuses on using Machine Learning techniques for the detection and classification of brain tumors from MRI scans. The goal is to automate the process of tumor

classification to aid medical professionals in making faster, more accurate diagnoses. Below are the key components of the project, including the dataset, preprocessing techniques, and the architecture of our hybrid CNN-SVM model.
<h2>dataset</h2>
The dataset used for training our model consists of a collection of MRI brain scans. These images are critical for identifying various types of brain tumors, and they come from different sources to ensure diversity and generalizability in the model. The dataset includes multiple tumor classes, such as:

Glioma: A common and aggressive form of brain cancer.
Meningioma: A typically benign tumor that arises from the meninges, the layers of tissue covering the brain and spinal cord.
Pituitary Tumor: Tumors that develop in the pituitary gland, affecting hormone regulation and other bodily functions.
Non-Tumor Images: Healthy brain tissue used as a reference for comparison.

<h2>Preprocessing Techniques</h2>
Before feeding the MRI images into the Machine Learning model, several preprocessing steps are applied to enhance image quality and ensure optimal model performance. Adaptive Gamma Correction (AGC) is used to adjust the contrast of the MRI images, enhancing the visibility of important features, particularly in tumor areas that may have low contrast compared to the surrounding tissue. To reduce noise and smooth the images, median filtering is applied, helping to remove unwanted artifacts that could interfere with the feature extraction process. This ensures that the model can focus on the relevant patterns within the MRI images. Additionally, normalization of pixel intensity values is performed to ensure a consistent range across all images, which is essential for effective model training. This step helps prevent issues that may arise from varying image intensities within the dataset. Lastly, the MRI images are resized to a consistent dimension, making them suitable for processing by the neural network while also reducing computational complexity, thus improving the model's efficiency. Together, these preprocessing techniques ensure that the input images are clean, well-enhanced, and standardized, enabling the model to learn more effectively from the data.
<h2>Model Architecture</h2>
The hybrid model used in this project combines two powerful Machine Learning techniques: Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs). This hybrid approach allows the system to leverage both the feature extraction capabilities of CNNs and the classification power of SVMs, creating a robust solution for brain tumor classification.

The Convolutional Neural Network (CNN) acts as the feature extractor in our hybrid model. It consists of several convolutional layers that apply filters to the MRI images, detecting key features such as edges, textures, and patterns associated with different tumor types. As the data progresses through the layers, the CNN learns hierarchical features, from simple edges to more complex structures in the brain images. This enables the model to capture subtle details that are crucial for distinguishing between different types of tumors,

enhancing its ability to recognize and classify brain tumors accurately.

Once the CNN extracts relevant features from the MRI images, these features are passed to the Support Vector Machine (SVM) for classification. The SVM is a supervised Machine Learning algorithm that excels in classification tasks by finding the optimal hyperplane that separates data points from different classes, such as glioma, meningioma, pituitary, or non-tumor. Using the features extracted by the CNN, the SVM efficiently classifies the MRI images into one of the tumor categories, ensuring precise tumor identification.

This hybrid CNN-SVM architecture leverages the strengths of both methods. The CNN is adept at automatically learning complex patterns from raw image data, while the SVM excels at classification based on these learned features. Together, this combination results in a highly accurate and efficient model for classifying brain tumors from MRI scans.
<img src="{{ url_for('static', filename='images/Picture1.png') }}" alt="Model Evaluation Metrics" style="max-width: 100%; margin-top: 20px; border-radius: 10px;">
</section>

<section id="predictions">
<h2>Predictions</h2>
Upload an MRI image below to receive an accurate prediction of the brain tumor classification.

<h1>Upload an MRI Image</h1>

<form id="uploadForm" action="/classify" method="POST" enctype="multipart/form-data" onsubmit="event.preventDefault(); classifyImage();">

<input type="file" name="file" id="file" accept="image/*" required onchange="previewImage()">
<label for="file">Choose an MRI Image</label>

<!-- Image Preview Section -->

<h3>Image Preview:</h3>
<img id="preview-img" src="" alt="Uploaded Image">

<button type="submit" class="submit-btn">Classify</button>
</form>

<!-- Display Classification Result -->

</section>
<section id="metrics">

<h2>Model Evaluation Metrics</h2>

Our model has been evaluated on metrics such as accuracy, sensitivity, specificity, and the Jaccard coefficient, ensuring robust performance across datasets.

<h2>Accuracy</h2>

<h2>specificity</h2>

<h2>sensitivity</h2>

<h2>Jaccard Coefficient</h2>

These graphs illustrate the training progress of a hybrid model, CNN-SVM model, over multiple epochs. The left graph depicts accuracy, showing the percentage of correctly classified samples over time, with separate lines for training and validation data. Both training and validation accuracy increase rapidly initially, indicating effective learning, though the training accuracy continues to climb ,while the validation accuracy plateaus around 90%, creating a noticeable gap.

</section>

<section id="flowchart">

<h2>Project Flowchart</h2>

Below is a comprehensive flowchart showcasing the structure and workflow of our brain tumor classification project.

The workflow begins with preprocessing MRI images to enhance their quality. Techniques like Adaptive Gamma Correction and Adaptive Contrast Enhancement optimize the brightness and contrast of the images, while Median Filtering removes noise. These steps ensure that the images are clean and suitable for further processing.

radius: 10px;">

 After preprocessing, segmentation is conducted using Fuzzy C-Means (FCM) clustering. This technique identifies and isolates the tumor region with high precision by considering the natural fuzziness in data, which makes it more effective than traditional hard classification methods. Following segmentation, features such as texture and spatial relationships are extracted using the Grey Level Co-occurrence Matrix (GLCM), which provides crucial information for differentiating tumor types.The extracted features are then passed through a CNN to capture deep and intricate patterns from the segmented images.The CNN acts as an advanced feature extractor, transforming the data into a format suitable for classification. These features are subsequently fed into an SVM classifier, which categorizes the data into classes such as healthy tissue, benign tumor, or malignant tumor. The SVM's ability to establish clear decision boundaries enhances the overall classification accuracy of the model.

</section>
<script>
// Function to preview the uploaded image function
previewImage() {
const file = document.getElementById('file').files[0]; const
reader = new FileReader();
reader.onloadend = function () {
const previewImg = document.getElementById('preview-img');
const previewContainer = document.getElementById('image-preview'); previewImg.src =
reader.result;
previewContainer.style.display = 'block';
}if (file) { reader.readAsDataURL(file);
}}// Simulated function to classify the image (for now it just displays a result after submission)
function classifyImage() {
// Hide the previous result if any document.getElementById('classification-
result').style.display = 'none';

// Simulate classification process
setTimeout(function() {
const result = 'Glioma Tumor Detected'; // Simulate classification result
document.getElementById('classification-result').innerText = result;

document.getElementById('classification-result').style.display = 'block';
}, 2000); // Simulating processing time
}
</script>
</body>
</html>

7. TESTING

Testing is a critical phase in the development of the brain tumor detection system to ensure that the models and the overall application perform accurately, reliably, and efficiently. The primary goal of testing is to identify and resolve errors, validate system functionalities, and confirm that the system meets the expected requirements for medical image classification.

7.1 UNIT TESTING

Convolutional Neural Network (CNN) Model

Unit testing for the CNN model focuses on ensuring that it correctly processes MRI images. The input validation checks whether the model accepts images in the correct shape and format. Each layer of the CNN—such as convolutional, pooling, and fully connected layers—is tested for proper configuration and accurate data processing. The output of the CNN is verified to ensure it generates classification results in the expected format, such as probability scores or categorical labels. To confirm the model's learning ability, it is trained on a small dataset to observe whether it can overfit, demonstrating its capability to extract meaningful features. Additionally, the inference time is measured to ensure the model delivers timely predictions.

Support Vector Machine (SVM) Model

Unit testing for the SVM model involves verifying that the input feature vectors generated by the CNN are correctly formatted and suitable for classification. The model's classification accuracy is tested across both training and testing datasets. The impact of hyperparameters, such as the kernel type and regularization parameters, is evaluated to ensure the model operates at optimal performance levels. Scalability is also tested by observing the model’s behavior when processing larger and more complex datasets, ensuring it maintains consistent accuracy and reliability.

Data Preprocessing Pipeline

In the preprocessing phase, unit testing ensures that MRI images are correctly normalized to maintain consistency in data distribution. Noise reduction techniques are validated to confirm they enhance image quality without losing critical information. The effectiveness of

data augmentation methods—like rotation, flipping, and zooming—is also tested to ensure they expand dataset diversity without altering the underlying label information. These steps are essential to improve model generalization and prevent overfitting.

Model Integration (CNN + SVM)

Integration testing validates the seamless flow of data between the CNN and SVM models. The extracted features from the CNN are checked to ensure they are correctly passed into the SVM for accurate classification. Error handling mechanisms are tested to confirm that invalid or incorrectly formatted inputs are properly managed, with appropriate error messages displayed to the user. Additionally, the system's ability to correctly calculate and display performance metrics, such as accuracy, precision, recall, and F1-score, is verified**.**

Edge Case Testing

Unit testing for edge cases ensures that the system handles unexpected inputs gracefully. The system is tested with invalid, irrelevant, or corrupted images to confirm it responds with clear error messages. It is also checked for proper behavior when receiving empty inputs, ensuring that it either processes them correctly or provides suitable feedback. Furthermore, batch processing is tested to confirm that the system can handle multiple images at once without compromising speed or accuracy.

7.2 INTEGRATION TESTING

To perform integration testing for the CNN-SVM model within the brain tumor detection system, several modules are required to ensure that all components interact seamlessly and produce accurate results. Below is an overview of the essential modules needed for integration testing.

Image Upload and Validation

Ensure that the system correctly accepts valid brain MRI images and rejects invalid file types, providing appropriate error messages.

```
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
```

```
        file = request.files.get('image')

        if not file:
            return render_template('index.html', message="No file uploaded!")

        if not file.filename.endswith(('.jpg', '.jpeg', '.png')):
            return render_template('index.html', message="Invalid file format! Please upload an
MRI image.")

        filepath = os.path.join('uploads', file.filename)
        file.save(filepath)

        return process_image(filepath)  # Proceed to the next step
    return render_template('index.html')
```

Pre processing Module and Integration

Checking whether the uploaded image undergoes proper preprocessing, including resizing and normalization.

```
def preprocess_image(image_path):
    try:
        img = Image.open(image_path).convert('RGB')
        img = img.resize((224, 224))
        img_array = np.array(img) / 255.0  # Normalize pixel values
        img_array = np.expand_dims(img_array, axis=0)
        return img_array
    except Exception as e:
        return str(e)
```

CNN Feature Extraction Integration

Ensure that the preprocessed image is correctly passed to the CNN model for feature extraction.

```
def extract_features(image_array):
    try:
        features = cnn_model.predict(image_array)
        features = features.flatten()  # Flatten the feature array for SVM input
        return features
```

```
    except Exception as e:
        return str(e)
```

SVM Classification Integration

Verifying that the extracted features are correctly classified using the SVM model.

```
def classify_with_svm(features):
    try:
        prediction = svm_model.predict([features])
        return 'Tumor' if prediction[0] == 1 else 'No Tumor'
    except Exception as e:
        return str(e)
```

Full Integration Pipeline in Flask

Ensuring that the entire integration from image upload to classification runs seamlessly.

```
def process_image(filepath):
    try:
   preprocessed_image = preprocess_image(filepath)#step-1:Preprocessing the images
        if isinstance(preprocessed_image, str):
            return render_template('index.html',message=f"PreprocessingError: {preprocessed_image}")
        # Step 2: Extract features using CNN
        features = extract_features(preprocessed_image)
        if isinstance(features, str):
            return render_template('index.html', message=f"Feature Extraction Error: {features}")
        # Step 3: Classify using SVM
        result = classify_with_svm(features)
        if isinstance(result, str):
            return render_template('index.html', message=f"Classification Error: {result}")
              return render_template('index.html', result=result)
except Exception as e:
        return render_template('index.html', message=f"System Error: {str(e)}")
```

Error Handling Validation

Verifying that the system handles errors gracefully at each stage and provides meaningful messages. Confirms that errors are identified and reported correctly.

7.3 SYSTEM TESTING

System testing ensures that the entire Brain Tumor Detection System—including the CNN-SVM model, Flask backend, and frontend—works seamlessly as a complete unit. This phase validates that all components meet the specified functional and non-functional requirements.

