BRAIN TUMOUR DETECTION AND SEGMENTATION

A PROJECT REPORT

Submitted by

MD KAZI SAJIDUDDIN

ABHISHEK SINGH

In partial fulfilment for the award of the degree

Of

BACHELOR OF COMPUTER APPLICATIONS

MAHARAJA SURAJMAL INSTITUTE

C-4, JANAKPURI,

NEW DELHI – 110058

DECEMBER 2023

MAHARAJA SURAJMAL INSTITUTE

BONAFIDE CERTIFICATE

Certified that this project report "**Brain Tumour Detection and Segmentation**" Is the bonafide work of "**MD KAZI SAJIDUDDIN**" and "**ABHISHEK SINGH**" who carried out the project work under my supervision.

_____

SIGNATURE

Harjender Singh

(Assistant Professor)

Department of Computer Science

C-4 Market, Fire Station Rd, Janakpuri,

New Delhi 110058

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those experts who have already traded this past before and hence become master of it and as a result, our leader. So, we would like to take this opportunity to take all those individuals who have helped us in visualizing this project.

We express our deep gratitude to my project guide Mr. Harjender Singh for providing timely assistance to my query and guidance that he gave owing to his experience in this field for the past many years. He had indeed been a lighthouse for me on this journey. We would also like to take this opportunity to thank him in, for his guidance in selecting this project and also for providing us all the details on proper presentation of this project.

We extend our sincere appreciation to all our professors from the Maharaja Surajmal Institute for their valuable insight and tips during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge them individually.

Thanking You,

_____

SYNOPSIS

Brain tumours represent abnormal cell growth within the brain, posing significant health risks and potentially becoming life-threatening. The accurate identification, segmentation, and classification of these tumours play a pivotal role in the diagnosis, formulation of effective treatment plans, and monitoring the progression of the disease. Achieving precision in these processes is essential for tailoring personalized interventions and optimizing patient outcomes. Recent years have witnessed remarkable strides in the realm of medical imaging and artificial intelligence (AI) techniques, sparking a transformative revolution in the evaluation of brain tumours. Cutting-edge technologies and sophisticated algorithms have enhanced our ability to discern subtle nuances in imaging data, facilitating early and precise detection of tumours. This integration of AI into the diagnostic landscape not only expedites the decision-making process but also opens new avenues for innovative therapeutic approaches. The synergy between advanced imaging modalities and AI holds great promise in improving overall patient care and prognosis by providing clinicians with valuable insights into the characteristics and behaviour of brain tumours. As we continue to explore and harness the potential of these technologies, the landscape of brain tumour evaluation is poised for continuous evolution, offering hope for more effective and personalized treatments in the future.

Detection and classification of brain tumours represent a critical frontier in the realm of medical diagnostics, playing a crucial role in deciphering the complexities of neurological disorders and devising effective treatment strategies. Among the myriad of medical imaging techniques, Magnetic Resonance Imaging (MRI) has emerged as a linchpin, offering unparalleled insights into the intricate structure and composition of the brain. However, the conventional reliance on manual inspection by radiologists introduces inherent challenges, including time-intensive processes and the necessity for specialized expertise. This synopsis explores the transformative integration of Computer-assisted Diagnosis (CAD) and cutting-edge machine learning, particularly deep learning, to revolutionize the landscape of brain tumour detection.

The Role of MRI in Brain Tumour Detection

Magnetic Resonance Imaging (MRI) stands as an indispensable pillar in the domain of brain tumour detection, offering a non-invasive and highly detailed perspective into the intricacies of brain structures. Its ability to provide comprehensive and high-resolution images makes MRI a cornerstone in the diagnostic toolkit, allowing healthcare professionals to visualize subtle abnormalities with remarkable clarity. However, the conventional reliance solely on MRI for brain tumour detection introduces challenges, most notably the laborious and time-intensive nature of manual analysis required by radiologists.

The traditional paradigm, centred around manual analysis of MRI scans, faces inherent limitations that impact the speed and scalability of diagnosis. Recognizing this bottleneck, our project embarks on a mission to redefine the boundaries of precision in brain tumour diagnostics. In doing so, we leverage the capabilities of advanced medical imaging, pushing

beyond the confines of traditional methodologies. By incorporating state-of-the-art machine learning methodologies into the diagnostic process, our project seeks to revolutionize the way we interpret and extract meaningful insights from MRI data.

The integration of machine learning not only addresses the challenges posed by manual analysis but also holds the promise of significantly enhancing the efficiency and accuracy of brain tumour detection. Our approach aims to empower healthcare professionals with intelligent tools that can autonomously analyse MRI data, expedite the diagnostic workflow, and uncover nuanced patterns that may elude the human eye. Through this amalgamation of advanced medical imaging and cutting-edge machine learning, our project strives to propel the field of brain tumour detection into a new era, where precision, speed, and scalability converge to meet the evolving demands of modern healthcare.

Evolution from Traditional Methods to Machine Learning

The evolution from traditional machine learning methods to the sophisticated realm of deep learning signifies a paradigm shift in the approach to brain tumour detection. Traditional methodologies often required the painstaking creation of handcrafted features for effective classification, imposing limitations on adaptability and accuracy. However, the advent of deep learning methodologies has transformed this landscape by eliminating the need for manual feature extraction. Deep learning, with its ability to autonomously learn hierarchical representations, not only streamlines the detection process but also enhances accuracy, marking a pivotal moment in the contemporary approach to brain tumour detection.

Transfer Learning for Enhanced Precision

To augment the precision and efficacy of our brain tumour detection system, our project Employing transfer learning is a pivotal aspect of our project geared towards elevating the precision and effectiveness of our brain tumour detection system. Transfer learning stands out as a sophisticated strategy wherein pre-trained models, initially developed for one task, are adeptly repurposed to enhance performance in a related task—in this case, the intricate domain of brain tumour detection. In our relentless pursuit of excellence, our project strategically integrates three highly acclaimed transfer learning models: VGG16, and Inception V3.

These models were not chosen haphazardly; rather, they were meticulously selected based on their well-established track record of excellence in image classification tasks. VGG16, and Inception V3 have demonstrated prowess in learning intricate patterns and features from diverse datasets, thereby acquiring a wealth of knowledge that transcends specific domains. By leveraging these pre-existing foundations, our brain tumour detection system is endowed with a robust framework that goes beyond conventional training methods.

The incorporation of these transfer learning models equips our system with the ability to discern subtle nuances and intricate details in medical imaging data, contributing to more accurate and efficient brain tumour detection. The amalgamation of these models not only expedites the training process but also harnesses the rich information encapsulated in their pre-existing knowledge. This strategic fusion of advanced transfer learning techniques and the

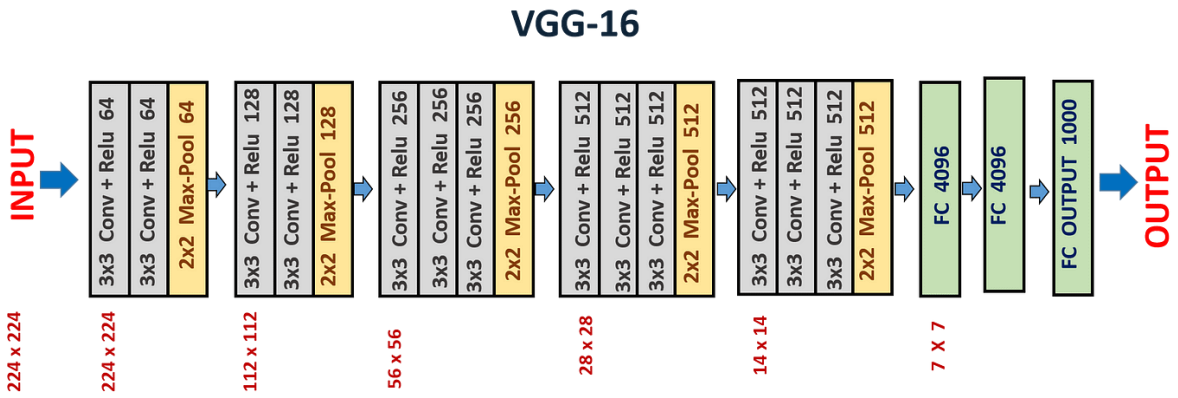
complexities of brain tumour detection propels our project to the forefront of cutting-edge advancements in medical imaging and artificial intelligence, promising enhanced precision and reliability in the diagnosis of brain tumours.

VGG16: Leveraging Depth for Understanding

VGG16 emerges as a cornerstone in our project's architecture, leveraging its profound impact on understanding intricate patterns within images through its deep architecture, which encompasses an impressive 16 layers. The significance of VGG16 lies in its ability to harness the power of depth, a key attribute that plays a crucial role in unravelling the complexities inherent in the visual representation of brain tumours. The 16-layer depth of VGG16 serves as a testament to the system's capacity to delve deep into the nuances of medical images, unveiling subtle details that may elude shallower architectures.

Within our project, VGG16's depth is strategically employed to augment the discernment of subtle features associated with brain tumours. The cascade of layers facilitates the extraction of hierarchical features, enabling the model to capture intricate details at multiple levels of abstraction. This hierarchical approach proves invaluable in constructing a more comprehensive representation of tumour characteristics, transcending the limitations of simpler architectures.

By delving into the intricate layers of VGG16, our project aims to improve the overall accuracy of our brain tumour detection system. The depth of VGG16 enables the model to grasp complex relationships within the image data, allowing for a more nuanced understanding of the varied manifestations of brain tumours. This not only enhances the system's ability to identify tumours but also contributes to a more holistic and detailed characterization of their diverse attributes. In essence, VGG16's depth serves as a catalyst for achieving a deeper and more nuanced comprehension of the intricacies embedded in medical imaging, fostering a more effective and precise approach to brain tumour detection within our project.

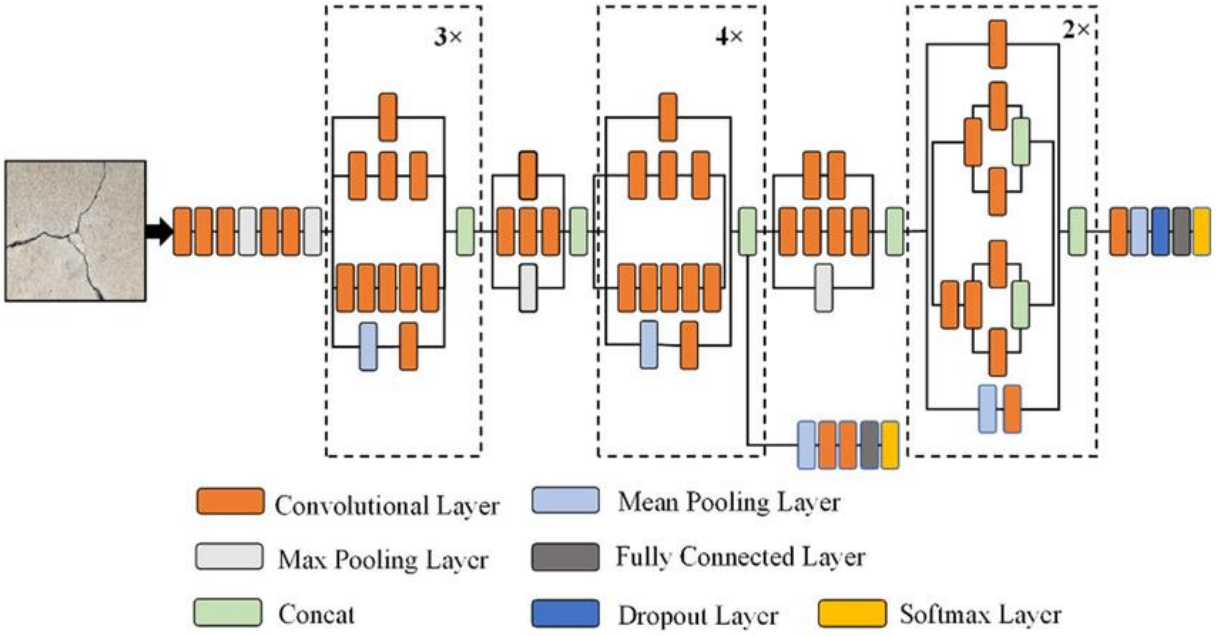


Inception V3: Balancing Computation and Accuracy

Incorporating Inception V3 into our ensemble of transfer learning models introduces a strategic dimension to our project by harmonizing computational efficiency with precision. At the heart of Inception V3's effectiveness lies its distinctive inception modules, which artfully navigate the delicate balance between computational demands and model accuracy. This nuanced architecture enables the model to process information in parallel, thereby elevating its capability to unravel complex patterns within the intricate landscape of medical imaging data.

The parallel processing capability of Inception V3 is a game-changer in our pursuit of enhancing brain tumour detection. By concurrently considering multiple pathways, the model excels at capturing diverse features at different scales, a feature particularly advantageous in the context of the multifaceted nature of brain tumour images. This ability to adapt to varying scales of information aligns seamlessly with the intricacies presented by different types and stages of brain tumours, making Inception V3 a valuable and complementary asset within our transfer learning framework.

Moreover, Inception V3's emphasis on balancing computational efficiency with accuracy is pivotal for the practical deployment of our brain tumour detection system. The streamlined processing of information allows for efficient model inference without compromising on the intricate details crucial for accurate tumour detection. In essence, Inception V3's unique architecture contributes to the synergy of our ensemble, enriching the overall capabilities of our system and affirming its potential to advance the precision and efficiency of brain tumour detection in the realm of medical imaging and artificial intelligence.