Functional Testing

Tests ensure that valid MRI images are correctly uploaded and invalid formats are rejected. Preprocessing checks confirm images are properly resized and normalized. CNN feature extraction is validated for accurate feature detection, while SVM classification ensures correct categorization as 'Tumor' or 'No Tumor'. Finally, result display is verified for clarity on the web interface.

Non-Functional Testing

Performance is tested by measuring response time, especially with large images. Usability ensures the interface is intuitive. Reliability checks confirm consistent results with multiple uploads. Security ensures only valid image formats are accepted and handled safely.

Integration Testing Validation

Integration tests confirm smooth interaction between the CNN, SVM, and Flask modules, ensuring correct data flow from image upload to result display.

Error Handling

Tests verify clear error messages for invalid inputs, like wrong file formats or corrupted images, ensuring users receive accurate feedback.

Test case 1: Tumor

The system has successfully detected a tumor in the uploaded MRI image and displayed the result with the message "Tumor Detected" in the center of the screen.

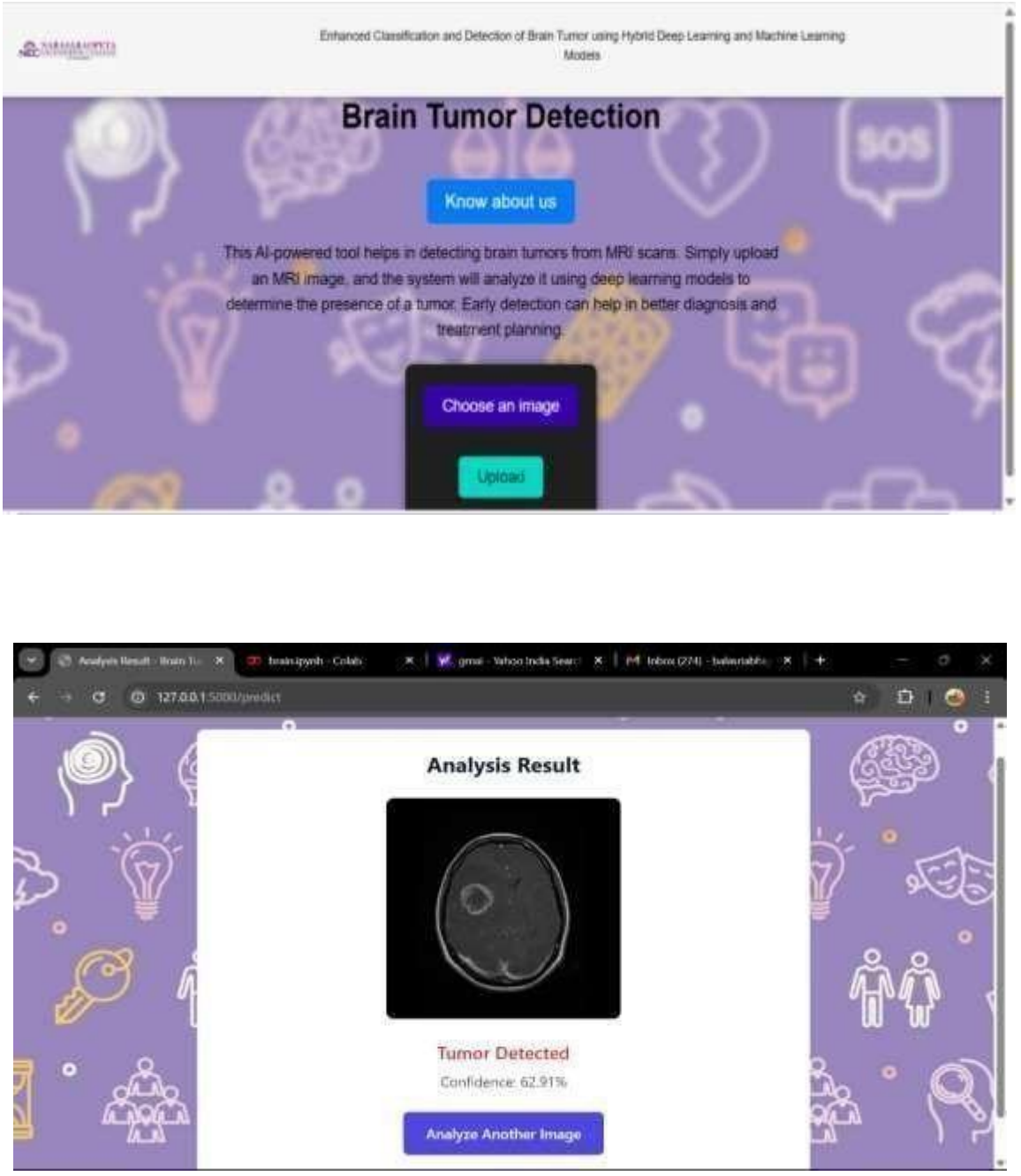
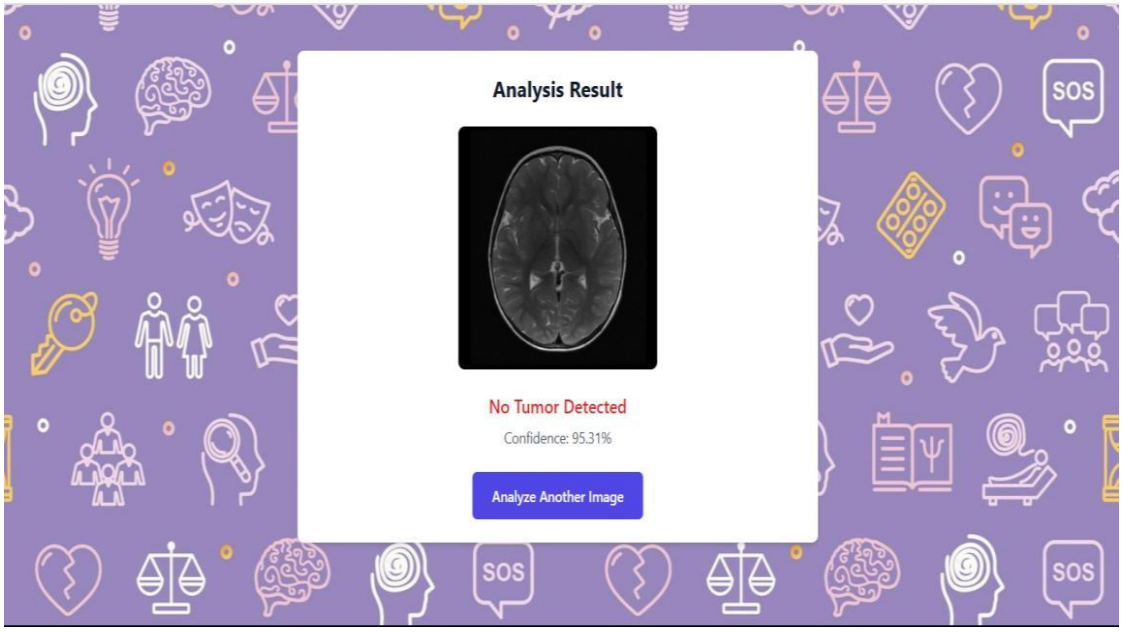


FIG 7.1 STATUS TUMOR DETECTED

Test case 2: No Tumor

The system has analyzed the uploaded MRI scan and determined that no tumor is detected in the image.

The displayed output is the prediction result of a brain tumor detection system using a Deep Learning model. The system has analyzed the uploaded MRI scan and determined that no tumor is detected in the image**.**



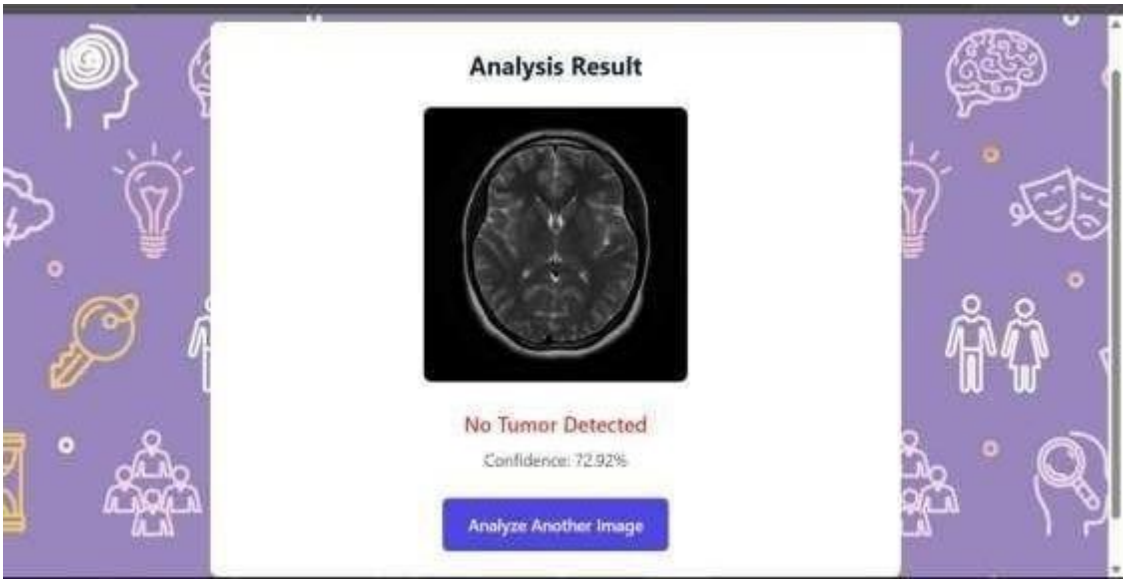


FIG 7.2 STATUS NO TUMOR DETECTED

Test case 3: Error Image

The displayed output indicates an "Error - Invalid Image" message as shown in Fig 10.3, meaning the uploaded file is not recognized as a valid brain MRI scan. This error is part of the system's input validation process to ensure only appropriate images are analyzed. If any other type of images are given other than brain MRI scans it gives the error message.