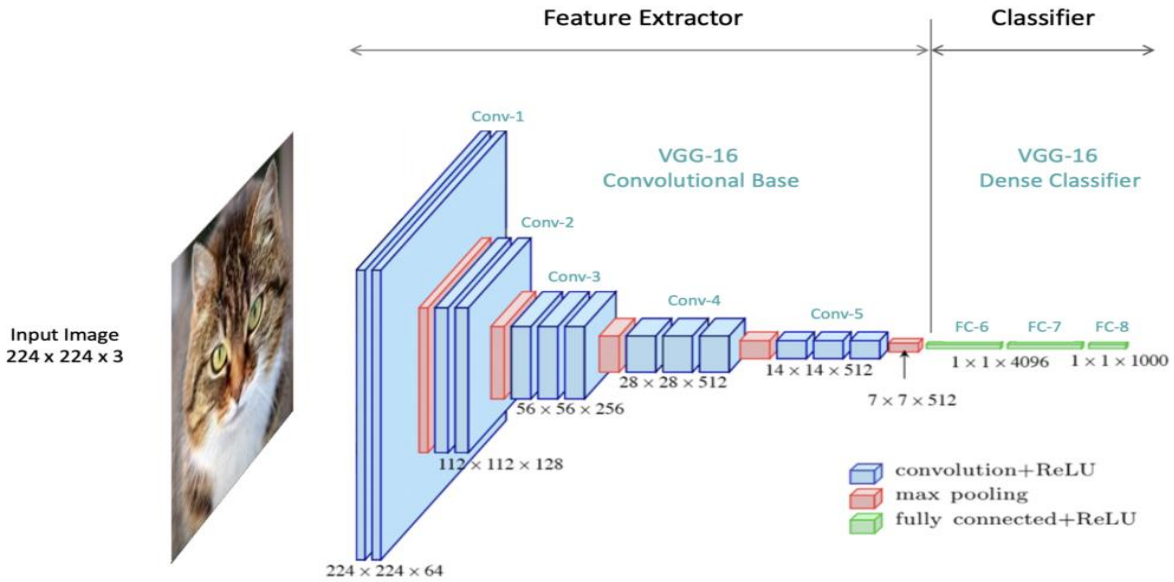
Convolutional Neural Network (CNN): Unveiling Insights Through Localized Feature Extraction

In our comprehensive approach to brain tumour detection, a fundamental component of our model architecture is the Convolutional Neural Network (CNN). Unlike the transfer learning models such as VGG16, and Inception V3 that leverage pre-existing knowledge, a CNN is designed to autonomously learn hierarchical representations directly from the raw input data.

Understanding Localized Feature Extraction: At the core of CNN's effectiveness is its capacity for localized feature extraction. The convolutional layers within the network systematically scan the input images using filters, each responsible for detecting specific patterns or features. This localized approach allows the CNN to focus on small, relevant portions of the image at a time, capturing intricate details that contribute to the overall understanding of the data.

Hierarchical Feature Learning: As the data traverses through the layers of the CNN, hierarchical feature learning unfolds. The initial layers identify simple features like edges and textures, while deeper layers progressively discern more complex and abstract representations. This hierarchical learning mimics the human visual system, where low-level features combine to form higher-level concepts, enabling the CNN to discern patterns relevant to brain tumour detection.

Customization for Brain Tumour Detection: In the context of our project, the CNN is tailored specifically for brain tumour detection. The network is trained to recognize distinctive patterns and structures associated with tumours, facilitating the identification of abnormalities within medical imaging data. The adaptability of CNNs to learn from diverse datasets empowers our model to capture the unique characteristics of different types and stages of brain tumours.



In conclusion, our project represents a groundbreaking advancement in the field of brain tumour detection, characterized by a seamless integration of cutting-edge medical imaging

techniques, innovative machine learning methodologies, and the collective strength of diverse models, including VGG16, Inception V3, and a customized Convolutional Neural Network (CNN). This holistic approach is meticulously designed to transcend the limitations of conventional methods, promising a paradigm shift in the precision and efficiency of our brain tumour detection system.

The strategic incorporation of transfer learning models—VGG16, and Inception V3—brings forth a wealth of pre-existing knowledge, honed through diverse datasets, enhancing our system's ability to discern intricate details and patterns within medical imaging data. Additionally, the utilization of a customized CNN underscores our commitment to autonomous learning, enabling the model to adapt and uncover localized features relevant to brain tumour detection.

This collaborative synergy between transfer learning and autonomous learning not only holds the potential for heightened accuracy in identifying and categorizing brain tumours but also positions our project at the forefront of technological innovation in the medical field. The outcomes of our endeavour extend beyond the immediate realm of improved patient outcomes, contributing to a deeper understanding of neurological disorders and fostering advancements in medical diagnostics and treatment planning.

As we navigate this intersection of medical imaging, machine learning, and transfer learning, our project signifies the dawn of a new era in brain tumour detection, where early and precise diagnosis becomes an attainable reality. The fusion of these technologies not only propels us towards enhanced clinical capabilities but also opens doors to transformative possibilities, paving the way for more proactive and personalized approaches in the realm of neurological healthcare.

MOTIVATION

Brain tumours represent a formidable challenge in the realm of medical diagnosis and treatment, necessitating innovative approaches to enhance accuracy, efficiency, and personalized care. Traditional methods of brain tumour classification, relying on manual analysis and subjective interpretation, encounter limitations in terms of time consumption and susceptibility to human errors. The motivation for employing advanced artificial intelligence (AI) techniques, specifically convolutional neural networks (CNNs) such as VGG16, and Inception V3, stems from the following key considerations:

1. Improved Accuracy: One of the primary motivations for using CNNs in brain tumour classification is their ability to achieve superior accuracy compared to traditional methods. CNNs excel at learning hierarchical representations of data. automatically extracting relevant features from medical images, and identifying complex patterns that may not be easily discernible to the human eye. By leveraging the power of CNNs, we can enhance the accuracy of brain tumour classification, leading to more reliable diagnoses.

2. Automated and Objective Approach: CNN-based models provide an automated and objective approach to brain tumour classification. Unlike manual methods that rely on subjective interpretation by radiologists, CNNs can analyse medical images with consistency and objectivity. This removes potential biases and variations in interpretations, leading to more standardised and reliable results. The use of Al-based models ensures a more consistent and unbiased approach to brain tumour classification.

3. Time and Cost Efficiency: Traditional methods of brain tumour classification often require substantial time and resources. Radiologists need to manually analyse and interpret a large number of medical images, which can be time- consuming and may lead to increased costs. By leveraging CNN models like VGG16 and Inception V3, we can significantly reduce the time and effort required for classification. These models can process a large volume of medical images in a relatively short time, enabling faster diagnoses and a more efficient allocation of healthcare resources.

4. Personalised Treatment Planning: Accurate classification of brain tumours using CNN models allows for personalised treatment planning. Different types of brain tumours may require specific treatment strategies, and the ability to classify tumours accurately can help medical professionals tailor treatment plans accordingly. This personalised approach improves patient outcomes and ensures that each individual receives the most appropriate and effective treatment for their specific tumour type.

5. Potential for Research and Advancements: The use of CNN models in brain tumour classification opens up avenues for further research and advancements in the field. By analysing the features and patterns learned by these models, researchers can gain valuable insights into the underlying characteristics of different tumour types. This knowledge can contribute to the development of new diagnostic techniques, treatment strategies, and improvements in overall patient care.

PROBLEM STATEMENT AND CHALLENGES

Brain tumours, intricate and potentially life-threatening medical conditions, demand a sophisticated diagnostic approach to facilitate timely and accurate treatment planning. The prevailing methods of brain tumour classification predominantly hinge on manual analysis performed by radiologists, a process fraught with subjectivity, time-consuming intricacies, and the inherent risk of human errors. These limitations underscore the pressing need for revolutionary techniques that can not only mitigate the drawbacks of traditional approaches but also significantly enhance the accuracy, efficiency, and objectivity of brain tumour classification.

In response to this imperative, our problem statement centres around the development and implementation of CNN-based models, specifically VGG16 and Inception V3, with the aim of achieving precise and automated brain tumour classification. The crux of this challenge lies in leveraging the immense potential of deep learning algorithms to discern subtle and complex patterns within medical images. The overarching objective is to empower these models to extract meaningful features that distinguish between various types and categories of brain tumours. By embarking on this problem-solving journey, we seek to transcend the constraints of conventional methods, providing healthcare professionals with a dependable, streamlined, and objective tool for brain tumour classification.

This ambitious undertaking not only aligns with the urgency of addressing the intricate nature of brain tumour diagnosis but also positions our project at the forefront of transformative advancements in medical imaging and artificial intelligence. The fusion of cutting-edge technologies and innovative methodologies within this problem statement sets the stage for a paradigm shift in the way we approach brain tumour classification, marking a significant leap towards more precise and effective healthcare interventions.

Key Challenges:

1. Complex and Diverse Tumour Patterns:

Addressing the complex and diverse patterns exhibited by brain tumours poses a formidable challenge. The intricate variations in shapes, sizes, and textures demand the development of CNN models that can robustly capture and interpret this diversity. Ensuring the models' adaptability to the myriad manifestations of brain tumours becomes paramount to achieving accurate and reliable classifications.

2. Limited Availability of Annotated Data:

The scarcity of large-scale annotated datasets for training deep learning models presents a significant hurdle. Acquiring a substantial number of labelled brain tumour images is essential for effective model training. Strategies such as data augmentation, transfer learning, and collaborative efforts become imperative to supplement the limited availability of annotated data and enhance the models' ability to generalize across diverse cases.

3. Model Generalisation:

The challenge extends to ensuring the generalization of CNN models, including VGG16 and Inception V3, to unseen data. It is crucial to train models that can accurately classify brain tumours in new and previously unseen images. Achieving model generalization is pivotal for the practical applicability of these models in real-world clinical scenarios, where encountering novel cases is routine.

4. Interpretability and Explain ability:

The inherent nature of CNN models as "black boxes" necessitates a focus on interpretability and explain ability. Understanding the decision-making process of these models in brain tumour classification is essential for gaining insights into the features and patterns that contribute to their predictions. Overcoming this challenge involves the development of techniques that provide transparent and clinically meaningful interpretations of the learned representations.

5. Imbalanced Data Distribution:

The distribution of brain tumour types in medical datasets may be uneven, leading to biased model training. Balancing the representation of different tumour categories becomes critical to prevent the model from favouring prevalent types and potentially misclassifying rarer but equally significant cases.

The goal is to address these challenges and develop CNN models, namely VGG16 and Inception V3, that can accurately classify brain tumours, leading to improved diagnostic accuracy, reduced subjectivity, and enhanced efficiency in clinical workflows. By doing so, we aim to explore healthcare professionals with a reliable and objective tool that can aid in the accurate diagnosis and treatment planning of brain tumour patients, ultimately improving patient outcomes and contributing to advancements in the field of neuro-oncology.

EXISTING SYSTEMS

The present landscape of brain tumour classification epitomizes a profound symbiosis between revolutionary convolutional neural networks (CNNs) and the dynamic realm of medical imaging. This amalgamation has sparked an extensive array of studies, each contributing to the exploration of CNNs as a formidable tool for the intricate task of classifying brain tumours. Notably, these investigations concentrate on the comprehensive analysis of medical imaging records, with a primary emphasis on the wealth of information encapsulated within magnetic resonance imaging (MRI) scans. This intersection of cutting-edge technology and healthcare not only reflects the relentless pursuit of innovation but has also given rise to a burgeoning field characterized by ongoing exploration, refinement, and the continuous quest for precision in brain tumour diagnostics.

Several key points characterize the present state of brain tumour classification using CNNs:

1. Research:

The landscape of brain tumour classification through CNNs is deeply rooted in extensive research endeavours, with numerous research papers and studies dedicated to refining and advancing computerized tumour classification. These research initiatives aim to broaden the scope of accurate and dependable models, providing invaluable support to clinical professionals in the intricate task of diagnosing brain tumours. Through a synthesis of machine learning and medical expertise, these studies contribute to the ongoing evolution of diagnostic methodologies.

2. CNN Architectures:

The exploration of CNN architectures is a cornerstone in the pursuit of robust brain tumour classification models. Notable architectures such as VGG16, Inception V3, and have been meticulously studied for their effectiveness in capturing relevant features from brain tumour images. These architectures, renowned for their depth and complexity, have demonstrated unparalleled capabilities in achieving high classification accuracy, showcasing the potential for sophisticated neural networks to discern intricate patterns within medical imaging data.

3. Datasets:

The existing system relies on publicly available brain tumour datasets, with a notable inclusion being the BRATS (Multimodal Brain Tumour Segmentation) dataset. Comprising diverse MRI scans capturing various tumour types, these datasets serve as the foundation for training and evaluating CNN models. The use of such datasets ensures a comprehensive and representative learning experience, fostering the models' ability to generalize across different tumour scenarios.

4. Preprocessing strategies:

Before subjecting the CNN models to training, a series of preprocessing strategies are employed to enhance the quality of MRI images. These steps encompass crucial processes such as skull stripping, image registration, and intensity normalization. The meticulous application of preprocessing techniques ensures that the input data is optimized, providing a clean and standardized foundation for the subsequent training phases.

5. Training and evaluation:

The training of CNN models involves the optimization of parameters through backpropagation and gradient descent, utilizing labelled MRI scans where each scan is associated with a specific tumour class. The subsequent evaluation of these models is a critical step, employing metrics such as accuracy, precision, recall, and F1 score to gauge their performance. This meticulous evaluation process ensures that the models not only learn effectively but also generalize well to unseen data, a crucial aspect for real-world applicability.

6. Results:

The culmination of these efforts has yielded promising results in brain tumour classification. CNN models, trained on substantial datasets, exhibit remarkable accuracy in distinguishing between different tumour types. This accomplishment paves the way for precise diagnoses and informed treatment planning, showcasing the potential of CNNs in enhancing clinical decision-making in the realm of neuro-oncology.

In conclusion, while the existing system has achieved commendable results, the journey continues with a call for further research and advancements to overcome current challenges and enhance the practical application of CNNs in real-world medical scenarios

OBJECTIVES AND PROPOSED INNOVATION

OBJECTIVES:

The overarching goal of this research endeavour is to propel the frontiers of accurate and efficient brain tumour classification by harnessing the transformative capabilities of Convolutional Neural Networks (CNNs), particularly employing the esteemed architectures of VGG16 and Inception V3. This ambitious objective unfolds across multifaceted layers, each aimed at refining the landscape of automated brain tumour categorization grounded in rich and diverse clinical imaging data.

The primary desires of this studies are as follows:

1. Correct Tumour type:

The foremost objective is to achieve unparalleled precision in the classification of brain tumours, categorizing them into distinct classes such as benign or malignant, gliomas, meningiomas, or metastatic tumours. This ambitious goal is pursued by harnessing the formidable capabilities of cutting-edge CNN architectures, with a specific focus on VGG16 and Inception V3. The objective extends beyond mere accuracy; it aspires to create a robust system capable of minimizing misclassifications, ensuring that each diagnosed tumour type aligns with the intricacies of its pathological nature.

The significance of correctly identifying tumour types cannot be overstated, especially when considering the stark differences in treatment approaches and prognostic outcomes associated with different classes of brain tumours. Achieving high classification accuracy becomes imperative not only for informed clinical decision-making but also for tailoring personalized treatment plans according to the unique characteristics of each tumour type.

The objective is not merely to create a binary classification system but to delve into the granularity of tumour subtypes, encompassing the diverse landscape of brain pathology. VGG16's depth and Inception V3's computational efficiency are strategically harnessed to discern intricate patterns, enabling the system to navigate the spectrum from benign to malignant, gliomas to meningiomas, and beyond.

Furthermore, the research underscores the commitment to minimizing misclassifications, recognizing the potential repercussions of erroneous diagnoses in clinical settings. By refining the models to be highly sensitive to subtle variations in tumour characteristics, the objective is to enhance the reliability of the classification system, fostering confidence among healthcare professionals and contributing to more accurate prognoses and treatment plans for individuals affected by brain tumours.

2. Improved Diagnostic selection assist:

The core objective is to provide medical specialists with an advanced and automated decision support system for brain tumour diagnosis. This entails the development of robust CNN

models with the primary goal of enhancing the accuracy and efficiency of tumour type identification.

This objective represents a pivotal shift towards augmenting the decision-making capabilities of medical specialists in the intricate field of brain tumour diagnosis. By leveraging the sophisticated capabilities of CNN models, the research aspires to not only enhance accuracy in tumour type identification but also to create a seamless and automated decision support system.

In practical terms, this means equipping medical professionals with a tool that serves as a trustworthy companion in the diagnostic process. The CNN models, meticulously crafted for this purpose, are designed to process complex medical imaging data, extract meaningful features, and provide clinicians with precise information regarding the type of brain tumour in question.

The essence of this objective lies in addressing the challenges faced by medical specialists in navigating the vast and intricate landscape of brain pathology. The automated decision support system acts as a reliable guide, offering insights derived from the wealth of data encapsulated within medical images. This, in turn, empowers doctors to make well-informed and timely treatment decisions, thereby contributing to improved patient outcomes.

Moreover, the emphasis on enhancing efficiency goes beyond mere speed. It entails streamlining the diagnostic workflow, reducing the time and cognitive load on medical professionals, and fostering a more collaborative and synergistic approach between human expertise and artificial intelligence. The objective is not just to develop a tool but to integrate it seamlessly into the clinical decision-making process, ensuring its practical utility and effectiveness in real-world medical scenarios.

By achieving this objective, the research endeavours to usher in a new era where diagnostic decisions are not just accurate but are also augmented by the power of AI, providing a valuable resource for medical specialists and ultimately enhancing the quality of care for individuals facing the complexities of brain tumours.

3. Efficient Model Training and Inference:

This pivotal objective is dedicated to the optimization of both the training and inference processes of CNN models. The overarching aim is to refine and streamline these processes, encompassing the exploration of techniques to handle large-scale brain tumour datasets, enhancing computational efficiency, and mitigating training time without compromising on the high accuracy demanded by sophisticated medical applications.

The heart of this objective lies in the recognition that the practicality of implementing CNN models in real-world medical scenarios hinges on their efficiency in both training and inference. The research aims to address the challenges associated with the scale and complexity of brain tumour datasets, acknowledging the need for innovative approaches to handle voluminous and diverse medical imaging data.

Handling Large-Scale Datasets: Efficient model training begins with the adept handling of large-scale brain tumour datasets, such as the BRATS dataset. The objective is to develop

strategies that ensure the models can effectively process and learn from diverse data, encompassing the multitude of tumour variations present in clinical settings. This involves exploring data augmentation techniques, transfer learning strategies, and other methodologies to enrich the learning experience of the models.

Enhancing Computational Efficiency: Optimizing computational efficiency is a core tenet of this objective. The research endeavours to explore innovative algorithms and architectures that not only bolster the speed of model training and inference but also ensure that these processes are feasible within the computational constraints of real-world healthcare environments. This includes the exploration of parallel processing, hardware acceleration, and model quantization techniques to achieve optimal computational efficiency.

Reducing Training Time while Preserving Accuracy: The research recognizes the importance of expediting the training process without compromising on the accuracy demanded in medical applications. Strategies to reduce training time involve the judicious selection of hyperparameters, leveraging parallel processing capabilities, and adopting techniques that strike a balance between rapid convergence and model robustness. The objective is to create models that can be trained swiftly without sacrificing the high precision required for reliable brain tumour classification.

By achieving efficiency in both model training and inference, this objective contributes to the feasibility and scalability of implementing CNN models in real-world clinical settings. The aim is to not only advance the technological capabilities of these models but to ensure their practical utility, facilitating seamless integration into the dynamic and time-sensitive landscape of healthcare decision-making.

4. Evaluation of CNN Architectures:

This objective centres on the meticulous evaluation of various CNN architectures, with a particular emphasis on VGG16 and Inception V3, for brain tumour classification. The overarching goal is to conduct a comprehensive analysis, comparing the strengths and weaknesses of each architecture in terms of accuracy, computational complexity, and model interpretability.

Comparative Analysis of VGG16 and Inception V3: The primary thrust of this objective involves a detailed examination of the performance exhibited by VGG16 and Inception V3 in the realm of brain tumour classification. Each architecture brings its own set of advantages and intricacies to the table. The research aims to conduct a comparative analysis, unravelling how these architectures handle the complexities inherent in medical imaging data. This involves scrutinizing their ability to capture intricate features, adapt to diverse tumour types, and navigate the trade-offs between model depth and computational efficiency.

Accuracy as a Benchmark: A crucial aspect of the evaluation revolves around accuracy as a benchmark. The research aims to quantify and compare the classification accuracy achieved by each architecture. This extends beyond a mere numerical assessment, delving into the specifics of correctly identifying tumour types and the models' resilience to variations within the dataset. The objective is to provide a nuanced understanding of how well each architecture aligns with the intricacies of brain tumour classification.

Model Interpretability: An often-overlooked yet crucial facet of the evaluation is model interpretability. The research aims to unravel the 'black box' nature of these architectures, shedding light on how well the models' decisions can be interpreted and understood by healthcare professionals. The objective is to explore avenues for enhancing interpretability, ensuring that the diagnostic decisions made by the models are not only accurate but also comprehensible to the experts who rely on them for patient care.

By achieving this objective, the research contributes valuable insights into the selection and optimization of CNN architectures for brain tumour classification. The goal is not only to identify the most accurate model but also to align the technological prowess of these architectures with the practical needs and interpretative capacities of healthcare professionals in real-world clinical scenarios.

5. Generalization and Robustness:

This pivotal objective revolves around ensuring that the advanced models showcased in this project report generalize effectively to unseen brain tumour information and demonstrate robust performance across distinct datasets and imaging modalities. The overarching aim is to create models that exhibit resilience in handling variations in tumour appearance, imaging protocols, and noise levels, establishing their practical applicability beyond the confines of specific training datasets.

The significance of this objective lies in the acknowledgment that the real-world deployment of models extends beyond the constraints of a controlled research environment. The research aims to validate the practicality and robustness of the developed models by focusing on two crucial aspects: generalization and resilience.

Generalization to Unseen Data: Ensuring that the models generalize well to unseen brain tumour data is a fundamental goal. This involves evaluating the performance of the models on datasets that were not part of the training process. The objective is to showcase that the models can adapt and extend their learned knowledge to new and diverse scenarios, reinforcing their reliability in handling a spectrum of brain tumour variations that may not have been explicitly encountered during the training phase.

Performance Across Different Datasets and Imaging Modalities: The research goes beyond the boundaries of a singular dataset, aiming to demonstrate the versatility of the developed models across different datasets and imaging modalities. This involves evaluating the models on diverse datasets with variations in imaging protocols, data acquisition methods, and noise levels. The objective is to showcase that the models can transcend dataset-specific idiosyncrasies and perform consistently across a broader spectrum of real-world scenarios.

Handling Variations in Tumour Appearance, Imaging Protocols, and Noise Levels: The crux of this objective lies in preparing the models to handle the inherent variations in brain tumour appearance, diverse imaging protocols, and varying levels of noise in the data. The research endeavours to equip the models with the adaptability to navigate through the intricacies of clinical imaging, ensuring that they can discern meaningful features even in the presence of variations arising from different imaging devices, protocols, and environmental factors.

By achieving success in generalization and robustness, the report aims to bridge the gap between theoretical advancements and practical applicability. The goal is to create models that not only perform well in controlled research settings but also prove their mettle in the dynamic and diverse landscape of real-world clinical scenarios, ultimately contributing to their adoption as valuable tools in healthcare decision-making.

6. Seamless Integration with Clinical Workflows:

This crucial objective is centred around the seamless integration of the developed models into existing clinical workflows. The primary goal is to ensure that the advanced models, particularly those leveraging CNN architectures like VGG16 and Inception V3, align harmoniously with the day-to-day practices of healthcare professionals. The objective extends beyond technical proficiency, emphasizing practical utility, interpretability, and user-friendliness in real-world clinical settings.

The successful integration of advanced models into clinical workflows represents a pivotal aspect of this project report. The research aims to bridge the gap between technological innovation and practical application, recognizing that the effectiveness of these models is contingent on their seamless incorporation into the existing routines of healthcare professionals.

User-Friendly Interface and Interpretability: The objective involves designing a user-friendly interface that facilitates easy interaction with the developed models. The research acknowledges the importance of interpretability in healthcare settings, and as such, endeavours to present diagnostic information in a clear and understandable manner. This not only enhances the models' acceptance among healthcare professionals but also ensures that the decisions made by the models are comprehensible and can be effectively communicated to patients.

Aligning with Clinical Decision-Making: The models are not intended to exist in isolation but rather as integral components of the clinical decision-making process. The research aims to understand the workflow nuances of healthcare professionals, ensuring that the models provide valuable insights at the right junctures. This involves aligning the predictions of the models with the decision points in a clinical workflow, facilitating a symbiotic relationship between the technological capabilities of the models and the nuanced decision-making expertise of healthcare professionals.

Practical Considerations for Deployment: Beyond technical proficiency, the research delves into practical considerations for deployment. This includes addressing computational efficiency, ensuring that the models operate within acceptable timeframes in a clinical setting. Moreover, ethical and legal considerations are taken into account, emphasizing the need for responsible AI deployment in the medical domain.

User Training and Support: Recognizing that successful integration requires user buy-in, the research considers the development of training materials and support systems for healthcare professionals. This ensures that users are adequately trained to interact with and interpret the outputs of the models, fostering a collaborative and informed approach to their incorporation into clinical workflows.

By achieving this objective, the research aims to propel these advanced models beyond the realm of theoretical innovation, making them tangible and valuable assets in the daily routines of healthcare professionals. The goal is not just to develop sophisticated models but to ensure their practical relevance, facilitating a paradigm shift in how diagnostic decisions are made in the dynamic and time-sensitive landscape of healthcare.

7. Validation and evaluation:

This critical objective is dedicated to the rigorous validation and evaluation of the developed CNN models using independent test datasets and appropriate assessment metrics. The overarching goal is to employ metrics such as accuracy, precision, recall, F1 score, and other relevant indicators to comprehensively assess the effectiveness of the models. By achieving these targets, the project aims to contribute significantly to the field of clinical imaging and brain tumour diagnosis. The ultimate purpose is to provide accurate, efficient, and reliable tools for automated tumour classification, thereby improving patient outcomes, facilitating timely treatment decisions, and supporting healthcare professionals in their diagnostic endeavours.

The culmination of the project is marked by the validation and evaluation of the developed CNN models. This phase represents the empirical foundation upon which the reliability and efficacy of the models are established.

Independence through Test Datasets: To ensure objectivity and real-world applicability, the research leverages independent test datasets. These datasets are distinct from the training data, providing an unbiased assessment of how well the models generalize to unseen information. The objective is to showcase the robustness of the developed models in handling novel cases, reinforcing their viability for deployment in diverse clinical scenarios.

Comprehensive Assessment Metrics: The evaluation process involves an array of assessment metrics carefully chosen to offer a holistic view of the models' performance. Metrics such as accuracy measure the overall correctness of predictions, while precision, recall, and the F1 score delve into the nuances of true positives, false positives, and false negatives. This multifaceted approach ensures that the models are evaluated not only on their ability to make correct predictions but also on their precision in identifying specific tumour types and their ability to minimize false positives and false negatives.

Comparison with Existing Methods: A pivotal aspect of the evaluation is the comparison of the developed models with existing methods. This involves benchmarking against established approaches in the field, providing a contextual understanding of the advancements achieved. The objective is to showcase how the developed models stand out in terms of accuracy, efficiency, and reliability, potentially offering superior solutions for automated tumour classification.

Contributions to Clinical Imaging and Brain Tumour Diagnosis: The ultimate ambition of this objective is to contribute meaningfully to the broader field of clinical imaging and brain tumour diagnosis. By providing accurate, efficient, and reliable tools for automated tumour classification, the research seeks to elevate the standards of diagnostic practices. The

objective extends beyond technological innovation, aiming to have a tangible impact on patient outcomes, treatment decisions, and the diagnostic methodologies employed by healthcare professionals.

Supporting Healthcare Professionals: A central tenet of this objective is to support healthcare professionals in their diagnostic approaches. The evaluation process not only validates the effectiveness of the models but also emphasizes their role as valuable assets in the diagnostic toolkit of healthcare professionals. The objective is to enhance the decision-making capabilities of medical experts, enabling them to make timely and well-informed decisions that positively impact patient care.

By achieving success in the validation and evaluation phase, the project aims to go beyond theoretical advancements, offering concrete and valuable contributions to the dynamic and crucial realm of clinical imaging and brain tumour diagnosis. The ultimate goal is to usher in a new era where automated tumour classification becomes a reliable and integral component of healthcare decision-making, fostering advancements that translate into improved patient outcomes and enhanced diagnostic practices.