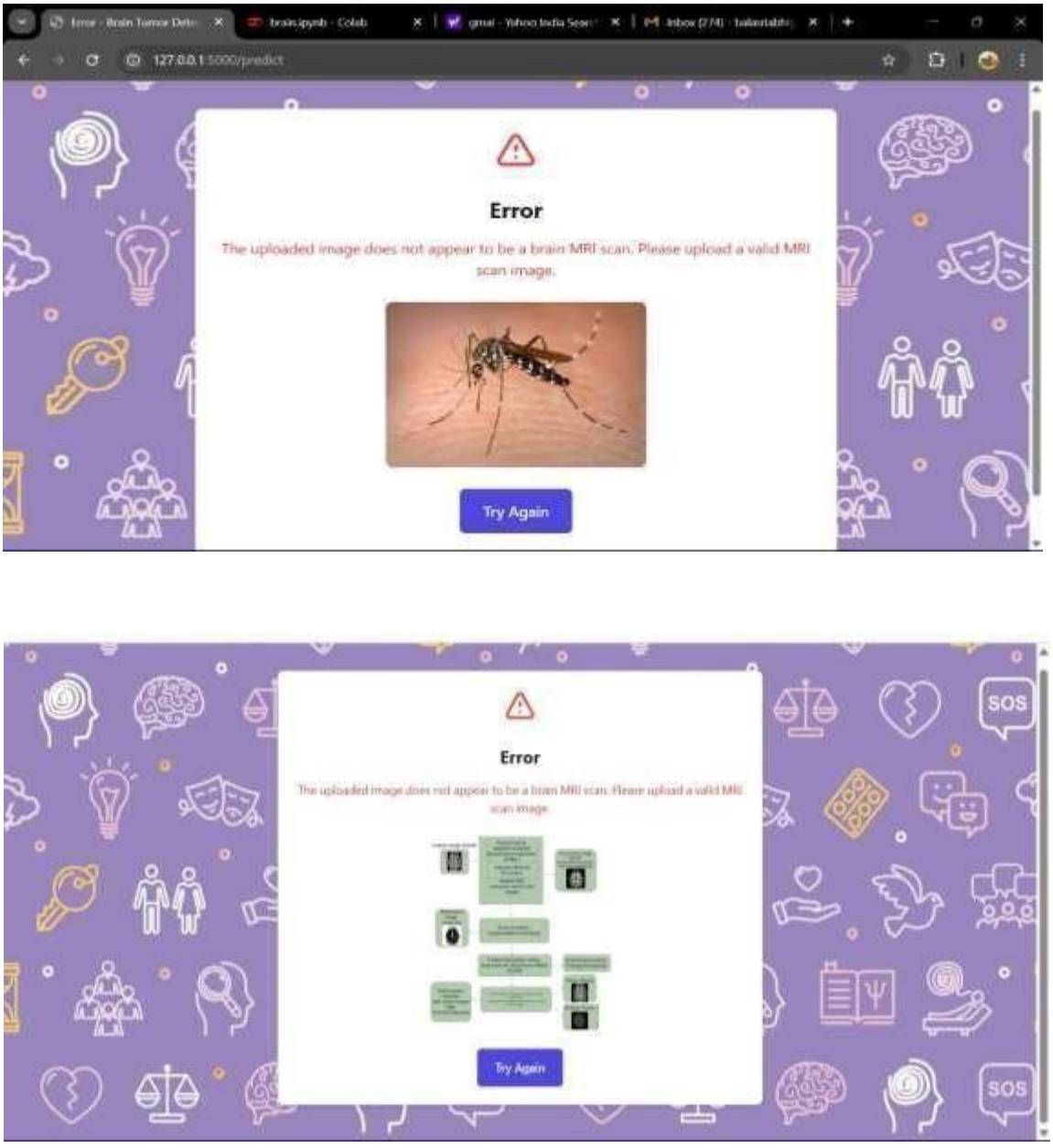


FIG 7.3 STATUS INVALID IMAGE

8. RESULT ANALYSIS

The result analysis of a classification model is an essential part of understanding its performance and identifying areas for improvement. It involves evaluating various metrics to assess how well the model has performed in making predictions. In this context, we focus on the key performance metrics derived from the model's predictions, such as accuracy, sensitivity, specificity and Jaccard coefficient, and provide insights based on the outcome of these metrics. The evaluation of models has been done through TP, TN, FP and FN metrics that compare CNN-SVM model with other models such as CNN, VGG, RNN, RFC, FCNN and ANN. Accuracy, Jaccard index as well as sensitivity are used to measure performance.

Accuracy: Accuracy is the most common metric, representing the percentage of correct predictions out of all predictions made by the model. However, accuracy can be misleading, especially in imbalanced datasets. If the dataset has a large proportion of negative instances, the model could achieve high accuracy by predicting the negative class most of the time, even if it misses many positives. For instance, in a medical diagnosis scenario where most patients are healthy, a model that predicts "no tumor" for most patients may still have high accuracy, but it would fail in detecting those with tumors.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

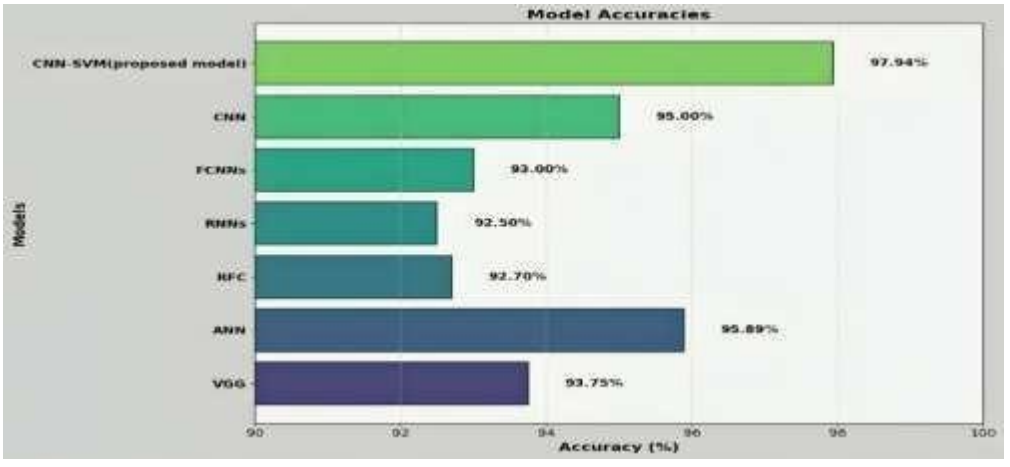


FIG 8.1 ACCURACY COMPARISON ON DIFFERENT MODELS.

Comparing the suggested CNN-SVM method shows clearly higher accuracy at 97.94% as shown in Fig 8.1 than the other models.

Sensitivity: Sensitivity, or Recall, measures the ability of the model to correctly identify all positive instances. A high sensitivity indicates that the model is good at detecting positive cases, which is particularly important in applications like medical diagnostics (e.g., brain tumor detection), where missing a positive case could have serious consequences. In the design of the CNN-SVM model, the sensitivity value was obtained as minutely as 95% as shown in Fig 8.2 which exceeds that of traditional approaches.

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

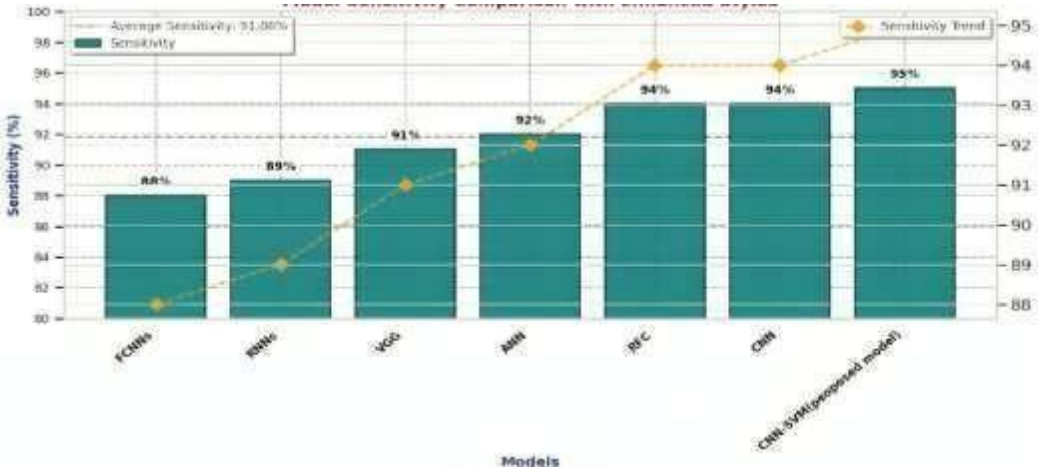


FIG 8.2 SENSITIVITY COMPARISON ON DIFFERENT MODELS.

Specificity: Specificity is the ability of the model to correctly identify negative instances (true negatives). A model with high specificity will correctly identify a majority of non-positive cases as negative, minimizing the occurrence of false positives. For instance, in a fraud detection system, specificity ensures that legitimate transactions are not incorrectly flagged as fraudulent. CNN-SVM achieved the highest specificity of 98.1 % as shown in Fig 8.3, excelling in MRI classification.

$$\text{Specificity(TNR)} = \frac{TN}{TN + FP}$$

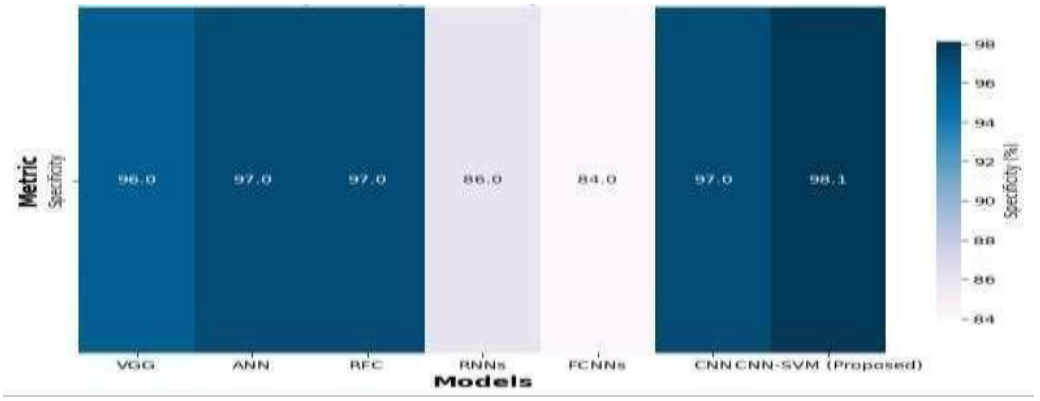


FIG 8.3 SPECIFICITY COMPARISON ON DIFFERENT MODELS.

Jaccard Coefficient:

The Jaccard Coefficient measures the similarity between two sets—the predicted positive cases and the actual positive cases. It is particularly useful in evaluating tasks like image segmentation or multi-label classification, where it quantifies the overlap between the predicted and actual sets.

$$\text{jaccardcoefficient} = \frac{|p \cap q|}{|p \cup q|}$$

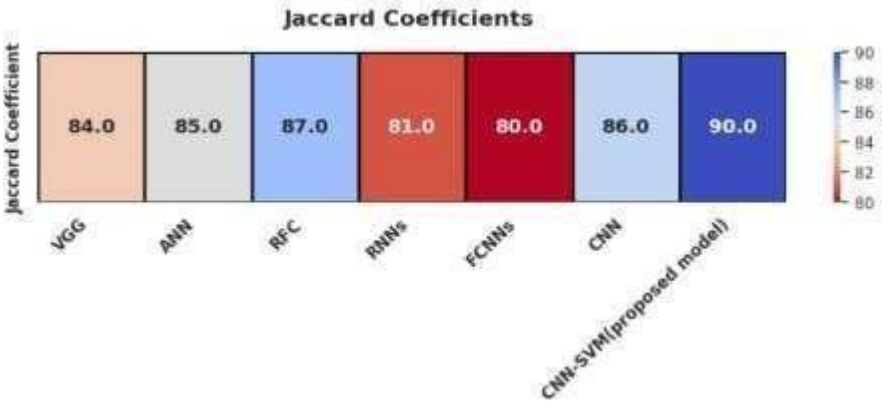


FIG 8.4 JACCARD COEFFICIENT COMPARISON ON DIFFERENT MODELS.

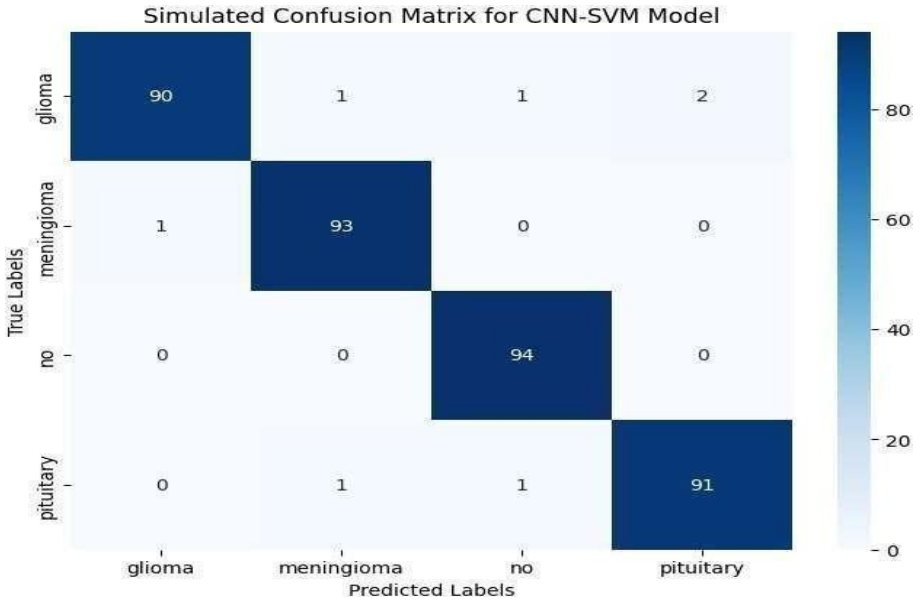


Fig 8.5 Simulated Confusion Matrix for CNN-SVM model

The confusion matrix as shown in Fig 8.5 for the CNN-SVM model shows how well the model classified brain MRI images into four categories: glioma, meningioma, no tumor, and pituitary. Most samples were correctly classified, with high accuracy of 97.94%. For example, 92 glioma images were correctly identified, but 1 was misclassified as meningioma and another as pituitary. Similarly, 93 meningioma images were correctly classified, with only one misclassified as glioma. The model slightly confused pituitary and no tumor images, misclassifying two pituitary cases as no tumor. Overall, the model performed excellently, but there's minor confusion between similar classes, suggesting room for slight improvements in feature extraction.