PROPOSED INNOVATION:

The proposed innovation represents a pioneering approach to brain tumour detection, harnessing the capabilities of Convolutional Neural Networks (CNNs) to revolutionize the diagnostic landscape. This comprehensive system operates through a meticulously designed framework, encompassing data collection, CNN model development, training, evaluation, deployment, image processing, classification, and result presentation. Let's delve into each step, unravelling the intricacies of this groundbreaking innovation.

1. Data Collection and Preparation: Building a Robust Foundation

The genesis of our innovative system unfolds with an exhaustive and meticulous endeavour to amass a comprehensive dataset of brain MRI images. This isn't a mere compilation; it's a curated repository crafted with precision, where each image is meticulously labelled to discern between those that encapsulate brain tumours and those that remain devoid of such anomalies. The precision in labelling is not a trivial detail; it forms the bedrock, setting the tone for subsequent stages of model development and validation. Once assembled, this dataset undergoes a strategic division into two distinct subsets: the training set and the testing set.

This strategic division serves as a cornerstone, ensuring the creation of a robust foundation upon which our CNN model can learn and generalize. The training set becomes the fertile ground where the model delves into the intricacies of brain MRI images, extracting meaningful patterns and features that distinguish between normal and tumorous conditions. This learning process is foundational, equipping the model with the cognitive tools necessary for accurate classification. Simultaneously, the testing set remains untouched during the training phase, functioning as an independent litmus test for the model's generalization capabilities.

The strategic segregation of the dataset into training and testing sets is not a mere procedural formality; it's a deliberate architectural choice to guarantee the reliability and real-world applicability of our model. The model's proficiency isn't just measured by its ability to memorize the intricacies of the training set but is rigorously evaluated against entirely new instances. This meticulous approach to data collection and preparation is not just a prerequisite; it's a commitment to building a foundation that ensures the precision, adaptability, and robustness of our CNN model in the dynamic landscape of brain tumour classification.

2. CNN Model Development: Architecting Intelligence

At the heart of our innovative system lies the architectural marvel of a sophisticated Convolutional Neural Network (CNN) model, meticulously crafted to transcend the conventional boundaries of brain tumour classification. This model is not a mere amalgamation of layers; it is an intricate web where convolutional layers, pooling layers, and fully connected layers harmoniously converge, forming the backbone of our cognitive framework. The essence of innovation isn't just in the model's complexity but in its profound ability to orchestrate intelligence specifically tailored for the intricacies of brain MRI images.

The journey towards intelligence for our CNN model commences with iterative learning, a deliberate process where it systematically absorbs the nuanced details encapsulated within the diverse array of brain MRI images. Each convolutional layer acts as a perceptive lens, capturing essential features, while pooling layers distill this information, and fully connected layers weave a narrative of contextual understanding. This iterative refinement becomes the crucible where the model hones its ability to discern intricate patterns, laying the foundation for accurate predictions.

The model's adaptability and its unique capacity to learn hierarchical representations distinguish it as the cognitive cornerstone of our innovative system. It doesn't merely recognize features; it understands the hierarchical relationships among them, a quality that is pivotal in unravelling the complexity of brain tumour classification. The architecture's ability to absorb, adapt, and learn positions it as an intelligent entity, capable of navigating the nuances of medical imaging and contributing significantly to the precision and efficacy of our brain tumour classification system.

3. Training the CNN Model: Iterative Refinement for Precision

The transformative journey of our CNN model unfolds in the crucible of training, a dynamic process where it engages in a meticulous dance with labelled MRI images. In this intricate choreography of forward and backward passes, the model undergoes a continuous and delicate adjustment of its parameters. This dance is not just about learning the steps; it's a nuanced process that minimizes the dissonance between the model's predicted outputs and the ground truth labels embedded in each image. This iterative refinement isn't a one-time occurrence; it's a rhythmic evolution that transpires over multiple epochs.

At each step of the training process, the model absorbs the intricacies of brain MRI images, learning to discern patterns and features that are indicative of tumorous or normal conditions. The dynamic adjustment of parameters is akin to a fine-tuning mechanism, ensuring that the model aligns its internal representations with the complexities present in the training data. This meticulous process doesn't merely aim at memorization; it strives for a deeper understanding, enabling the model to generalize its knowledge to new, unseen images.

This iterative refinement is the crucible where our CNN model hones its predictive prowess. It doesn't just learn to recognize features; it becomes adept at generalizing this knowledge to handle the variations and nuances present in diverse medical imaging scenarios. Achieving precision in brain tumour classification isn't just about accuracy on the training set; it's about the model's ability to apply its learned insights to entirely novel instances. This facet of the training process is instrumental in ensuring that our CNN model transcends rote memorization, emerging as a sophisticated tool capable of making accurate predictions in the realm of brain tumour diagnostics.

4. Model Evaluation: Quantifying Precision and Reliability

The crucible of model evaluation represents a pivotal phase, a moment where the intricacies of our trained CNN model undergo a rigorous examination against an independent testing set. This set, intentionally distinct from the training data, serves as a litmus test, meticulously

scrutinizing the model's capacity to generalize and sustain its performance on entirely new and previously unseen brain MRI images. This phase is not a mere checkpoint; it is an indispensable measure of the model's adaptability and reliability in the face of diverse real-world scenarios.

The independent testing set acts as a stringent benchmark, challenging the model to extend its learned insights beyond the familiar terrain of the training data. This intentional separation ensures that the evaluation is not influenced by prior exposure, demanding the model to showcase its proficiency in making accurate predictions on entirely novel instances. The battery of performance metrics, including accuracy, precision, recall, and F1-score, assumes a critical role in this evaluation, providing a comprehensive and quantitative assessment of the model's diagnostic capabilities.

This rigorous evaluation is not just a ritual of validation; it's a testament to the robustness and reliability of our CNN model. It symbolizes the model's resilience to the complexities inherent in diverse medical imaging scenarios, where factors such as variations in tumour appearance, imaging protocols, and noise levels introduce challenges that demand precision. The metrics generated in this phase are not mere numbers; they represent a quantitative reflection of the model's capacity to perform consistently and accurately, reaffirming its potential for deployment in real-world healthcare contexts.

5. Model Deployment: Bridging the Gap to Practical Application

As the culmination of successful training and rigorous evaluation unfolds, our CNN model undergoes a transformative shift from theoretical prowess to tangible impact. It finds a purposeful abode on a Flask server, not merely as a hosting platform but as a dynamic bridge connecting sophisticated machine learning capabilities to an intuitive and user-friendly interface. This strategic deployment marks a pivotal juncture, where the model's potential is no longer confined to the experimental domain but opens up avenues for practical applications that can directly impact healthcare and diagnostics.

The Flask server emerges as a gateway, facilitating a seamless synergy between the intricacies of the CNN model and the end-users – be they medical professionals or individuals seeking diagnostic insights. This deployment is not a static placement; it's a deliberate act that transforms the CNN model into a tool accessible to those who can benefit from its cognitive capabilities. Through the user-friendly interface, individuals gain the power to upload their own brain MRI images, triggering a cascade of events that transcends the boundaries of theoretical innovation and steps into the realm of practical application.

6. Image Processing: Crafting Compatibility for Informed Analysis

Upon the initiation of user image upload, our system seamlessly transitions into a meticulous pre-processing stage, akin to a virtuoso performance orchestrating a harmonious symphony of adjustments. The primary objective at this juncture is to ensure absolute compatibility between the user-uploaded MRI images and the intricacies embedded within the deployed CNN model. This multifaceted process begins with the resizing of images, a thoughtful adjustment tailored to meet the precise specifications required by the model. This not only

aligns the images with the expected dimensions but also facilitates an optimal presentation of visual data for subsequent analysis.

7. Classification: Unleashing Cognitive Precision

The pre-processed MRI image embarks on a transformative journey through the deployed CNN model, an orchestra of algorithms unleashing cognitive precision. The model, now finely tuned and enriched with learned features, dissects the nuances of the brain image, making a precise prediction regarding the presence or absence of a tumour. The model's output is not a mere binary classification; it represents a culmination of cognitive processes, providing valuable insights into the diagnostic outcome with unprecedented precision.

8. Result Presentation: Empowering Informed Decision-Making

The pinnacle of our innovation lies in empowering users with timely and comprehensible information. The classification result, a culmination of the model's cognitive prowess, is presented to the user through an intuitive interface or a responsive message. This feedback is not just a notification; it is a beacon of informed decision-making. Whether in a clinical setting or at the fingertips of an individual seeking answers, this presentation of results ensures that users are equipped with the knowledge necessary to make timely and informed decisions, potentially expediting crucial medical interventions and significantly impacting patient outcomes.

HYPOTHESIS, DESIGN AND METHODOLOGY

❖ METHODOLOGY:

PYTHON:

Python is a high-level, interpreted programming language renowned for its simplicity and readability. It stands out as an object-oriented language with dynamic semantics, incorporating built-in data structures, dynamic typing, and automatic memory management. Python's appeal lies in its capacity for rapid application development and its utility as a scripting or glue language for interfacing with existing components. The language features clear and concise syntax, minimizing the cost of program maintenance.

Python supports modules and packages, fostering code modularity and reuse. The language's extensive standard library, available in precompiled and integrated form, further contributes to its versatility. Python is freely distributed for various platforms, making it accessible and cost-effective. Programmers appreciate Python's expressive power, and its lack of a compilation step facilitates a rapid edit-test-debug cycle.

One of Python's strengths is its robust exception handling. In the absence of compilation, errors lead to exceptions, preventing segmentation faults. The interpreter promptly detects errors, raises exceptions, and, if unhandled, prints a detailed stack trace. This approach ensures that bugs or unexpected data do not lead to program crashes.

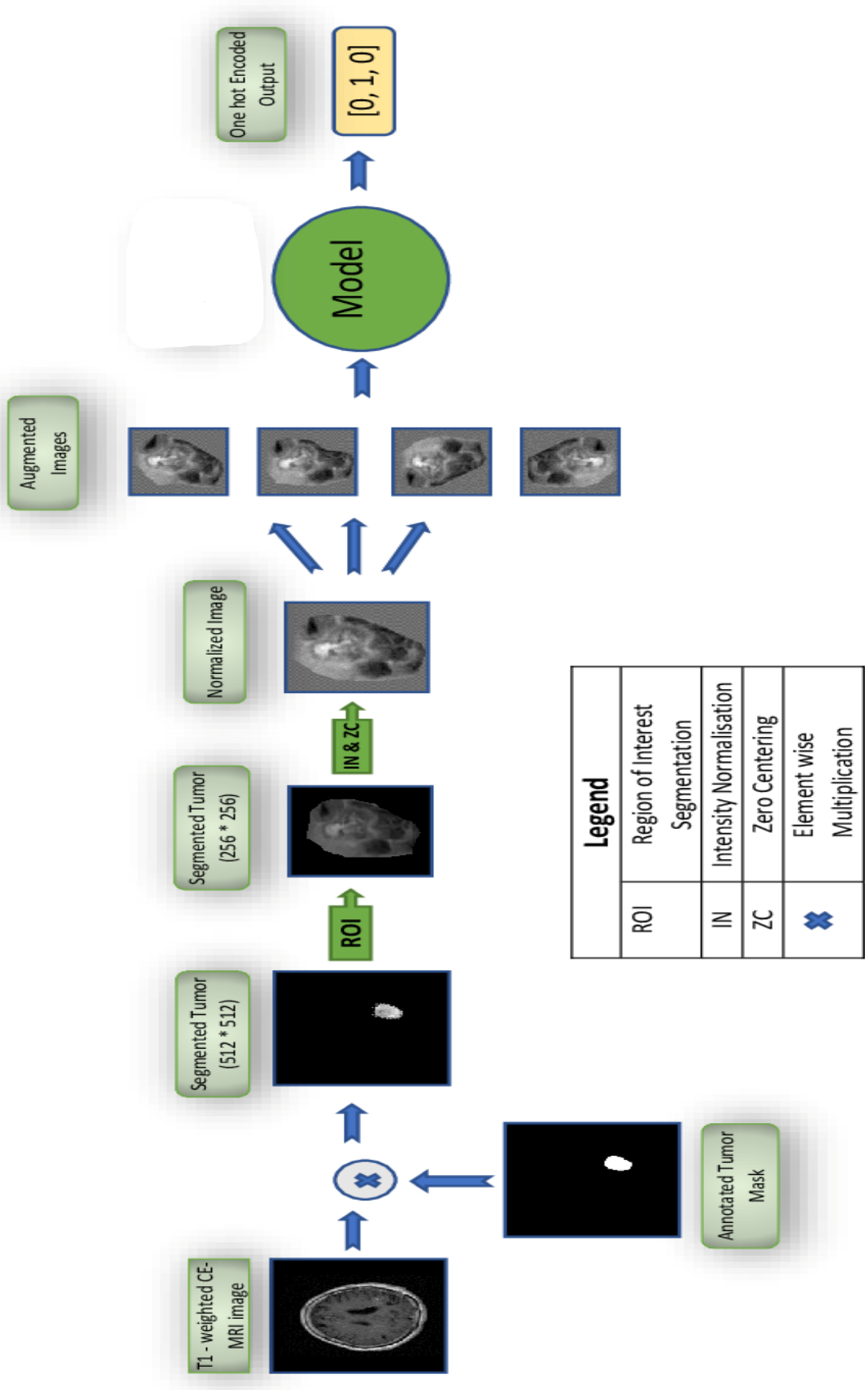
Python's interpreter offers a comprehensive environment for program examination and debugging. It allows the inspection of local and global variables, evaluation of expressions, setting breakpoints, and step-by-step code traversal. The language's simplicity facilitates quick modification and testing, making Python an exceptionally powerful tool for diverse programming tasks.

❖ ALGORITHM USED:

Image classification algorithms are a specialized class of models designed to categorize images into predefined classes. This classification process is particularly crucial for applications like image recognition and diagnosis. The primary objective is to discern the specific category or label associated with a given image. In the context of our project, we employ four prominent algorithms for image classification: **CNN, Inception V3, and VGG16.**

Our approach involves implementing these three algorithms in the context of brain tumour classification. Through rigorous testing and evaluation, we aim to determine the algorithm that exhibits superior performance in accurately classifying brain MRI images. The selection of the most effective algorithm is pivotal for the success and reliability of our brain tumour detection system.