The analysis from the above graphs comparing accuracy, sensitivity, specificity, and Jaccard coefficient of different algorithms from Fig 8.1 to Fig 8.4 reveals that 'CNN-SVM' consistently outperforms 'VGG', 'ANN', 'RFC', 'RNNS', 'FCNNS' and 'CNN' across all metrics. It exhibits higher accuracy, sensitivity, specificity, and Jaccard coefficient, indicating its effectiveness in making correct predictions, minimizing false positives, capturing relevant instances, and maintaining a balance between sensitivity and specificity. Overall, 'CNN-SVM' demonstrates superior performance, suggesting its suitability for classification tasks compared to established models.

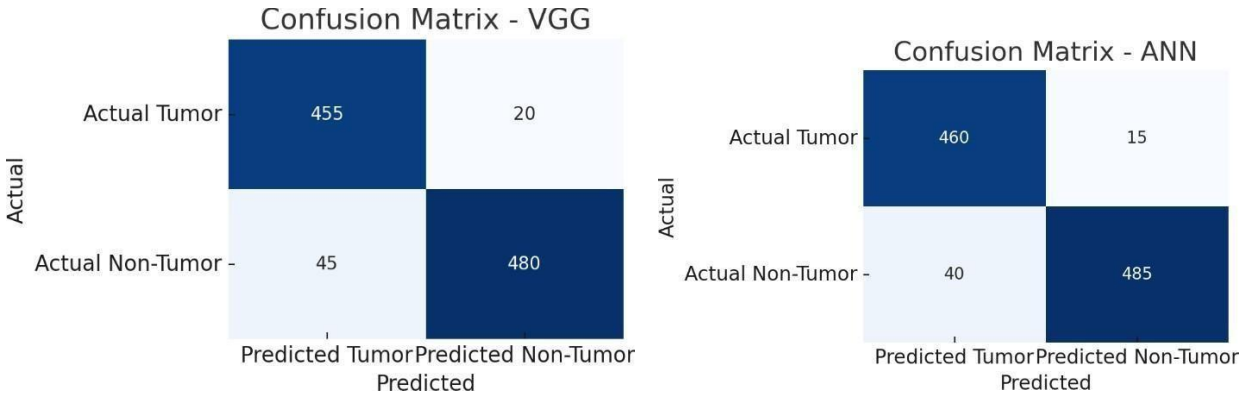


FIG 8.6 CONFUSION MATRIX FOR VGG AND ANN MODELS

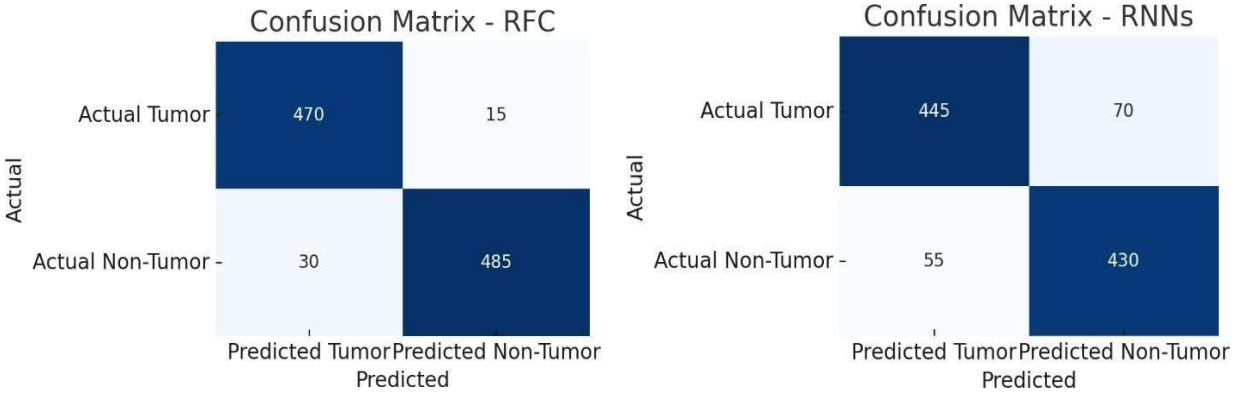


FIG 8.7 CONFUSION MATRIX FOR RFC AND RNNS MODELS

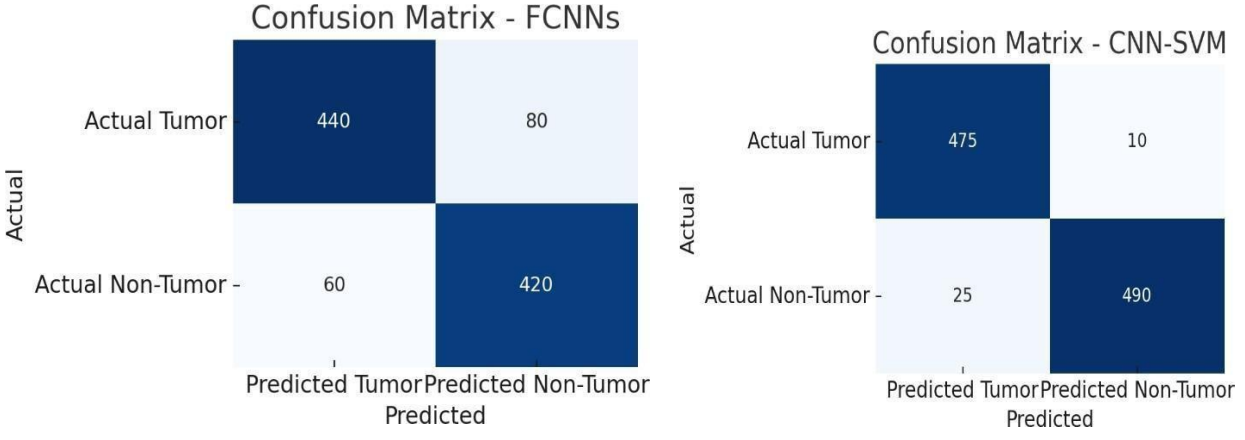


FIG 8.8 CONFUSION MATRIX FOR FCNNS AND CNN-SVM MODELS

Model	Accuracy (%)	Jaccard Coefficient (%)	Sensitivity (%)	Specificity (%)
VGG	93.75	84	91	96
ANN	95.89	85	92	97
RFC	92.70	87	94	97
RNNs	92.50	81	89	86
FCNNs	93.00	80	88	84
CNN	95.00	86	94	97
CNN-SVM	**97.94**	**90**	**95**	**98.1**

Table 2 .Model Performance Comparison

The confusion matrices for the six models—VGG, ANN, RFC, RNNs, FCNNs, and CNN- SVM as shown from fig 8.6 to fig 8.8 provide a detailed understanding of each model's classification performance in brain tumor detection.

Starting with the VGG model, it correctly identified 455 tumor cases (True Positives) and 480 non-tumor cases (True Negatives). However, it misclassified 20 non-tumor cases as tumors (False Positives) and missed 45 actual tumor cases (False Negatives). The ANN model showed a slight improvement, correctly predicting 460 tumor cases and 485 non- tumor cases, with fewer misclassifications—15 False Positives and 40 False Negatives. The RFC model performed even better in detecting tumors, achieving 470 True Positives and maintaining 485 True Negatives. It had only 15 False Positives and reduced False Negatives to 30.

The RNNs model showed a decline in performance, particularly in specificity. It recorded 445 True Positives and 430 True Negatives but had a significantly higher number of False Positives (70) and False Negatives (55), indicating more misclassifications. Similarly, the FCNNs model struggled, with 440 True Positives and 420 True Negatives, while registering the highest False Positive (80) and False Negative (60) rates among the models.In contrast, the CNN-SVM proposed model demonstrated superior performance. It achieved the highest count of correctly classified tumor cases (475 True Positives) and non-tumor cases (490 True Negatives), while minimizing misclassifications with just 10 False Positives and 25 False Negatives. This indicates that the CNN-SVM model is not only more accurate but also more reliable in minimizing both types of errors, making it the most effective model for brain tumor detection among the compared approaches.

9. OUTPUT SCREENS

The User Interface (UI) of the brain tumor detection system is designed to be intuitive, user-friendly, and visually appealing. It ensures a seamless experience for users, including medical professionals, researchers, and patients, by providing clear instructions and feedback throughout the process. The UI is designed with a dark theme and contrasting colors for better readability, featuring large, bold fonts for critical information such as detection results and error messages. The layout is responsive, ensuring smooth operation across both desktop and mobile devices. Additionally, interactive elements such as hover effects, real-time validation, and a loading animation during MRI analysis enhance user engagement. The system provides a simple, efficient, and accessible experience, allowing users to quickly upload an MRI scan and obtain reliable tumor detection results. Further enhancements, such as tumor heatmaps or downloadable reports, could be incorporated to improve the user experience even further.