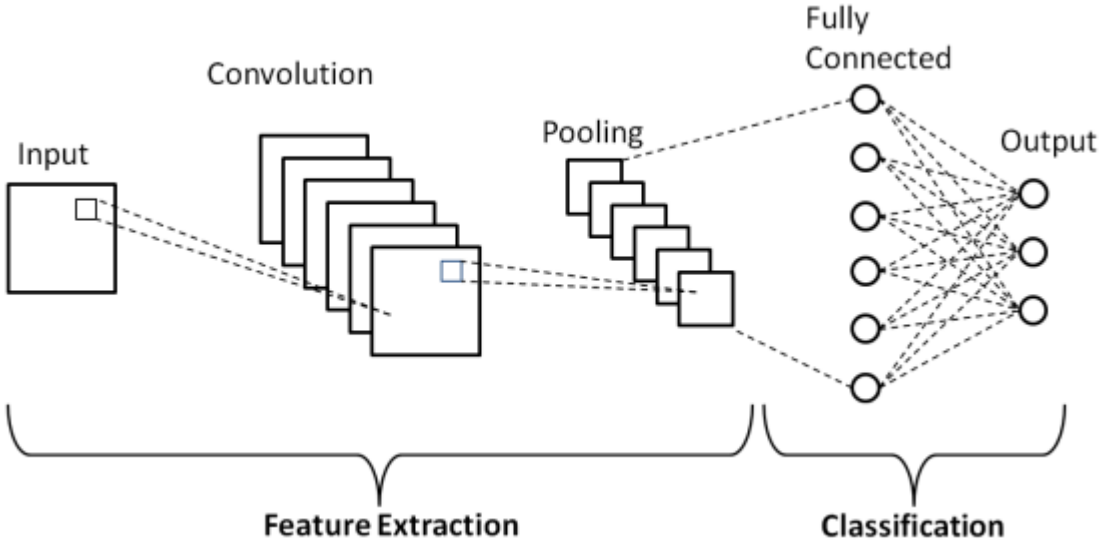
Overall, in general our workflow with any model is shown in this diagram:



CNN

Convolutional Neural Networks (CNNs) represent a specialized category of deep neural networks meticulously crafted for the intricate processing of structured grid data, predominantly images. Their robust efficacy has been extensively demonstrated across a spectrum of computer vision applications, encompassing image classification, object detection, and segmentation. The conceptual underpinning of CNNs draws inspiration from the intricate visual processing mechanisms inherent in the human brain. This design choice empowers CNNs to autonomously and adaptively glean spatial hierarchies of features from input data, mirroring the nuanced complexities of visual information processing in biological systems.

There are a lot of functions/operations which are used in the convolutional neural networks algorithms some of them are convolution, activation, padding pooling etc. Below we have mentioned about few important and most used functions/operations.



- **CONVOLUTION:**

Convolution is a mathematical operation that is performed on two functions, typically on an image and a small matrix called a kernel or filter. The operation involves sliding the kernel over the input image and computing a dot product between the values of the kernel and the corresponding pixels in the image. The sliding is done with the help of strides, which are nothing but the number of pixels a filter will skip after one iteration. This process is repeated at every position in the input image, resulting in a new matrix of values called a feature map or activation map. The convolution operation can be seen to extract local features from the input image by scanning the image with the kernel. The kernel typically has smaller dimensions than the input image and contains learned parameters that are adjusted during the training process to detect specific features, such as edges, comers, or textures. By applying multiple convolutional filters in parallel, a convolutional neural network (CNN) can learn to detect complex features and patterns in images.

- **PADDING:**

It is used in CNNs to preserve the spatial dimensions of an input image when performing convolution. As we now know that in convolution a small filter is slid over the input image to extract features. If the filter size is larger than the input image size, the output feature map would be smaller than the input image, which can result in information loss. Padding involves adding a border of zeros (or some other value) around the input image before convolution, such that the output feature map has the same spatial dimensions as the input image. Padding ensures that the kernel can be centred at every position in the input image. There are two types of padding:

1) Valid padding: In this type of padding, no padding is added to the input image, and the kernel is only applied to positions where it completely overlaps with the input image.

2) Same padding: In this type of padding, zeros (or some other value) are added to the input image such that the output feature map has the same spatial awareness as the input image.

- **POOLING:**

Pooling is used to reduce the spatial dimensions of an input feature map while retaining its important features. Pooling is typically applied after convolution and activation to progressively reduce the size of the feature maps and improve the robustness of the model. The pooling operation involves partitioning the input feature map into non-overlapping regions and computing a summary statistic (e.g., maximum, average, or sum) of each region to obtain a smaller output feature map. The summary statistic is chosen based on the specific application and can help to retain the most important information in the input feature map. The main types of pooling are mentioned below:

1) Max pooling: In max pooling, the maximum value of each region is taken as the summary statistic. This type of pooling is used when we want to detect some noticeable features such as edges or corners.

2) Average pooling: In average pooling, the average value of each region is taken as the summary statistic. This type of pooling is useful for preserving the overall structure of the input feature map and reducing noise

Pooling helps in reducing the dimensions of the input and makes the input prone in overfitting and makes the task of processing easy. However, pooling can also result in information loss and may reduce the spatial resolution of the output feature map.

- **ACTIVATION FUNCTIONS:**

Activation functions are mathematical functions used to introduce non-linearity into the model. They are typically applied after the convolution and pooling layers to transform the output of each neuron into a non-linear representation that can better capture the complex relationships between the input and output Activation functions take a single input and for that

input based on the activation function used it generates an output between 0 and 1 or between -1 and 1. Some commonly used activation functions are mentioned below:

1) Rectified Linear Unit (ReLU): The ReLU function is defined as fix) = max (0, x). ReLU sets all negative input values to zero, which introduces non- linearity and sparsity into the model.

2) Sigmoid: The sigmoid function is defined as f(x)=1/(1+e^x). It takes any input value and gives back an output value between 0 to 1. And hence it can be used in scenarios where we have binary classification. A value less than 0.5 can be treated as false and greater than 0.5 as true or any other class on which you are performing the classification.

3) Tanh: The tanh function is defined as f(x) (e^x-e^x)/(e^x+e^x). It maps any input value to a value betlen-1 and 1, which can help to centre the input data around zero and improve the convergence of the model.

4) SoftMax: The SoftMax function is used for multi-class classification tasks and maps a vector of input values to a vector of probabilities that sum to one. It is defined as f(x)=e^xi/sum(e^xi), where xi is the ith element of the input vector.

- ## DROPOUT REGULARISATION:

Dropout regularization is used in deep learning to reduce overfitting in neural networks. It is a form of regularization that works by randomly dropping out (le, setting to zero) some of the neurons in the network during training. The idea behind dropout is to prevent the neurons from co-adapting to each other by reducing noise into the network and forcing each neuron to learn more 31 independent features. During training, dropout is applied to a certain percentage of neurons in each layer with a probability value between 0 and 1. The dropout rate is typically set between 0.2 and 0.5, which means that 20-50% of the neurons are randomise dropped out during training. Dropout can be applied to any layer of a neural network, including convolutional layers, fully connected layers, and current layers. However, it is important to note that dropout should only be used during training and not during testing. During testing, the full network with all serons should be used to obtain accurate predictions.

- ## FLATTENING:

Flattening is a process to convert multi-dimensional arrays or tensors into one dimensional array. It is typically used after the convolution and pooling layers to prepare the data for the fully connected layers. To pass this output through the fully connected layers, which require a one-dimensional input, the tensor needs to be flattened into a one-dimensional array. The process of flattening amply involves reshaping the multi-dimensional tensor into a one-dimensional array Flattening is a simple and important step in CNNs that enables the output of the convolutional and pooling layers to be passed through the fully connected layers for classification or regression tasks. It ensures that the output data is in the correct format for the subsequent layers of the network to process.

- **FULLY CONNECTED LAYER:**

A fully connected layer, also known as a dense layer, is a type of layer in which each neuron in the layer is connected to every version in the previous layer. The input to a fully connected layer is typically a flattened vector of the output of the previous convolutional or pooling layer. In a fully connected layer, each neuron performs a linear operation on its input, followed by an activation function that introduces non-linearity into the output. The output of a fully connected layer is typically fed into another fully connected layer or a final output layer, depending on the specific task of the neural network. Fully connected layers are commonly used in classification tasks, where the final 32 output layer is a SoftMax layer that produces a probability distribution over the classes.

Advantages of CNN:

- Spatial Hierarchical Feature Learning: CNNs automatically learn hierarchical representations of spatial features, capturing both low-level and high-level patterns in data.
- Translation Invariance: CNNs leverage weight sharing through convolutional layers, providing translation-invariant features, making them robust to variations in object positions.
- Effective for Image Processing: CNNs excel in image-related tasks due to their ability to recognize spatial patterns and hierarchies of features.
- Reduced Parameter Dependence: Parameter sharing reduces the number of trainable parameters compared to fully connected networks, making CNNs more efficient.

Disadvantages of CNN:

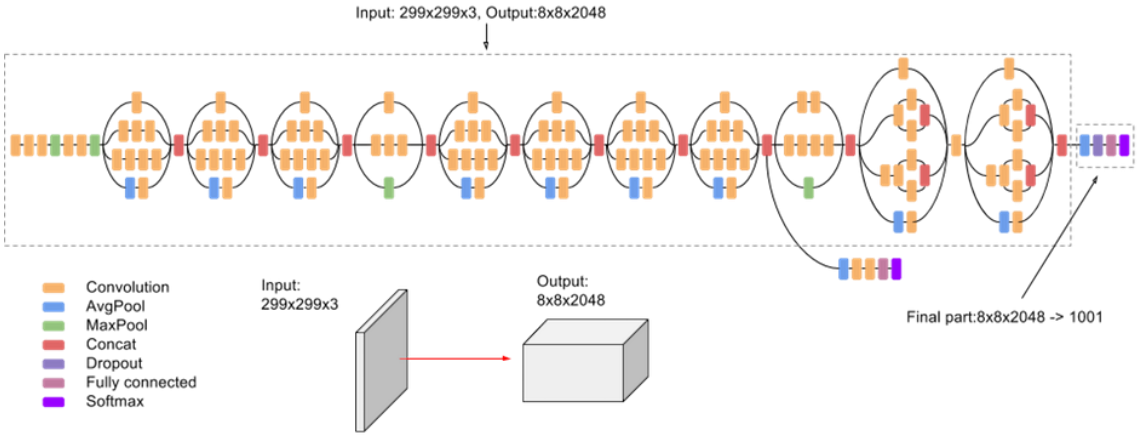
- Computational Intensity: CNNs can be computationally demanding, especially with deeper architectures. Adequate computational resources may be required for training and inference.
- Interpretability: Understanding the inner workings of complex CNN architectures may be challenging, and interpretability can be limited compared to simpler models.
- Data Size: CNNs may require large amounts of labelled data for effective training. Transfer learning can be employed to mitigate data limitations.

In summary, CNNs are powerful tools for image-related tasks, leveraging spatial hierarchies to automatically learn features. They are a crucial component in the methodology, complementing transfer learning models for comprehensive analysis in brain tumour classification.

Inception v3:

Inception-v3, an evolution of the original Inception architecture (GoogleNet), was introduced by Google researchers in 2015. This convolutional neural network (CNN) architecture stands out for its innovative approach to feature extraction and parameter reduction, making it a formidable choice for image classification tasks.

Architectural Design: The Inception-v3 architecture is designed around the concept of utilizing multiple paths or "modules" within the network. Each module focuses on capturing information at different levels, enabling the extraction of both intricate details and global context information from input images. This multi-path approach enhances the network's ability to comprehend diverse features.



Key Innovations:

1. **Factorization Methods:** Inception-v3 incorporates factorization methods, replacing large convolutional filters with smaller ones. This technique effectively reduces the number of parameters in the network, addressing concerns of overfitting. It also enhances training efficiency.
2. **Modules and Paths:** The architecture features modules designed to extract features at various scales and resolutions. This enables the network to process images comprehensively, capturing nuanced details and overall context simultaneously.
3. **Batch Normalization:** Batch normalization is extensively applied in Inception-v3, contributing to stable and accelerated training. Activation inputs undergo batch normalization, optimizing the learning process.
4. **Loss Computation:** SoftMax is employed for loss computation, ensuring effective classification outcomes for different datasets with varied outputs.
5. **Data Preprocessing:** The preprocessing involves resizing images from 1024x1024 to 224x224. Transfer learning is employed using pretrained models, particularly effective for small datasets, as Inception-v3 has been trained on extensive datasets with high processing power.

Advantages and disadvantages of Inception v3:

Advantages:

1. **State-of-the-Art Performance:**
    - Inception-v3 has demonstrated excellence in achieving state-of-the-art results across diverse image recognition tasks, including the challenging domain of brain tumour disease imaging datasets.
2. **Multiscale Feature Capture:**
    - The architecture's inception modules excel at capturing features of varying scales and resolutions. This multiscale approach enhances the model's accuracy by comprehensively analyzing both intricate details and broader contextual information in images.
3. **Efficient Parameter Reduction:**
    - Leveraging factorization methods, Inception-v3 reduces the number of parameters in the network. This efficiency is crucial for training and running models on brain tumour disease images, particularly when computational resources are limited.
4. **Transfer Learning Capability:**
    - Inception-v3 supports transfer learning, making it valuable for brain tumour disease imaging datasets with a scarcity of labelled data. The pretrained model's ability to adapt to specific tasks aids in achieving meaningful results with limited training samples.

Disadvantages:

1. **Overfitting Concerns:**
    - When trained on small datasets, especially with very deep networks, Inception-v3 may be prone to overfitting. This is a challenge in scenarios where brain tumour disease datasets have limited samples, potentially leading to inaccurate predictions.
2. **Complex Architecture:**
    - The architecture of Inception-v3 is intricate, making it challenging to interpret how the model arrives at its predictions. This lack of interpretability poses difficulties, particularly in brain tumour disease imaging datasets where transparency and understanding of predictions are crucial.
3. **Limited Flexibility:**
    - Inception-v3 might lack the flexibility to capture the full range of variability in brain tumour disease images. Variations in imaging modalities or patient populations may not be fully accommodated, limiting the model's adaptability to diverse datasets.

VGG16:

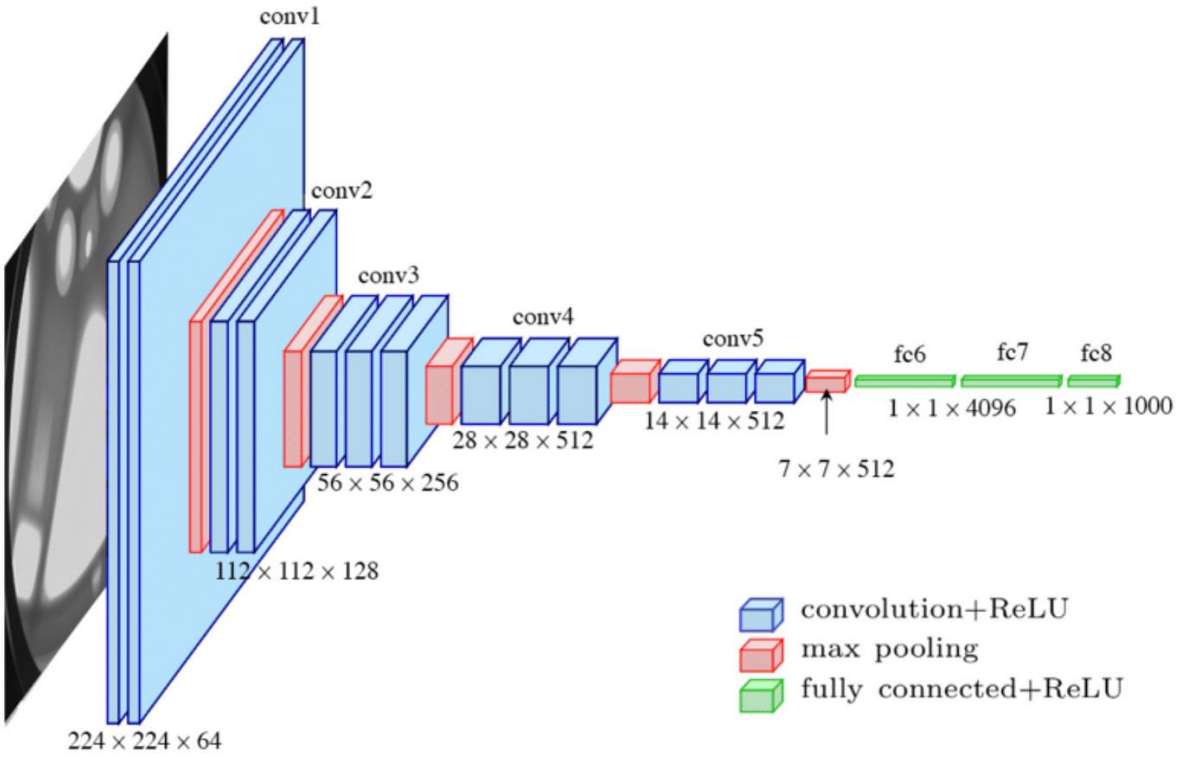
VGG-16, developed by the Visual Geometry Group (VGG) at the University of Oxford, emerged in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Recognized for its effectiveness in image classification tasks, VGG16 has become a cornerstone in convolutional neural network (CNN) architectures.

Key Components:

1. **Layer Configuration:**
    - VGG16 boasts a deep architecture with a total of 16 convolutional layers. Each layer applies a set of filters to the input data, followed by a non-linear activation function. This hierarchical structure enables the extraction of increasingly complex features from the input data.
2. **Down sampling and Feature Preservation:**
    - Following each convolutional layer, down sampling or max-pooling operations are employed to reduce the spatial dimensions while preserving crucial features. This approach allows VGG16 to effectively handle image classification tasks even with relatively small datasets.
3. **Fully Connected Layers:**
    - VGG16 incorporates two fully connected (FC) layers towards the end of its architecture. These layers play a crucial role in making predictions based on the extracted features. Neurons in the FC layers are interconnected by weights and biases, facilitating the learning of intricate relationships between different parts of images.
4. **Hyperparameters:**
    - VGG16 includes various hyperparameters, such as learning rate, momentum terms, and weight decay terms. Proper tuning of these parameters is essential to optimize the model's performance on specific datasets or tasks.
5. **Simplicity and Effectiveness:**
    - Despite its depth, VGG16 is appreciated for its simplicity compared to many contemporary deep learning architectures. This simplicity contributes to its versatility and ease of understanding, making it a powerful tool for diverse image classification tasks.

Functional Workflow:

VGG16 follows a systematic workflow, starting with the application of convolutional filters, subsequent down sampling to preserve essential features, and culminating in fully connected layers for accurate predictions. This process enables the network to learn hierarchical representations of input images.



VGG16's architecture, characterized by its layer depth, down sampling strategies, and fully connected layers, positions it as a robust solution for image classification tasks. Its simplicity, coupled with effective feature extraction, makes VGG16 a valuable asset in the realm of deep learning architectures.

Advantages and Disadvantages of VGG16:

Advantages:

1. **Effective Feature Extraction:**
    - VGG16's deep architecture facilitates the extraction of intricate features from input images. The hierarchical structure allows the model to learn and represent complex patterns, making it effective for image classification tasks.
2. **Versatility and Generalization:**
    - The simplicity of VGG16 contributes to its versatility, making it applicable across various image-related tasks. Its ability to generalize well on different datasets and tasks is a notable advantage.
3. **Interpretability:**
    - Compared to more complex architectures, VGG16's straightforward design enhances interpretability. This makes it easier for researchers and practitioners to understand and analyse the inner workings of the model.
4. **Availability of Pre-trained Models:**

- Pre-trained VGG16 models on large datasets, such as ImageNet, are readily available. This availability facilitates transfer learning, allowing practitioners to leverage pre-trained weights for improved performance on specific tasks, even with limited labelled data.

5. **Simplicity in Implementation:**
    - VGG16's architecture is relatively simple compared to some deep learning models, making it easier to implement, train, and fine-tune. This simplicity can be advantageous for researchers and developers.

Disadvantages:

1. **Computational Intensity:**
    - VGG16's depth and the number of parameters make it computationally intensive. Training and running the model may require substantial computational resources, which can be a limitation for applications with constraints on processing power.
2. **Prone to Overfitting:**
    - VGG16, especially when applied to smaller datasets, may be susceptible to overfitting. The model's deep architecture can memorize training data, leading to reduced performance on unseen data. Regularization techniques may be necessary to mitigate this issue.
3. **Large Memory Footprint:**
    - The extensive layering in VGG16 contributes to a large memory footprint, requiring more memory for storage and processing. This can be a concern, particularly for applications with limited memory resources.
4. **Limited Receptive Field:**
    - The use of small-sized filters throughout the network results in a limited receptive field. While this contributes to effective feature learning, it may not capture global context well, impacting the model's performance on certain tasks.

In summary, while VGG16 offers effective feature extraction, versatility, and interpretability, considerations such as computational intensity, susceptibility to overfitting, large memory footprint, and limited receptive field should be taken into account when choosing the model for specific applications.

DESIGN

FLOW-DIAGRAM:

Flow diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, Flow diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. A Flow diagram shows the overall flow of control. Flow diagrams are constructed from a limited repertoire of shapes, connected with arrows. Flow diagrams are constructed from a limited repertoire of shapes, connected with arrows.

In the context of our brain tumour classification project, the flow diagram stands as a crucial schematic representation, encapsulating the orchestrated sequence of operations that guide the system from data input to insightful output.

Key Components of the Flow Diagram:

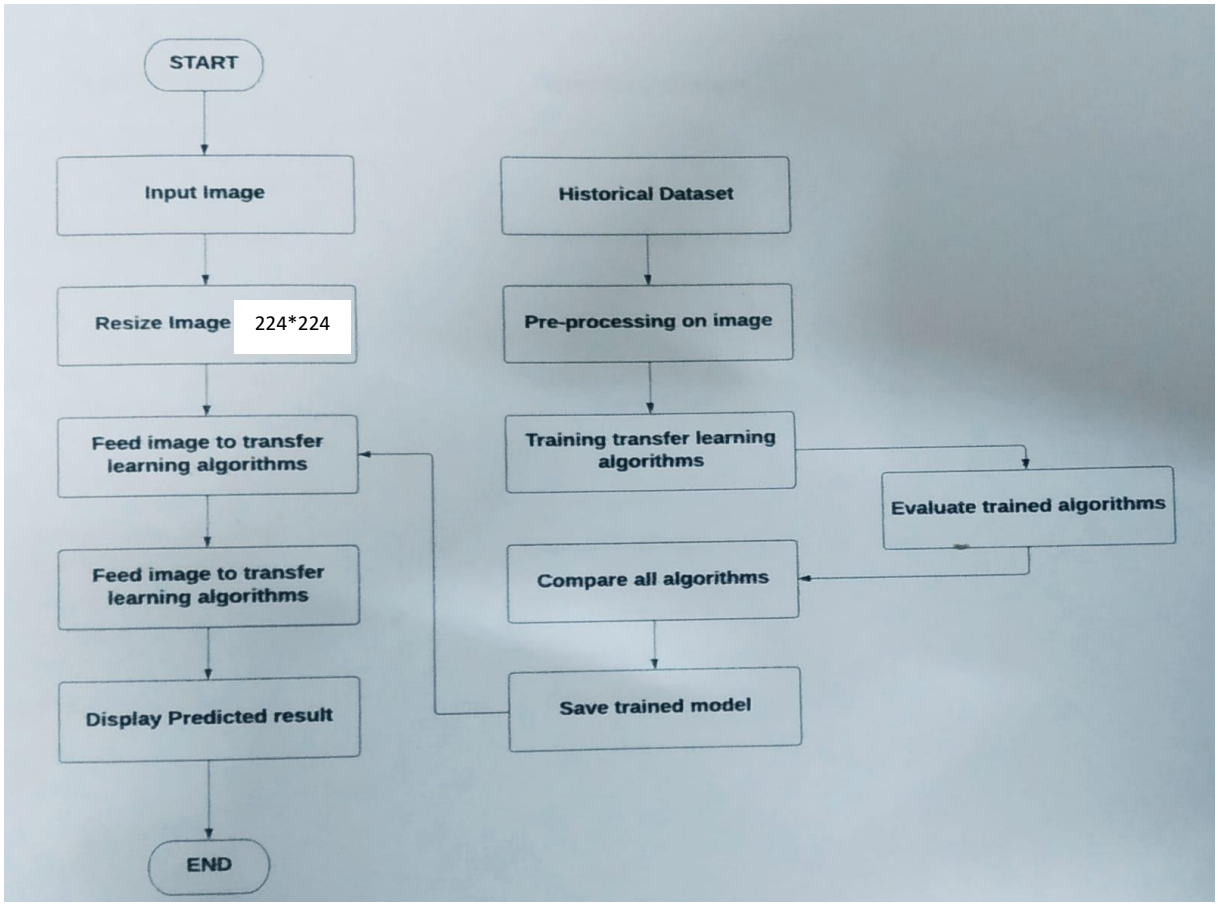
1. **Rectangles for Flow Representation:** The fundamental building block, represented by rectangles, captures the flow of activities within the system. Each rectangle encapsulates a specific task or operation, creating a visual narrative of the sequential progression through different stages of the brain tumour classification process.
2. **Diamonds for Decision Points:** Diamonds embedded within the flow diagram symbolize decision points, where the system evaluates conditions and determines the subsequent course of action. These decision points are critical junctures that introduce a level of adaptability and intelligence into the workflow, allowing the system to dynamically respond to varying inputs.
3. **Bars for Concurrent Activities:** Bars play a dual role in the flow diagram. They signify both the initiation (split) and conclusion (join) of concurrent activities. This visual cue emphasizes the parallel execution of certain tasks, enabling the system to handle multiple operations simultaneously and enhancing overall efficiency.
4. **Rectangles for Workflow States:** The flow diagram incorporates rectangles to signify the start (initial state) and end (final state) of the workflow. These rectangles demarcate the system's entry point, where the journey begins, and the exit point, where the system culminates its processing, presenting the final outcomes.
5. **Arrows for Sequential Order:** Arrows, the dynamic connectors within the flow diagram, illustrate the sequential order in which activities unfold. These arrows provide a directional guide, outlining the logical progression from one task to the next. The interconnectedness of these arrows constructs a narrative that mirrors the system's chronological execution.

Constructing the Flow Diagram:

In the construction of our flow diagram, these distinct shapes harmonize to convey a cohesive representation of the brain tumour classification workflow. As the rectangles unfold, they delineate the specific tasks involved in data processing, model training, image analysis, and result presentation. Decision diamonds interject, steering the flow based on conditions such as model evaluations or user inputs.

Bars strategically punctuate the diagram, marking the simultaneous execution of tasks where concurrency plays a pivotal role. Concurrent activities, initiated and joined by these bars, underscore the system's ability to handle diverse operations concurrently, enhancing overall efficiency and responsiveness.

The rectangles denoting workflow states encapsulate the initiation and conclusion points, anchoring the flow in a comprehensible structure. Arrows gracefully weave through the diagram, guiding us through the intricacies of the system's logical progression.



ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. An activity diagram shows the overall flow of control. Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows.