FIG 9.1 HOME PAGE

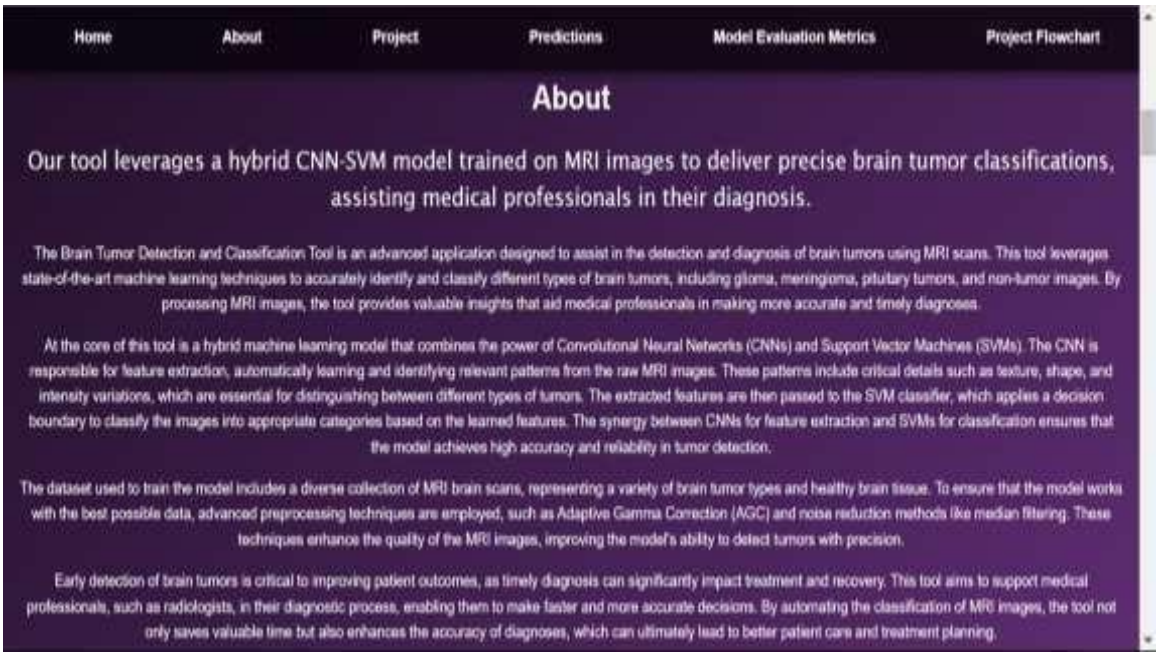


FIG 9.2 ABOUT PAGE

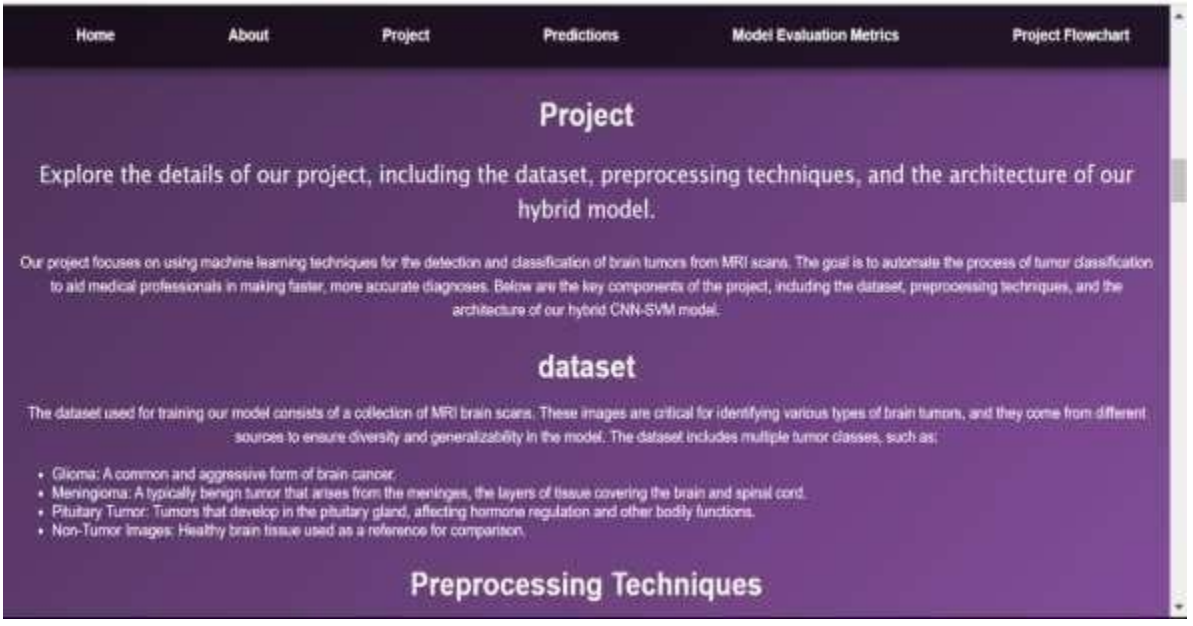


FIG 9.3 PROJECT PAGE

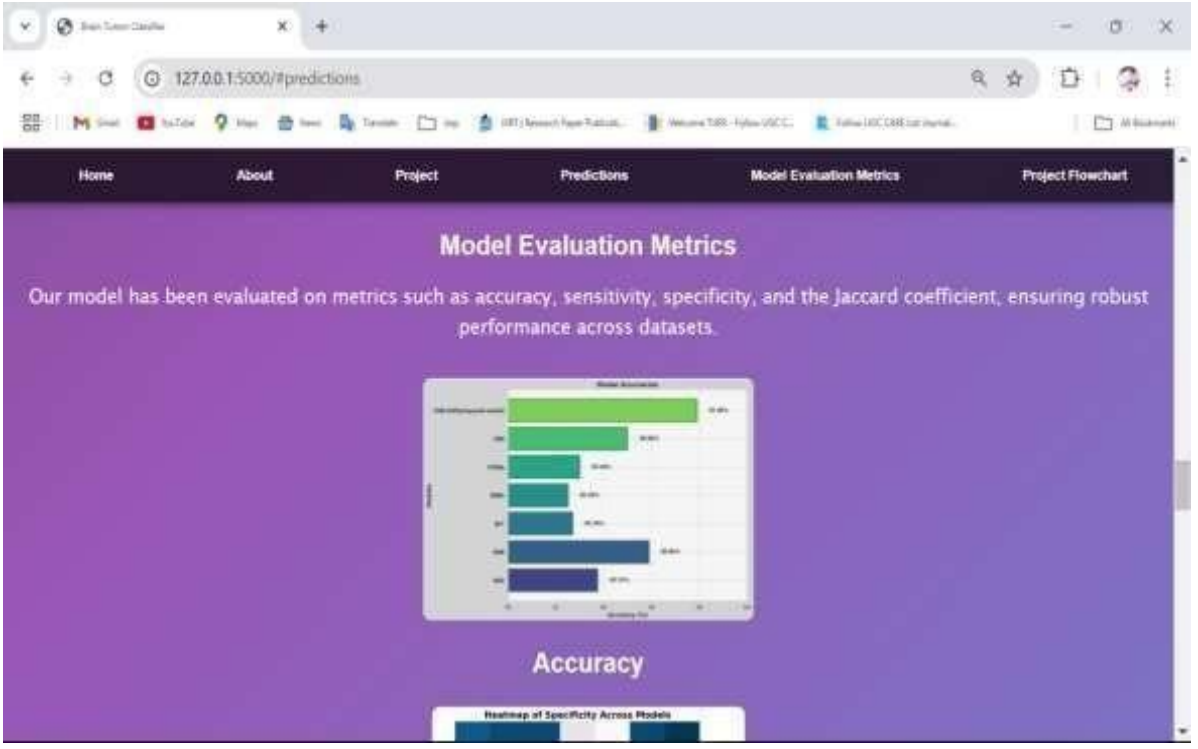


FIG 9.4 MODEL EVALUATION PAGE

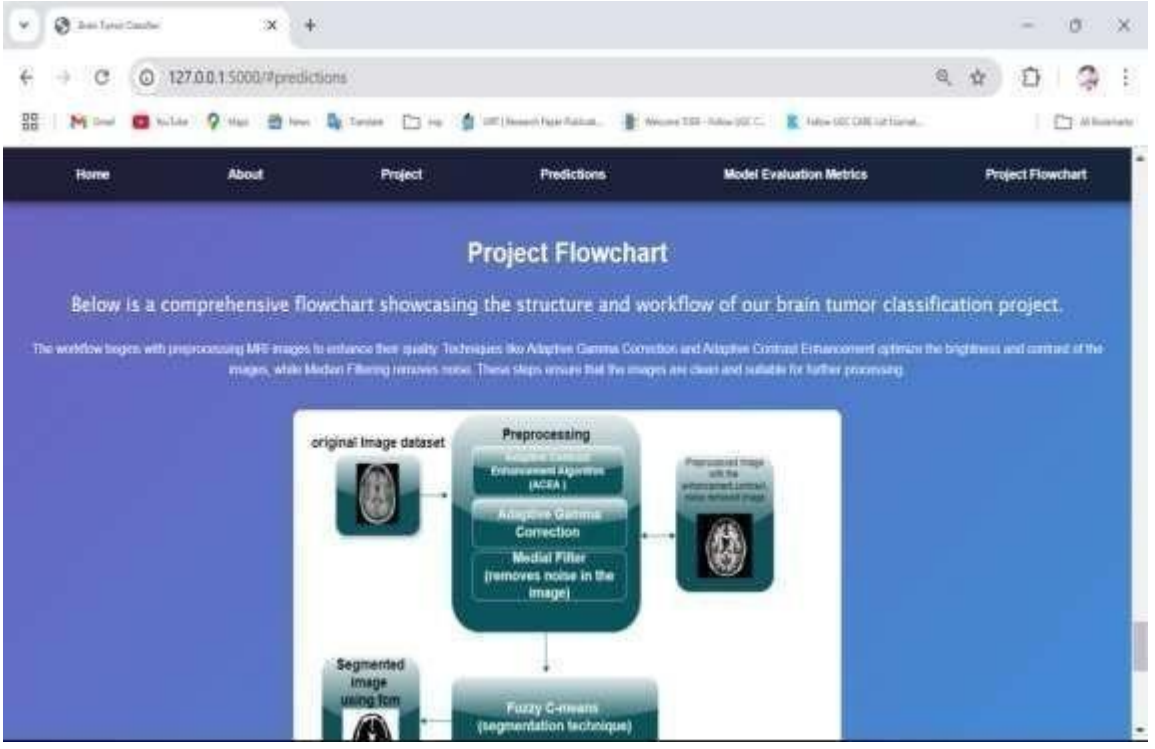


FIG 9.5 PROJECT FLOWCHART DETAIL PAGE

10. CONCLUSION

The CNN-SVM model is a promising tool for detecting brain tumors with impressive accuracy, achieving 97.94% in simulations. This hybrid model combines the strengths of Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) to automate the process of feature extraction and classification. CNNs automatically extract important features from medical images, reducing the need for manual intervention, while SVMs effectively classify the images based on those features. This combination offers a more efficient and accurate approach to detecting brain tumors compared to traditional methods, which often require extensive human input and manual feature selection.

One of the key advantages of the CNN-SVM model is its ability to simplify the detection process, making it faster and easier than traditional methods. In clinical environments where time is crucial, this model's ability to quickly analyze and detect tumors can lead to faster diagnoses, ultimately improving patient care. Additionally, the high accuracy rate demonstrated in simulations suggests that this method has great potential for real-world applications, offering better tumor detection and fewer missed diagnoses.

Looking ahead, further studies are needed to test and refine the CNN-SVM model. One key focus should be integrating the model into existing healthcare systems. This would involve making the model compatible with various medical imaging technologies and ensuring it can be used in real-time for quick decision-making. Additionally, efforts should be made to validate the model across diverse patient populations and imaging conditions to ensure its reliability in a wide range of clinical scenarios.

To make the model more accessible to healthcare professionals, researchers should also develop an intuitive user interface. Such an interface would allow clinicians to easily interpret the results, visualize tumor segments, and make informed decisions about treatment. Overall, the CNN-SVM model holds great potential in advancing brain tumor detection, and with continued research and development, it could become a vital tool in clinical practice, helping doctors make quicker and more accurate diagnoses.

11. FUTURE SCOPE

The CNN-SVM model is a promising tool for detecting brain tumors with impressive accuracy, achieving 97.94% in simulations. This hybrid model combines the strengths of Convolutional Neural Networks (CNNs)and Support Vector Machines (SVMs) to automate the process of feature extraction and classification. CNNs automatically extract important features from medical images, reducing the need for manual intervention, while SVMs effectively classify the images based on those features. This combination offers a more efficient and accurate approach to detecting brain tumors compared to traditional methods, which often require extensive human input and manual feature selection.

One of the key advantages of the CNN-SVM model is its ability to simplify the detection process, making it faster and easier than traditional methods. In clinical environments where time is crucial, this model's ability to quickly analyze and detect tumors can lead to faster diagnoses, ultimately improving patient care. Additionally, the high accuracy rate demonstrated in simulations suggests that this method has great potential for real-world applications, offering better tumor detection and fewer missed diagnoses.

While the model shows significant promise, future research should focus on integrating it into clinical software for wider use in hospitals and medical centers. To improve tumor detection further, future work could involve incorporating more advanced imaging techniques, such as color images and 3D scans, which provide more detailed and comprehensive views of the brain. These advanced imaging methods could enhance the model's ability to segment tumors more accurately, leading to better treatment planning and outcomes.

Incorporating 3D scans or colored images into the CNN-SVM model could also help with tumor segmentation, ensuring that the boundaries of tumors are more precisely defined. This would allow clinicians to plan more targeted treatments, such as surgery or radiation therapy, while minimizing damage to healthy tissue. By using richer imaging data, the model can better detect and analyze tumors, providing more reliable results for cliniciansThis would involve making the model compatible with various medical imaging technologies and ensuring it can be used in real-time for quick decision-making. Additionally, efforts should be made to validate the model across diverse patient populations and imaging conditions to ensure its reliability in a wide range of clinical scenarios.

12. REFERENCES

1. S. Solanki, U. P. Singh, S. S. Chouhan, and S. Jain, “Brain tumor detection and classification using intelligence techniques: an overview,” IEEE Access, vol. 11, pp.12870–12886, 2023.
2. S. Anantharajan, S. Gunasekaran, T. Subramanian, and R. Venkatesh, “MRI brain tumor detection using Deep Learning and Machine Learning approaches,” Measurement: Sensors, vol. 31, p. 101026, 2024.
3. N. Noreen, S. Palaniappan, A. Qayyum, I. Ahmad, M. Imran, and M. Shoaib, “A Deep Learning model based on concatenation approach for the diagnosis of brain tumor,” IEEE Access, vol. 8, pp. 55135–55144, 2020.
4. M. I. Razzak, M. Imran, and G. Xu, “Efficient brain tumor segmentation with multiscale two-pathway-group conventional neural networks,” IEEE Journal Biomedical and Health Informatics, vol. 23, no. 5, pp. 1911–1919, 2018.
5. S. Ahmad and P. K. Choudhury, “On the performance of deep transfer learning networks for brain tumor detection using MR images,” IEEE Access, vol. 10, pp. 59099–59114, 2022.
6. M. Li, L. Kuang, S. Xu, and Z. Sha, “Brain tumor detection based on multimodal information fusion and convolutional neural network,” IEEE Access, vol. 7, pp. 180134–180146, 2019.
7. H. A. Shah, F. Saeed, S. Yun, J. H. Park, A. Paul, and J. M. Kang, “A robust approach for brain tumor detection in magnetic resonance images using finetuned efficient net,” IEEE Access, vol. 10, pp. 65426–65438, 2022.
8. A. Younis, L. Qiang, Z. Afzal, M. J. Adamu, H. B. Kawuwa, F. Hussain, H. Hussain, “Abnormal Brain Tumors Classification using ResNet50 and its Comprehensive Evaluation,” IEEE Access, 2024.
9. M. Agarwal, G. Rani, A. Kumar, P. Kumar, R. Manikandan, and A. H. Gandomi, “Deep Learning for enhanced brain tumor detection and classification,” Results in Engineering, vol. 22, p. 102117, 2024.