Key Components of the Activity Diagram:

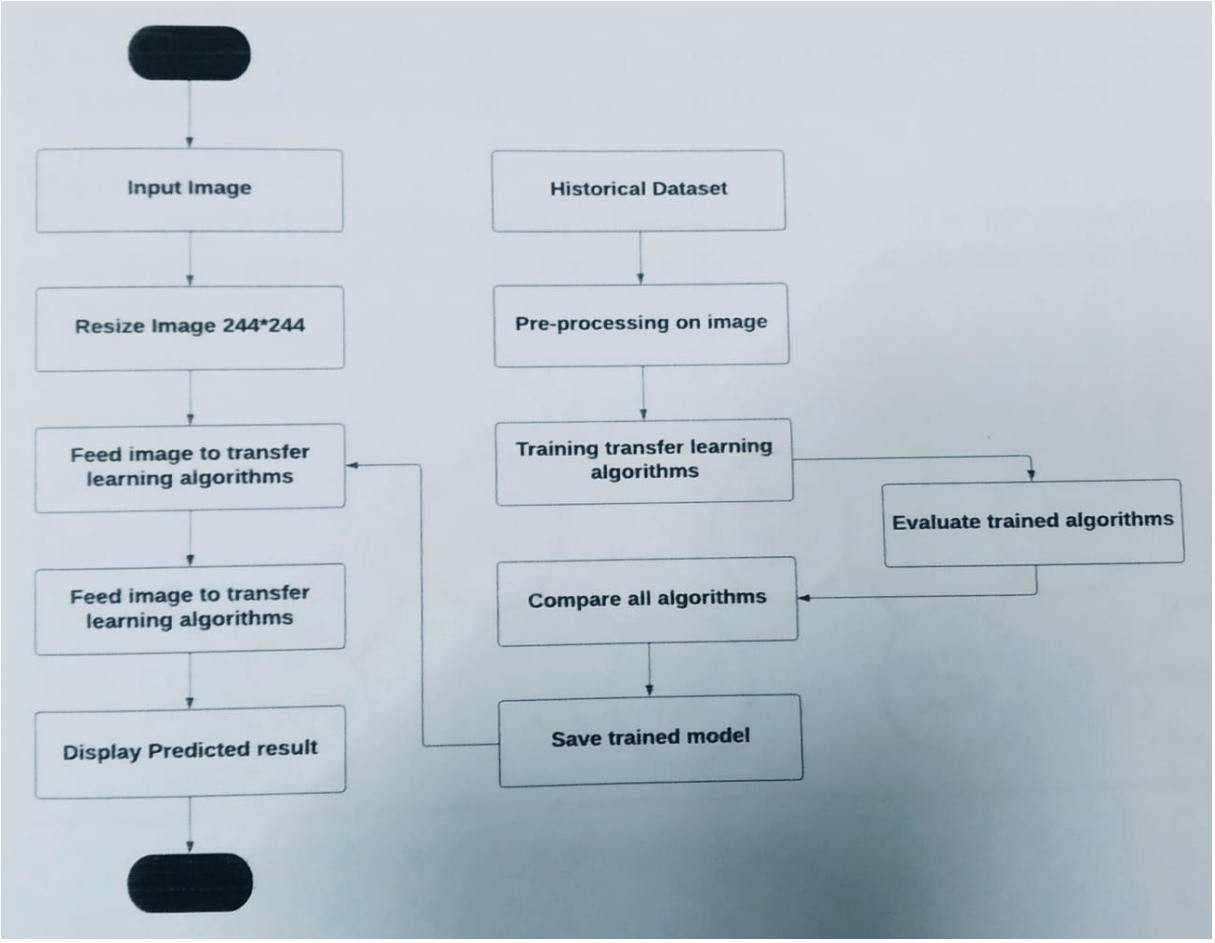
1. **Rounded Rectangles for Activity Representation:** The heartbeat of the activity diagram is the rounded rectangle, symbolizing individual activities within the system. Each rounded rectangle encapsulates a specific task or operation, delineating the stepwise progression through the brain tumour classification workflow.
2. **Diamonds for Decision Points:** Embedded diamonds within the activity diagram signify decision points, where the system evaluates conditions and dynamically determines the subsequent course of action. These decision diamonds infuse adaptability into the workflow, allowing the system to respond intelligently to varying inputs.
3. **Bars for Concurrent Activities:** Bars in the activity diagram mark the initiation (split) and conclusion (join) of concurrent activities. These bars visually emphasize parallel execution, showcasing the system's ability to perform multiple tasks concurrently, optimizing efficiency and responsiveness.
4. **Black Circles for Workflow States:** A black circle designates the commencement (initial state) of the workflow, while an encircled black circle denotes the culmination (final state). These symbols demarcate the entry and exit points of the system's workflow, anchoring the dynamic activities within a coherent structure.
5. **Arrows for Sequential Order:** Dynamic arrows interconnect the various shapes in the activity diagram, illustrating the sequential order of activities. The arrows guide the viewer through the logical progression, portraying the flow of control and the order in which activities unfold.

Constructing the Activity Diagram:

The construction of our activity diagram involves a harmonious interplay of these distinct shapes, weaving a narrative that unfolds the intricacies of the brain tumour classification workflow. Rounded rectangles articulate individual activities, guiding us through processes like data collection, model training, and result presentation. Decision diamonds introduce forks in the flow, steering the system based on conditions like model evaluations or user interactions.

Bars strategically punctuate the diagram, delineating the initiation and conclusion of concurrent activities. The black and encircled black circles encapsulate the initiation and

conclusion points, anchoring the workflow's beginning and end in a visually cohesive manner. Arrows gracefully interconnect the shapes, outlining the logical progression and facilitating a comprehensive understanding of the system's dynamic functionality.



Data Flow Diagram (DFD):

Data Flow Diagram (DFD) is a graphical representation of data flow in any system. It is capable of illustrating incoming data flow, outgoing data flow and store data. Data flow diagram describes anything about how data flows through the system. Sometimes people get confused between data flow diagram and flowchart. There is a major difference between data flow diagrams and flowchart. The flowchart illustrates flow control in program modules. Data flow diagrams illustrate flow of data in the system at various levels. Data flow diagram does not have any control or branch elements. A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one.

DFD rules and tips

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

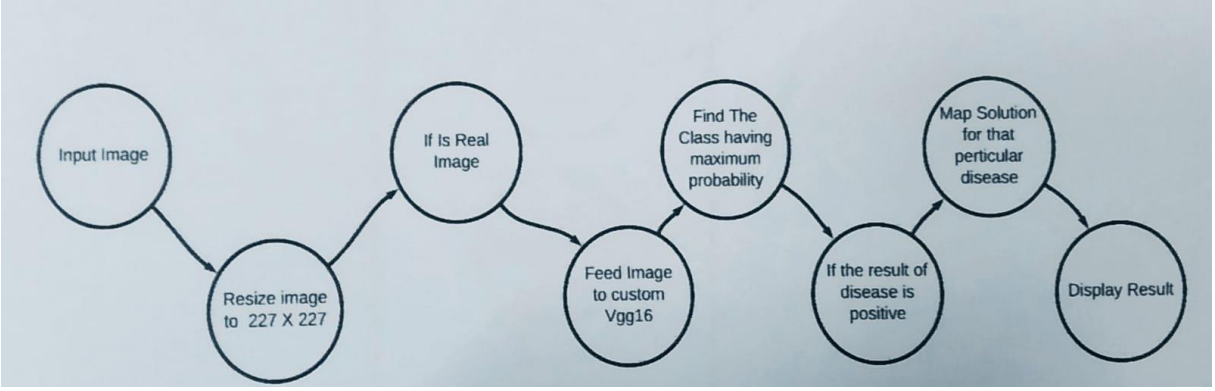
Components of Data Flow Diagram:

1. **Entities:** Gateways of Data Interaction Entities serve as the source and destination points for the data within the system. These are visually represented by rectangles, each adorned with its corresponding name. Entities essentially encapsulate external elements interacting with the system, such as users or external databases.
2. **Process:** The Heartbeat of Data Transformation Processes are the tasks or operations performed on the data within the system. Represented by circles, processes denote the dynamic transformations or manipulations undergone by the data. Occasionally, rounded edge rectangles may be employed to symbolize processes, providing a visual cue to the data's journey through transformation.
3. **Data Storage:** Safeguarding the Data Repository Data storage points encapsulate the databases or repositories within the system. Represented by rectangles with both smaller sides missing or enclosed within two parallel lines, these storage components safeguard the data awaiting retrieval or future utilization. Each rectangle signifies a distinct storage entity within the system architecture.
4. **Data Flow:** The Pulsating Arteries of Information Data flow denotes the movement of data within the system, capturing the essence of its journey from source to destination. Depicted by arrows, with the tail signifying the source and the head indicating the destination, these arrows form the pulsating arteries of information flow.

The path traced by these arrows vividly illustrates how data traverses through entities, processes, and storage points.

Constructing a Data Flow Diagram:

The construction of a DFD involves orchestrating these components in a synchronized fashion to portray the comprehensive data dynamics within the system. Entities mark the entry and exit points of data, processes showcase the manipulations it undergoes, data storage points safeguard its repository, and data flows trace its trajectory. The interplay of these elements forms a visual narrative, unravelling the intricate pathways of data movement.



USE CASE DIAGRAM:

In UML, use-case diagrams model the behaviour of a system and help to capture the requirements of the system.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

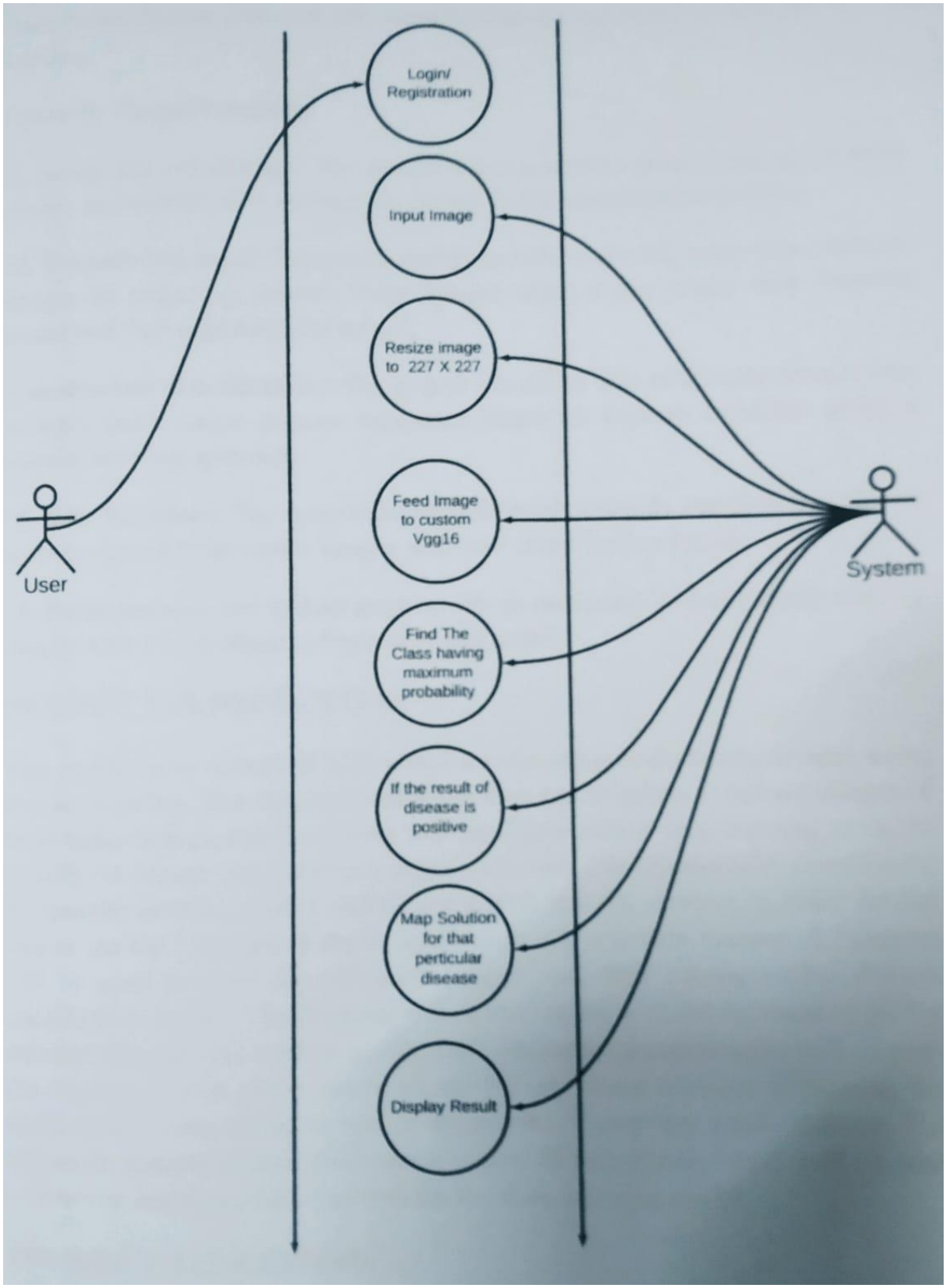
Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.

Use-case diagrams are helpful in the following situations:

- Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.
- While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.
- During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.
- During the testing phase, you can use use-case diagrams to identify tests for the system.

Components of Use Case Diagram:

- **Use cases**: A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system.
- **Actors**: An actor represents a role of a user that interacts with the system that you are modelling. The user can be a human user, an organization, a machine, or another external system.
- **Subsystems**: In UML models, subsystems are a type of stereotyped component that represent independent, behavioural units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modelling.
- **Relationships in use-case diagrams**: In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behaviour between the model elements.


Login/
Registration
Input Image
Resize image
to 227 X 227
Feed Image
to custom
Vgg16
Find The
Class having
maximum
probability
If the result of
disease is
positive
Map Solution
for that
perticular
disease
Display Result
User
System

REQUIREMENT ANALYSIS

Below are some examples of functional and non-functional requirements for a brain tumour disease detection and classification system based on transfer learning.

Feature Requirements:

1. **Image Pre-processing: Enhancing Data for Classification Precision**

Image pre-processing is a pivotal feature, ensuring that the system can handle brain tumour images effectively. This process involves a series of steps to refine and enhance the raw images, extracting relevant features that are crucial for accurate classification. Tasks such as resizing, normalization of pixel values, and potentially applying transformations ensure that the input data is optimized for the subsequent stages of the classification process. This feature plays a foundational role in preparing the data for the intricate analysis carried out by the system.

2. **Transfer Learning: Leveraging Pre-trained Models for Enhanced Accuracy**

The system incorporates transfer learning, a sophisticated technique where a pre-trained deep learning model's knowledge is harnessed and fine-tuned for the specific task of brain tumour classification. This feature allows the system to benefit from the wealth of information captured by established models such as VGG16, or Inception V3. By adapting these models to the nuances of brain tumour images, the system can achieve a higher level of accuracy and efficiency in classification, showcasing the synergy between pre-existing knowledge and task-specific adaptation.