10. Jagannadham, S. L., Nadh, K. L., Sireesha, M. (2021, November). Brain tumour detection using CNNs. In 2021 Fifth International Conference on I- SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 734-739). IEEE.
11. R. Ezhilarasi and P. Varalakshmi, "Tumor detection in the brain using faster RCNN," in 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 388–392, IEEE, August 2018.
12. A. A. Radhi, "Efficient algorithm for the detection of a brain tumor from an MRI image," International Journal of Computer Applications, vol. 975, p. 8887, 2017.
13. Toğaçar, M., Ergen, B., Cömert, Z. (2020). BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model. Medical hypotheses, 134, 109531.
14. Sharma, K., Kaur, A., Gujral, S. (2014). Brain tumor detection based on Machine Learning algorithms. International Journal of Computer Applications, 103(1).
15. Moturi, Sireesha, SN Tirumala Rao, and Srikanth Vemuru. "Optimized feature extraction and hybrid classification model for heart disease and breast cancer prediction." International Journal of Recent Technology and Engineering 7, no. 6 (2019): 1754-1772.
16. Greeshma, B., Sireesha, M., Thirumala Rao, S. N. (2022, February). Detection of arrhythmia using convolutional neural networks. In Proceedings of Second International Conference on Sustainable Expert Systems: ICSES 2021 (pp. 21-30). Singapore: Springer Nature Singapore.
17. Islam, M. K., Ali, M. S., Miah, M. S., Rahman, M. M., Alam, M. S., & Hossain, M. A. (2021). Brain tumor detection in MR image using super pixels, principal component analysis and template-based K-means clustering algorithm. Machine Learning with Applications, 5, 100044.

18. Amin, J., Sharif, M., Yasmin, M., & Fernandes, S. L. (2020). A dis-tinctive approach in brain tumor detection and classification using MRI. Pattern Recognition Letters, 139, 118-127.
19. Abiwinanda, N., Hanif, M., Hesaputra, S. T., Handayani, A., & Mengko,T. R. (2019). Brain tumor classification using convolution-al neural network. In World Congress on Medical Physics and Bio-medical Engineering 2018: June 3-8, 2018, Prague, Czech Republic (Vol. 1) (pp. 183-189). Springer Singapore.
20. Khan, H. A., Jue, W., Mushtaq, M., & Mushtaq, M. U. (2021). Brain tumor classification in MRI image using convolutional neural network. Mathematical Biosciences and Engineering.
21. Abd El Kader, I., Xu, G., Shuai, Z., Saminu, S., Javaid, I., & Salim Ahmad,
I. (2021). Differential deep convolutional neural network model for brain tumor classification. Brain Sciences, 11(3), 352
22. Panda, B., & Panda, C. S. (2019). A review on brain tumor classifi-cation methodologies. Int J Sci Res Sci Technol, 6, 346-359.
23. Khan, M. A., Ashraf, I., Alhaisoni, M., Damaševičius, R., Scherer, R., Rehman, A., & Bukhari, S. A. C. (2020). Multimodal brain tu-mor classification using deep learning and robust feature selection: A machine learning application for radiologists. Diagnostics, 10(8), 565.
24. AG, B., Srinivasan, S., P, M., Mathivanan, S. K., & Shah,
M. A. (2024). Robust brain tumor classification by fusion of deep learning and channel-wise attention mode approach. BMC Medical Imag-ing, 24(1), 147.
25. Mohanty, B. C., Subudhi, P. K., Dash, R., & Mohanty, B. (2024). Feature- enhanced deep learning technique with soft attention for MRI-based brain tumor classification. International Journal of In-formation Technology, 16(3), 1617-1626.
26. Seetha, J., & Raja, S. S. (2018). Brain tumor classification using convolutional neural networks. Biomedical & Pharmacology Jour-nal, 11(3), 1457

Enhanced Classification and Detection of Brain Tumor using Hybrid Deep Learning and Machine Learning Models

Sireesha Moturi[1], Divvela Bala Sri Abhigna[2], Gaddam Saranya[3], Shaik Sameera[4], Chinnam Harini[5], and Dodda Venkata Reddy[6]

[1,2,3,4,5,6] Department of Computer Science Engineering, Narasaraopeta Engineering College, Narasaraopet, India
*sireeshamoturi@gmail.com, abhidiveela@gmail.com, gaddamsaranya4@gmail.com, sameerashaik22498@gmail.com, chinnamharini30@gmail.com, doddavenkatareddy@gmail.com

Abstract. In recent years, deep learning has become an influential tool in medical imaging,especially the present work illustrates a new technique for brain tumor identification based on magnetic resonance images(MRI) using the combination of convolutional neural networks and support vector machine techniques. The proposed model combines CNNs performing feature extraction effectively, while SVMs classify with high precision. To begin with, some preprocessing methods like adaptive gamma correction, adaptive contrast enhancement, and median filtering are applied to MRI images so as to improve image quality and minimize noise interference. Fuzzy c-means clustering technique is used to extract important texture features like energy, mean values or entropy from GLCMs represented as matrices of grey levels co-occurrence pairs. CNN helps with acquiring deeper features from segmented images which serve as inputs during training phases of SVM classifiers based on those features thus obtained from them (CNN), eventually yielding high performance levels. An hybrid model using CNN-SVM showed an accuracy rate of 97 .94%, sensitivity 95% , specificity 98.1% when it came to distinguishing between normal and abnormal brain tissue. Therefore, this hybrid model demonstrates its capability in tumor detection due to combination of both CNN and SVM hence providing an alternative automated approach for early diagnosis.

Keywords: Magnetic Resonance(MRI) Images,Fuzzy c-means Segmentation,CNN-SVM Hybrid Model,GLCM Feature Extraction

1 Introduction

Among the most terrible types of cancer, brain tumors necessitate immediate detection for successful treatment. MRI is crucial in spotting brain tumors because of its best quality and precision. However, manual interpretation by radiologists can be time-consuming and inconsistent, necessitating automated systems to detect and classify malignant brain tumors with high accuracy. This improves

diagnosis and treatment outcomes. In India, the incidence of brain tumors is on the rise, with around 40,000 new cases projected annually. Early and precise detection is essential to improve survival rates, as delays or errors can lead to misdiagnosis and further suffering. Accurate classification of tumors, such as glioma,meningioma, and pituitary tumors [1], ensures patients receive the in evaluation of healthcare images,particularly advanced Convolutional Neural Networks [2] and Support Vector Machines are used. SVMs are known for their classification accuracy as well as their ability to generalize while CNNs excel in image classification by automatically learning hierarchical features.This research paper suggests a hybrid model integrating CNN with SVM for tumor recognition in brain [3]. There are several advanced preprocessing methods such as Adaptive Contrast Enhancement Algorithm (ACEA), adaptive gamma correction, normalization, equalization, median filtering , and data augmentation which improve performance. Fuzzy C-Means (FCM) segmentation and GLCM technique are used to precisely select tumor areas and extract features. This hybrid model neural networks aims at achieving higher accuracy and reliability in detecting and classifying brain tumors as shown in Fig 1, so as to provide more effective and personalized treatments approaches.

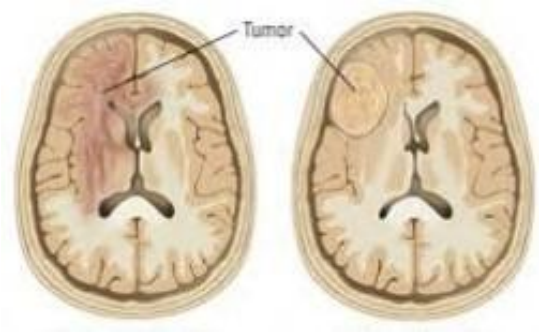


Fig. 1. Classification of healthy brain tissues and tumor brain tissue

The major contributions of this study consist of:

1. Incorporation of cutting-edge preprocessing algorithms such as Adaptive Contrast Enhancement Algorithm (ACEA), non-linear gamma enumeration tool, scaling, histogram modification, all within a single scan, to enhance image resolution and facilitate the extraction of features from MRI images.
2. Utilization of Fuzzy C-Means (FCM) segmentation for more accurate tumor region extraction, along with Grey Level Co-Occurrence Matrix (GLCM) for reliable feature extraction.
3. Development of a hybrid CNN-SVM model for better classification and differentiation of brain tumors using MRI scans.
4. Comprehensive analysis of the effectiveness of our proposed model based on a dataset of MRI images, with comparisons to conventional methods.

5. Examination of different evaluation metrics to assess the performance and reliability of the suggested approach.

This entails evaluating several performance metrics to ascertain the validity and stability of the said model. Particularly, the study will explain the model's reliability and its capability of brain tumor detection through the exploration of other metrics like accuracy, sensitivity, specificity, and precision. The goal of the suggested model with hybrids systems incorporated is the improvement of the detection and classification of brain tumors coupled with machine learning and deep learning techniques alongside a lot of pre processing techniques. This would provide a lot of improvements in the dissociative diagnosis enabling health care workers to make appropriate decisions in a short period, thereby reducing the chances of patient mistrust. To wrap up, this CNN-SVM model offers a great opportunity to be successfully employed for the goal of treating brain tumors by optimizing the existing treatment resources.

2 Related Works

Solanki S. addresses techniques for identifying and classifying brain tumors using CNNs and transfer learning models including RESNET-100 and VGGNET.Traditional methodologies like SVMs and RFs present difficulties in automation while CNN's enhance accuracy of classification by a margin of 10-15%. Furthermore, propelling up its performance is transfer learning which adds an additional 5-10% .M. Imran [4]enhanced CNN-based detection with a DCNN and a three-stage preprocessing technique, lead for enhanced output evaluated to models like VGG16 and VGG19.According to Ahmad S. and Choudhury P.K. [5], they have conducted a systematic examination of the progress made in machine learning and deep learning with respect to identifying tumors based on MRI, whereby it was shown that models which had undergone training before, like Inception-v3 and DenseNet201, can attain above 90% level of accuracy although there still remain some optimization challenges that must be addressed. To solve problems in single-modal techniques, Shah H. A [6]presented a CNN aimed at multi-modal MRI fusion. As noted in [8], EfficientNet-B0 is also considered to be one of the best models on tumor detection; therefore, it is recommended to use transfer learning and data augmentation methods to deal with limited datasets.M. Agrawal proposed enhanced brain tumor detection and classification using deep learning models used inception V3 model for better results [9]. Sireesha,M. used cnn for better tumor classification [10] used magnetic resonance images with this efficient techniques brain tumor detection and classification is done efficiently to secure more accuracy.The article discusses an approach to detecting brain tumors using the Faster R-CNN [11] model which efficiently performs tumor identification in MRI images by region proposal networks enabling real time detection. This technique is shown to be highly precise making it possible to reduce the time taken in diagnosis in the imaging process.

2.1 Problem Statement:

Brain tumors quite serious, could cause paralysis and long term treatments. Their implications vary based on their size, location and type. Early diagnosis prevents complications.There is therefore a need for better instruments for quick, precise diagnosis to facilitate prompt treatment with improved out comes for patients.

3 Proposed methodology

The emphasis of this investigation is to detect brain tumors in MRI images [12] by employing deep and machine learning methods [14] as in Fig 2. The pre-processing phase is performed through the ACEA algorithm and median filter after segmented using fuzzy c-means and features are extracted through GLCM. Following this, a hybrid CNN-SVM model is applied to classify these features thereby increasing the accuracy of detection.

3.1 Dataset collection:

We assessed our model employing Kaggle dataset [2]encompassing T1-sequenced improved MRI scans [7] obtained from individuals diagnosed with brain tumors, as shown in Fig 3. This dataset contains meningioma, glioma, pituitary, and no tumor slice images, respectively collected from https://www.kaggle.com/datasets/vivekp7039/figshare-brain-tumor-dataset-converted-to-png.

3.2 Preprocessing:

ACE - Adaptive Contrast Enhancement Algorithm (improvised algorithm) ACE algorithm improves the image contrast by means of local statistics such as mean pixel intensity and standard deviation. ACE adjusts the image's brightness depending on its different parts. Hence, while some areas may be highlighted more than others, each one remains clear and easily visible than before.

The first step is to slice the picture into segments for localized contrast modification by(1).

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{1}$$

Next, we calculate the standard deviation,the mean using equation (2):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - \mu)^2} \tag{2}$$

After gathering local averages and standard deviations (equations 1 and 2),as shown in equation (3). The modified pixel value can be expressed by this formula:

$$p_{\text{new}} = \mu + k \cdot (p - \mu) \tag{3}$$

With the use of local means and contrasts, this method improves pixel and image quality thereby giving a better appearance to the whole image.

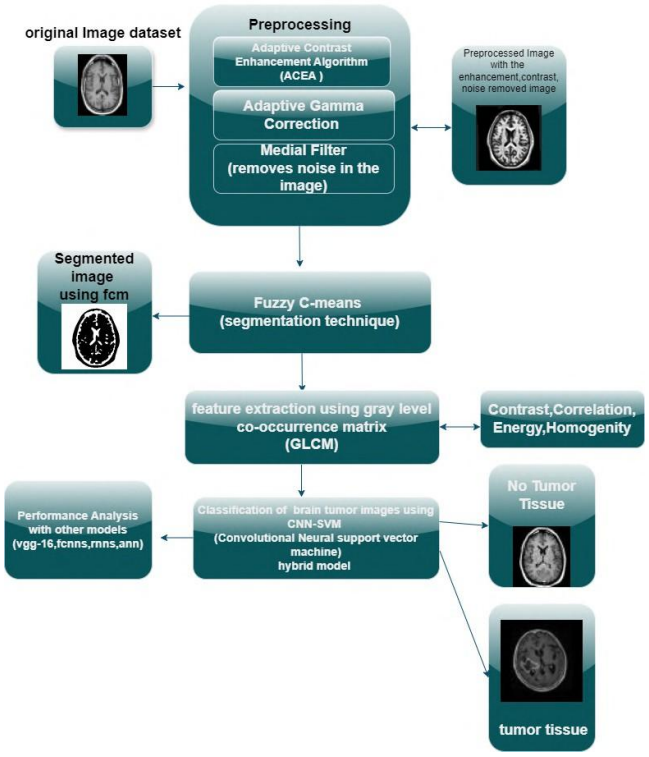


Fig. 2. Procedure Outline for the Developed Technique.

Adaptive Gamma Correction Algorithm(AGC): AGC alters brightness and contrast locally to enhance MRI images for brain tumor detection; it enables better visualization of tumors than their neighboring tissues. In contrast to conventional gamma correction, AGC adjusts the value of gamma for each zone.The formula used to determine the corrected pixel intensity $I_{\text{corrected}}$ (equation 4) is as follows:

$$I_{\text{corrected}}(x, y) = I(y)^{\gamma_{\text{local}}} \tag{4}$$

To obtain clear, high resolution images, AGC is used to increase local gamma value, $gamma_{\text{local}}$, in MRI images.

Median filter: Median filtering is a widely used technique for reducing noise. It helps clean up images while preserving important details.At each position, the central pixel's value is replaced by the median of the values within the window. The new pixel value is computed using the formula(5):

$$u_{\text{updated}}(a, b) = \text{median}\{p(a - m, b - n), \ldots, p(a + m, b + n)\} \quad (5)$$

Here,$u_{\text{updated}}(a, b)$(5)represents the new parameter of (a,b) where (a,b) is the updated parameter of the pixel at position (a,b). This median takes account of the pixel values gathered from a window around (a, b) such that k represents half the window size.

3.3 Enhanced Image Segmentation Through Fuzzy C-Means Clustering:

The Fuzzy C-Means (FCM) segmentation is an often-used unsupervised clustering technique Certainly, brain tumors can be detected and segmented using fuzzy c-means method (FCM). This is an improvement over conventional methods because FCM takes into account the inbuilt fuzziness of data unlike traditional techniques that apply hard classifications.
In the process of Cluster Centre Computation, new cluster centers are computed through the formula (6):

$$\mu_j = \frac{\sum_{i=1}^{N} v_{ij}^{m} x_i}{\sum_{i=1}^{N} v_{ij}^{m}} \quad (6)$$

Then, update the membership values based primarily on the distance between data points and cluster centers (6) using the formula:

$$p_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|a_i - x_j\|}{\|a_i - x_k\|} \right)^{\frac{2}{v-1}}} \quad (7)$$

The objective function can be calculated to check for convergence using formula (8):

$$j_m = \sum_{i=1}^{N} \sum_{j=1}^{C} v_{ij}^{m} \cdot \|x_i - \mu_j\|^2 \quad (8)$$

This process ends when it reaches a predetermined level of the objective function (8) or when it has carried out its maximum number of iterations. Otherwise, the processes are repeated.

3.4 Feature Extraction using GLCM :

A gray level co-occurrence matrix known as GLCM is a common textual study method that examines spatial correlations of pixel intensities in MRI images. This statistical technique captures texture information, helping to differentiate tissues and detect pathological features like tumors in medical MRI scans. The

Following several statistical features can be obtained from GLCM to characterize the MRI image texture. In this approach was suggested, contrast, dissimilarity, homogeneity, energy, correlation.

1. Contrast: It measures the intensity difference between neighboring pixels. A

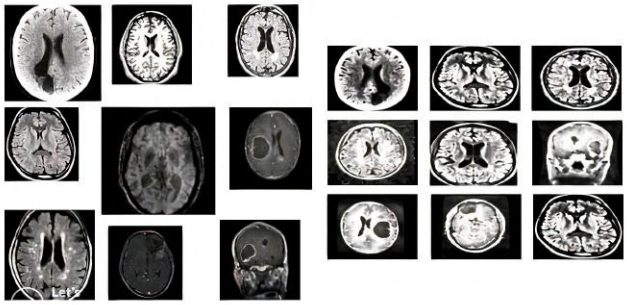


Fig. 3. Comparisons of Different and Same Sizes Sample MRI Images

higher contrast value from (9) which may be useful in differentiating tumor areas from normal tissue.

$$\text{contrast} = \sum_{p,q} (p - q)^2 a(p, q) \tag{9}$$

2. Correlation:In an image, correlation measures of how much image data points(pixels) are related linearly to one another. This indicative bare skeleton measure (10) indicates how intensity correlates with the pixel map of different positions in any given place.

$$\text{correlation} = \frac{\sum_{l,m} (l - \mu_x)(j - \mu_y) P(l, m)}{\sigma_x \sigma_y} \tag{10}$$

3. Energy: It represents (11) the combination of the class features in the GLCM, which means that the text is identical.

$$\text{energy} = \sum_{i,j} R(i, j)^2 \tag{11}$$

4. Homogeneity:In (12) measures how close the distribution of features in the GLCM is to the diagonal.

$$\text{homogeneity} = \sum_{i,j} \frac{Q(i, j)}{1 + (i - j)^2} \tag{12}$$

3.5 Integrated Hybrid Models WithCNN and SVM for Enhanced Classification:

This model Combines the feature extraction of CNN neural network with the classification of SVM; hence optimizing image analysis and category boundaries.

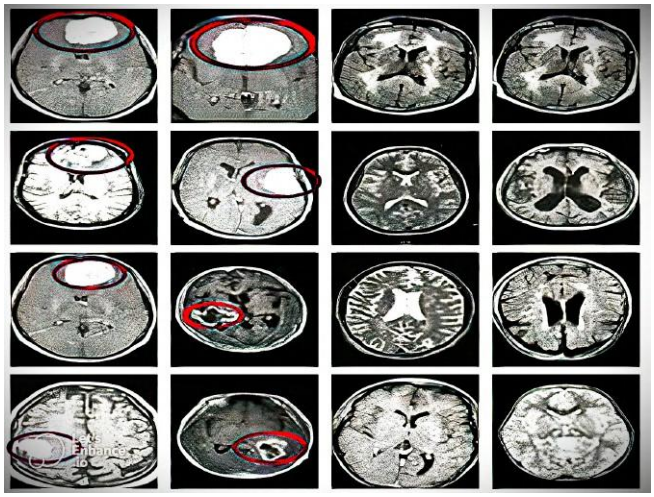


Fig. 4. Hybrid CNN-SVM Model for Classification Approach

Designing Approach for Hybrid CNN - SVM system: The CNN-SVM model in it, various raw images are passed through a CNN [13] where rectified linear unit (ReLU) activations and pooling layers help to maintain important information while reducing dimensions. After that, a fully connected layer(13) compresses the flattened output which is then classified by the SVM machine learning model that searches for the best hyperplane for class separation.

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \beta_i z_i \Phi(x_i, x) + c\right) \tag{13}$$

Implementation of CNN-SVM Hybrid Model Architecture: In order to effectively create a CNN-SVM structure , the initial step is teaching a CNN using a loss function, for instance cross entropy loss, in order to achieve the extraction of useful features.

This training consists of image segmentation which improves the feature extraction and the entire system performance(14).

$$\text{LCNN} = -\sum i = 1^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{14}$$

Model Output: The outcome model of CNN-SVM combine CNN to extract features and SVM for the ultimate classification which enhances prediction accuracy, as in Fig 5.

Developing the CNN-SVM Hybrid Model A Training Methodology: The training mode for CNN-SVM hybrid model uses dual optimization strategy making use of the two. The CNN [11] is initially trained to extract feature vectors from input MRI images. These are fed into SVM [15] for classifying purposes.This system performs alternating updates of both CNN and SVM based on metrics a. results than when only CNN or SVM were used separately. The developed model identifies and detect the tumor as shown in Fig 4 efficiently.

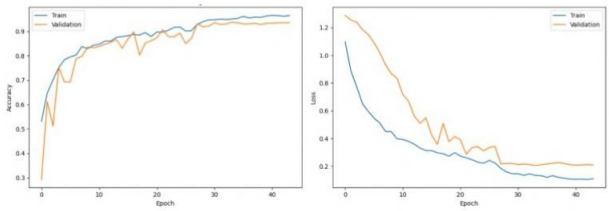


Fig. 5. CNN-SVM model accuracy and loss

4 Results and Discussions:

Using Python 3.10 based deep learning or Machine Learning methods,this section is devoted in brain tumor detection from MRI images. The evaluation of models has been done through TP, TN, FP and FN metrics that compare CNN-SVM model with other models such as CNN, VGG, RNN, RFC, FCNN and ANN. Accuracy, Jaccard index as well as sensitivity are used to measure performance.

4.1 Accuracy

The performance of a model in accurately distinguishing between tumor types and normal tissue is referred to as the accuracy(15).It is a key metric for measuring performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

Comparing the suggested method shows clearly higher accuracy at 97.94% as shown in Fig 6.

4.2 Specifivity:

Specificity(16) measures the proportion of true negatives, indicating how well a test identifies the absence of disease. As shown in Fig 7, CNN-SVM achieved the highest specificity of 98.1 %, excelling in MRI classification.

$$\text{Specificity(TNR)} = \frac{TN}{TN + FP} \tag{16}$$

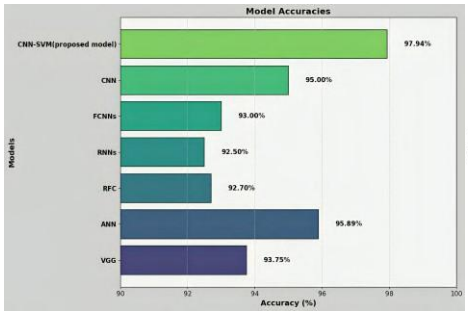


Fig. 6. Accuracy comparison.

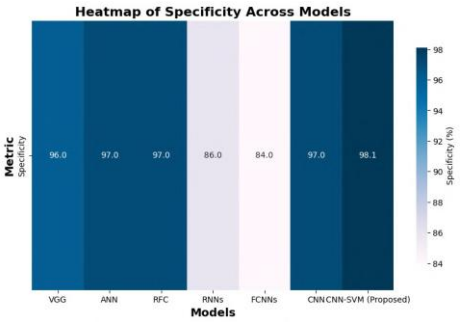


Fig. 7. Specificity comparison.

4.3 Sensitivity:

True positive probability is what sensitivity assesses. In the design of the CNN-SVM model, the sensitivity value(17) was obtained as minutely as 95% which exceeds that of traditional approaches.

$$\text{sensitivity} = \frac{TP}{TP + FN} \tag{17}$$

4.4 Jaccard Coefficient

Jaccard coefficient, commonly referred to as the Jaccard index or intersection over union (IoU) (18), is a primary measure in image segmentation and analysis. Here,Fig 9 shows Jaccard index of the analysis of the different models.

$$\text{jaccardcoefficient} = \frac{|p \cap q|}{|p \cup q|} \tag{18}$$

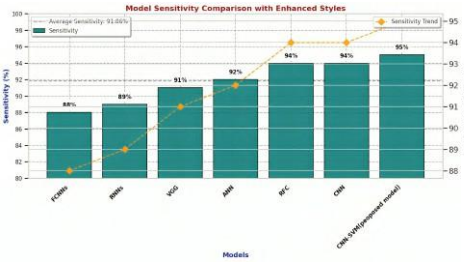


Fig. 8. Sensitivity Comparison.

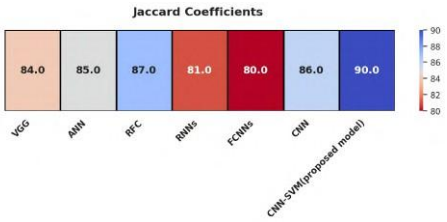


Fig. 9. Jaccard Coefficient.

5 Discussion:

In Table 1, the performance measures of several machine learning techniques are compared. These techniques include CNN, ANN,RFC,RNN,FCNN among others and the proposed model CNN-SVM plus the VGG architecture. Though each model has its level of accuracy, speed and overall performance, the costs in terms of computational power also differ across the models. For example, high-performance ANN models require sophisticated hardware where they are deployed and are dependent on tuning of hyperqualities which often results to differences in their performances. Random Forest Classifiers have a reputation for being versatile and capable of dealing with diverse data, the ensemble methods tend to be expensive in terms of computing power as the number of trees increases which in most cases leads to delays especially with bigger data sets. RNNs are great for tables and graphs that have all the data points in a particular order, but they cannot handle farther back gaps so well and tend to be heavier on the processing.

Table 1. Performance comparison of different models.

Model	Accuracy	Jaccard Coefficient	Sensitivity	Specificity
VGG	93.75	84	91	96
ANN	95.89	85	92	97
RFC	92.70	87	94	97
RNNs	92.5	81	89	86
FCNNs	93	80	88	84
CNN	95	86	94	97
CNN-SVM(propsed)	97.94	90	95	98.1

The hybrid model that combines CNN [16] and SVM proves to be a suitable contender that is less time-consuming and more efficient in terms of performance by combining the feature extraction advantage of CNNs and the classification

prowess of Support Vector Machines (SVM). An in-depth presentation of the performance metrics, such as accuracy, specificity, sensitivity, and Jaccard's index for each model across experiences with various datasets is demonstrated in the Fig 6-9. Such figures indicate the performance capabilities of each model in various scenarios.Visual assessments of the models under review further confirm the superiority of the combined CNN-SVM model in terms of performance metrics with respect to the discussed other models.

6 Conclusion:

Developed CNN-SVM model is very effective at detecting brain tumors with an accuracy of 97.94% Through advanced types of deep neural networks enabling easier approach to automatic feature extraction by using a supporting vector machine classifiers combination, it becomes quite simpler than traditional methods thereby offering faster and easier way out. This model was found by simulations to have potential for better accuracies, and future researchers should focus on incorporating it into clinical software by using color images or 3D scans in order to achieve better tumor segmentation. Simulations have shown how viable this method is thus promising improved accuracy levels in regard to brain tumor detection using CNN-SVM models. Future research here is most important however; incorporating these algorithms into clinical software would enhance their practical uses in hospitals while working with colored photographs or three-dimensional scans could also assist better tumor segmentation leading to a more comprehensive instrument for diagnosing and planning treatment of brain tumors. As a whole, the CNN-SVM model is a promising step forward in the detection of brain tumors; it includes hybrid approaches that utilize deep learning techniques as well as machine learning ones for effective and accurate results. To enhance its usefulness in medical imaging, subsequent studies must seek to diversify its usage and implement it in health care systems

References

1. S. Solanki, U. P. Singh, S. S. Chouhan, and S. Jain, "Brain tumor detection and classification using intelligence techniques: an overview," *IEEE Access*, vol. 11, pp. 12870–12886, 2023.
2. S. Anantharajan, S. Gunasekaran, T. Subramanian, and R. Venkatesh, "MRI brain tumor detection using deep learning and machine learning approaches," *Measurement: Sensors*, vol. 31, p. 101026, 2024.
3. N. Noreen, S. Palaniappan, A. Qayyum, I. Ahmad, M. Imran, and M. Shoaib, "A deep learning model based on concatenation approach for the diagnosis of brain tumor," *IEEE Access*, vol. 8, pp. 55135–55144, 2020.
4. M. I. Razzak, M. Imran, and G. Xu, "Efficient brain tumor segmentation with multiscale two-pathway-group conventional neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 5, pp. 1911–1919, 2018.
5. S. Ahmad and P. K. Choudhury, "On the performance of deep transfer learning networks for brain tumor detection using MR images," *IEEE Access*, vol. 10, pp. 59099–59114, 2022.

6. M. Li, L. Kuang, S. Xu, and Z. Sha, “Brain tumor detection based on multimodal information fusion and convolutional neural network,” *IEEE Access*, vol. 7, pp. 180134–180146, 2019.
7. H. A. Shah, F. Saeed, S. Yun, J. H. Park, A. Paul, and J. M. Kang, “A robust approach for brain tumor detection in magnetic resonance images using finetuned efficient net,” *IEEE Access*, vol. 10, pp. 65426–65438, 2022.
8. A. Younis, L. Qiang, Z. Afzal, M. J. Adamu, H. B. Kawuwa, F. Hussain, and H. Hussain, “Abnormal Brain Tumors Classification using ResNet50 and its Comprehensive Evaluation,” *IEEE Access*, 2024.
9. M. Agarwal, G. Rani, A. Kumar, P. Kumar, R. Manikandan, and A. H. Gandomi, “Deep learning for enhanced brain tumor detection and classification,” *Results in Engineering*, vol. 22, p. 102117, 2024.
10. Jagannadham, S. L., Nadh, K. L., Sireesha, M. (2021, November). Brain tumour detection using cnn. In 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 734-739). IEEE.
11. R. Ezhilarasi and P. Varalakshmi, “Tumor detection in the brain using faster R-CNN,” in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 388–392, IEEE, August 2018.
12. A. A. Radhi, “Efficient algorithm for the detection of a brain tumor from an MRI image,” *International Journal of Computer Applications*, vol. 975, p. 8887, 2017.
13. Toğaçar, M., Ergen, B., Cömert, Z. (2020). BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model. Medical hypotheses, 134, 109531.
14. Sharma, K., Kaur, A., Gujral, S. (2014). Brain tumor detection based on machine learning algorithms. International Journal of Computer Applications, 103(1).
15. Moturi, Sireesha, SN Tirumala Rao, and Srikanth Vemuru. "Optimized feature extraction and hybrid classification model for heart disease and breast cancer prediction." International Journal of Recent Technology and Engineering 7, no. 6 (2019): 1754-1772.
16. Greeshma, B., Sireesha, M., Thirumala Rao, S. N. (2022, February). Detection of arrhythmia using convolutional neural networks. In Proceedings of Second International Conference on Sustainable Expert Systems: ICSES 2021 (pp. 21-30). Singapore: Springer Nature Singapore.

Abhigna_brain

ORIGINALITY REPORT

9%	4%	8%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2021 Publication	1%
2	1library.net Internet Source	1%
3	Karrar Neamah, Farhan Mohamed, Myasar Mundher Adnan, Tanzila Saba, Saeed Ali Bahaj, Karrar Abdulameer Kadhim, Amjad Rehman Khan. "Brain Tumor Classification and Detection Based DL Models: A Systematic Review", IEEE Access, 2023 Publication	1%
4	dokumen.pub Internet Source	1%
5	Thompson Stephan. "Artificial Intelligence in Medicine", CRC Press, 2024 Publication	<1%
6	Submitted to RMIT University Student Paper	<1%

7	Submitted to Sikkim Manipal University Student Paper	< 1 %
8	Submitted to University of Wales, Bangor Student Paper	< 1 %
9	www.freepatentsonline.com Internet Source	< 1 %
10	Majid Nour, Zafer Cömert, Kemal Polat. "A Novel Medical Diagnosis model for COVID-19 infection detection based on Deep Features and Bayesian Optimization", Applied Soft Computing, 2020 Publication	< 1 %
11	Niu, X.X.. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits", Pattern Recognition, 201204 Publication	< 1 %
12	arxiv.org Internet Source	< 1 %
13	mdpi-res.com Internet Source	< 1 %
14	"Artificial Intelligence Applications and Innovations", Springer Science and Business Media LLC, 2020 Publication	< 1 %
15	"Soft Computing for Security Applications", Springer Science and Business Media LLC,	< 1 %

2023
Publication

16	A. Priya, V. Vasudevan. "Brain tumor classification and detection via hybrid alexnet-gru based on deep learning", Biomedical Signal Processing and Control, 2024 Publication	<1%
17	Altyeb Altaher Taha, Sharaf Jameel Malebary. "An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine", IEEE Access, 2020 Publication	<1%
18	Aruna Chen, Da Lin, Qiqi Gao. "Enhancing brain tumor detection in MRI images using YOLO-NeuroBoost model", Frontiers in Neurology, 2024 Publication	<1%
19	Pranjal Agrawal, Nitish Katal, Nishtha Hooda. "Segmentation and classification of brain tumor using 3D-UNet deep neural networks", International Journal of Cognitive Computing in Engineering, 2022 Publication	<1%
20	Vivek Kumar, Pinki Chugh, Bhuprabha Bharti, Anchit Bijalwan, Amrendra Tripathi, Ram Narayan, Kapil Joshi. "MRI-Based Brain Tumor Detection Using Machine Learning", Wiley, 2024	<1%

	Publication	
21	assets.researchsquare.com Internet Source	<1%
22	ouci.dntb.gov.ua Internet Source	<1%
23	www.mdpi.com Internet Source	<1%
24	E Pushparaj, Arun A. "A Study on Brain Tumor in Various Fields using Machine Learning", 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), 2023 Publication	<1%
25	Lalit Mohan Goyal, Tanzila Saba, Amjad Rehman, Souad Larabi-Marie-Sainte. "Artificial Intelligence and Internet of Things - Applications in Smart Healthcare", CRC Press, 2021 Publication	<1%
26	Maria Nazir, Sadia Shakil, Khurram Khurshid. "Role of Deep Learning in Brain Tumor Detection and Classification (2015 to 2020): A Review", Computerized Medical Imaging and Graphics, 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

CERTIFICATE

CERTIFICATE
OF PRESENTATION

AMRIT 2024

Paper ID: 127

Paper Title: Enhanced Classification and Detection of Brain Tumor using Hybrid Deep Learning and Machine Learning Models

Authors: Sireesha Moturi, Divvela Bala Sri Abhigna, Shaik Sameera, Chinnam Harini.

*This is to certify that **SIREESHA MOTURI** of **Narasaraopeta Engineering College** has presented the paper at the **Second International Conference on Advanced Computing, Machine Learning, Robotics and Internet Technologies (AMRIT-2024)**, organised in hybrid mode by the Department of Computer Science & Engineering and Department of Electrical Engineering, National Institute of Technology Agartala, Tripura, jointly organised by the Department of Computer Science, Assam University, Silchar, Assam, during November 8-9, 2024.*

Dr. Prodipto Das
Program Chair, AMRIT-2024
Dept. of CS, AU, Silchar

Dr. Sadhan Gope
Organizing Chair, AMRIT-2024
Dept. of EE, NIT Agartala

Dr. Ranjita Das
Organizing Chair, AMRIT-2024
Dept. of CSE, NIT Agartala