3. **Multi-class Classification: Comprehensive Categorization of Brain Tumours**

Multi-class classification is a key requirement, reflecting the system's capability to categorize brain tumour images into multiple disease categories. Leveraging the features extracted through transfer learning, the system can discern intricate patterns associated with different types of brain tumours. This feature enhances the diagnostic potential of the system by providing a nuanced classification that goes beyond a binary output, allowing for a more comprehensive understanding of the diverse range of brain tumour diseases.

4. **User Interface: Seamless Interaction for Enhanced User Experience**

The system boasts a user-friendly interface, designed to facilitate effortless interaction for users ranging from medical professionals to individuals seeking a diagnosis. This feature ensures a streamlined process where users can easily upload brain tumour images for analysis. The interface provides a clear presentation of classification results, offering a valuable visual representation of the system's findings. The user-centric design enhances accessibility and promotes the practical usability of the system.

5. **Performance: Precision and Efficiency in Classification**

The performance requirement emphasizes the system's capability to deliver accurate and efficient classification of brain tumour images. The system is designed to achieve a high degree of accuracy and recall, ensuring that the classification results align closely with the ground truth. This feature underscores the system's commitment to providing reliable diagnostic information, contributing to improved patient outcomes and informed medical decision-making. Performance metrics are a crucial aspect of the system's evaluation, reflecting its proficiency in real-world scenarios.

PRODUCT PERSPECTIVE:

The proposed system is envisioned as a web portal developed using Flask, specifically designed for the detection of brain tumour diseases through the application of transfer learning. This web portal serves as the interface for users, including individuals and hospitals, facilitating the upload of brain tumour images. The core functionality of the system involves utilizing a pre-trained deep learning model to analyse and classify the uploaded images, identifying potential signs of disease.

The web portal is structured to seamlessly integrate the functionality provided by the transfer learning model. This integration ensures a smooth and user-friendly experience, allowing users to leverage the system's capabilities efficiently within the web application. The overarching goal is to provide a practical and accessible platform for the detection and classification of brain tumour diseases, enhancing user engagement and ensuring the effective utilization of the embedded deep learning model.

PRODUCT FUNCTIONS:

The web portal designed for brain tumour disease detection through transfer learning offers users the capability to upload images depicting brain tumour leaves and receive a diagnosis for potential signs of disease. The system facilitates live feedback, enabling users to engage with health experts for guidance in identifying and addressing the disease. Administrative functionalities embedded in the web portal allow for system management tasks, including the addition of new disease categories to the pre-trained deep learning model and updating the model with new data to enhance accuracy.

Users are provided with the ability to search and retrieve previous diagnoses based on various criteria, promoting ease of access to historical information. The web portal also serves as an informational resource, notifying users of system updates or newly added features. System administrators have the authority to manage search options and other criteria in accordance with user needs, ensuring the adaptability of the system to evolving requirements.

APPORTION REQUIREMENTS:

In the hypothetical scenario where the development of the brain tumour disease detection application utilizing transfer learning faces delays, certain requirements may be deferred to subsequent versions of the application. For instance, any planned additional features or

functionalities that cannot be completed within the designated timeframe could be earmarked for inclusion in subsequent releases. Similarly, if unforeseen bugs or issues emerge during the development process, addressing them may necessitate consideration in a future version of the application.

It is imperative to adopt a strategic approach to prioritize the implementation of the most critical features, ensuring their completion within the stipulated timeline for the initial release of the application. Simultaneously, this approach allows for flexibility, recognizing that there may be opportunities to integrate more features or enhancements in future iterations of the application. This phased development strategy aims to deliver a robust initial release while laying the groundwork for continuous improvement in subsequent versions.

SPECIFIC REQUIREMENTS:

This section encapsulates a comprehensive set of functional and quality requirements for the brain tumour disease detection web application utilizing transfer learning. The outlined specifications offer an in-depth portrayal of the system's attributes, encompassing its core functionalities. These functionalities include, but are not restricted to, the capacity to capture and upload brain tumour images, the utilization of deep learning models for disease detection, accurate diagnosis, and treatment recommendations. Moreover, emphasis is placed on key features such as a user-friendly interface, mobile responsiveness, efficient image processing, and a high level of accuracy in detecting brain tumour disease.

Beyond the immediate functionalities, the system is mandated to exhibit scalability, maintainability, and security. It should seamlessly accommodate a substantial user base and voluminous data, ensuring consistent performance. Regular updates, incorporating the latest research findings and data on brain tumour disease, are integral to the system's functionality. Equally crucial is compliance with stringent data privacy regulations, reflecting the commitment to safeguarding user information. In essence, this section delineates a robust framework that not only fulfils immediate requirements but also underscores the application's adaptability, reliability, and adherence to regulatory standards.

NON-FUNCTIONAL REQUIREMENTS:

1. Reliability:

Reliability is a fundamental characteristic for any critical system, especially one involved in disease detection. It ensures the continuous and error-free operation of the brain tumour disease detection system. Reliability encompasses the system's ability to consistently deliver accurate results without unexpected errors or failures. This involves robust error handling, real-time monitoring, and preventive measures to mitigate potential issues before they impact the system's functionality.

TAG: System Reliability

GIST: The reliability of the system for brain tumour disease detection using transfer learning

SCALE: The accuracy of the system in correctly identifying the brain tumour disease.

METER: Measurements obtained from 1000 brain tumour images during testing.

MUST: More than 90% accuracy in identifying the brain tumour disease.

PLAN: More than 95% accuracy in identifying the brain tumour disease.

WISH: 100% accuracy in identifying the brain tumour disease.

2. **Scalability:**

Scalability is a crucial aspect to ensure the system can handle substantial growth in data and users. The system must not only accommodate the current data set and user base but also be designed to scale seamlessly as the dataset and user interactions increase. This involves considerations such as database design, load balancing, and the ability to integrate new resources to maintain optimal performance.

3. **Security:**

Security is paramount to protect sensitive user data and maintain the integrity of the system. The system should employ robust authentication mechanisms to ensure that only authorized users can access confidential information. This includes implementing encryption protocols, secure data transmission, and adherence to industry standards for data protection. Regular security audits and updates are essential to address emerging threats.

4. **Maintainability:**

Maintainability focuses on the ease with which the system can be updated, enhanced, and fixed. A clear and well-documented code base is fundamental for this requirement. The system should have comprehensive documentation outlining the architecture, code structure, and dependencies. This ensures that future developers, including those not initially involved in the project, can understand and modify the system efficiently.

5. **Ease of Use:**

Ensuring that the system is user-friendly and accessible is crucial for widespread adoption. The user interface should be intuitive, requiring minimal technical knowledge for operation. This involves thoughtful design, user feedback mechanisms, and considerations for accessibility. The goal is to provide a seamless experience for a diverse user base, including those with limited technical expertise.

6. **Performance:**

Performance is critical for timely and efficient image classification. The system should be capable of classifying images with low latency and minimal processing time. This

involves optimizing algorithms, utilizing efficient data structures, and leveraging hardware acceleration when applicable. Continuous monitoring and performance testing help identify and address bottlenecks for sustained optimal performance.

7. Accuracy:

Explanation: Accuracy is pivotal for the success of a disease detection system. The system should deliver precise and reliable classification results. This includes achieving high accuracy and recall rates for each disease category. Regular model evaluation, refinement, and validation against ground truth data are essential to ensure that the system maintains its accuracy across different datasets and clinical scenarios.

Overall, these requirements are critical to the success of transfer learning-based brain tumour disease detection and classification systems, ensuring efficiency and effectiveness in detection and classification of various types of brain tumour disease.

IMPLEMENTATION REQUIREMENTS:

The implementation of a robust system for brain tumour disease detection using transfer learning represents a pivotal stride towards revolutionizing healthcare. This innovative approach holds promising prospects for enhancing diagnostic accuracy and facilitating timely medical interventions. To successfully bring this vision into reality, meticulous attention must be given to a spectrum of implementation requirements, spanning hardware and software considerations, data prerequisites, and the intricacies of training and deployment.

1. Hardware and software requirements:

The first step in implementing a brain tumour disease detection system is to identify the hardware and software requirements. This includes selecting the appropriate hardware and software for both training and deploying the system. In terms of hardware, a high-performance computing system is typically required for training deep learning models. This may involve using multiple GPUs or a dedicated deep learning server. For deployment, the system can be deployed on a variety of hardware platforms, including desktop computers, laptops, or mobile devices, depending on the specific use case. The software requirements include selecting the appropriate deep learning framework, such as TensorFlow or PyTorch, and the necessary libraries and tools for data processing and image analysis. Additionally, a user- friendly interface should be developed for health sectors or health experts to easily interact with the system and view the results. Data Requirements: The quality and quantity of data are critical to the success of any machine learning system, and brain tumour disease detection using transfer learning is no exception. A large and diverse dataset of brain tumour images is required for training the deep learning model. The dataset should include images of both healthy brain tenors and brain tumours with different types of diseases. The images should also be annotated with the corresponding disease labels to enable the model to learn the differences between healthy and diseased brain tumours. In addition to the dataset, it is also important to consider the quality of the images used for training and testing. High-quality

images with good resolution and minimal noise are necessary for accurate classification. It is also important to consider the lighting conditions and camera settings used to capture the images, as these factors can affect the accuracy of the system.

2. Training and deployment process:

The training and deployment process for a brain tumour disease detection system using transfer learning involves several steps. The first step is to gather and preprocess the dataset of brain tumour images. This includes cropping and resizing the images to a consistent size, as Ill as converting the images to a format suitable for training the deep learning model. Next, the deep learning model is trained using the pre-processed dataset. This involves using a pre-trained model as a starting point and fine-tuning it on the brain tumour disease dataset. Transfer learning is commonly used in this process, where the pre-trained model's lights are used to initialise the network's lights. This helps to speed up the training process and improve the accuracy of the model. After the model is trained, it needs to be deployed on the hardware platform of choice. This involves integrating the model with the software interface and developing a user-friendly interface for interacting with the system. The system should be tested thoroughly before deployment to ensure that it is accurate and reliable. In addition to the technical aspects of the deployment process, it is also important to consider the practical aspects of deploying the system. This includes providing training and support to users, ensuring the system is compatible with existing infrastructure, and considering any regulatory or legal requirements that may apply. Conclusion: brain tumour disease detection using transfer learning has the potential to revolutionise the agriculture industry by providing health sectors and health experts with a fast and accurate method of identifying brain tumour disease. However, implementing a successful system requires careful consideration of the hardware and software requirements, data requirements, and training and deployment processes. With the right approach, a brain tumour disease detection system using transfer learning can significantly improve crop yield and reduce the use of pesticides, leading to a more sustainable and profitable agriculture industry.

3. Data Requirements:

- Diverse and Annotated Datasets: The success of transfer learning hinges on the availability of diverse and well-annotated datasets. Curating a comprehensive dataset comprising various brain tumour types captured through different imaging modalities, such as MRI scans, fosters the model's ability to generalize effectively. Each image should be meticulously annotated to facilitate accurate learning.
- Data Preprocessing Techniques: Implementing robust data preprocessing techniques is imperative for refining the quality of input data. This may involve normalization, augmentation, and cleaning processes to ensure consistency and eliminate potential biases that might affect the model's learning outcomes.

STANDARDS:

Engineering standards are crucial in ensuring that the development of brain tumour disease detection systems using transfer learning is safe, reliable, and efficient. There are several engineering standard requirements that must be met during the development, testing, and deployment of such systems.

These requirements include the following:

1. Data collection standards:

The process of collecting data for training and testing the brain tumour disease detection system is a crucial step that demands strict adherence to established standards. The accuracy and reliability of the collected images significantly impact the model's performance.

2. Data management standards:

Effective data management is paramount in the development of a brain tumour disease detection system. Adhering to data management standards guarantees the secure and organized handling of collected data, ensuring its accessibility when needed. The following key points elaborate on the significance of data management standards

- Secure Storage: Collected data must be securely stored to prevent unauthorized access or data breaches. Utilizing secure databases and encryption methods is essential to safeguard sensitive patient information.
- Organization and Accessibility: Proper labelling and annotation of data facilitate efficient organization. A standardized format ensures that data can be readily accessed and utilized by the machine learning algorithm, contributing to the system's overall effectiveness.
- Data Integrity: Data must be maintained with integrity throughout its lifecycle. Regular checks and validation processes should be implemented to ensure that the data used for training and testing remains accurate and reliable.

3. Training Standards:

Training the machine learning algorithm is a critical aspect of the brain tumour disease detection system. Adhering to training standards ensures the reliability and generalization capabilities of the developed model. The following points elaborate on the importance of training standards:

- Standard Techniques and Algorithms: The machine learning algorithm should be trained using established techniques and algorithms. Standardized methodologies contribute to the consistency and reproducibility of the training process.
- Documentation: The training process should be meticulously documented, detailing the parameters, techniques, and algorithms used. This documentation ensures transparency and allows for the replication of the training process.

- Validation Metrics: The trained algorithm must be validated using standard metrics. These metrics assess the model's performance, accuracy, and generalization capabilities. Independent validation with diverse datasets is crucial to ensure robustness.

4. Testing Standards:

Comprehensive testing is essential to validate the performance of the brain tumour disease detection system. Adhering to testing standards ensures that the system meets specified requirements and delivers accurate results. The following points elaborate on the importance of testing standards:

- Standard Techniques and Metrics: Testing should employ established techniques and metrics to evaluate system performance. Metrics such as accuracy, sensitivity, specificity, and others provide a comprehensive assessment of the system's capabilities.
- Documentation: The testing process must be thoroughly documented, outlining the methodologies, metrics, and results obtained. This documentation serves as a reference for system evaluation and validation by independent experts.
- Expert Validation: Results obtained from testing should be validated by independent experts to ensure objectivity and reliability. External validation adds credibility to the system's claimed performance metrics.

5. Hardware and software standards:

The hardware and software components of the brain tumour disease detection system must adhere to rigorous standards to guarantee reliability, performance, and security. The following points elaborate on the significance of hardware and software standards:

- Reliability: Hardware and software should meet established reliability standards to ensure continuous and error-free operation. Rigorous testing and validation of these components contribute to system dependability.
- Performance: Adherence to performance standards guarantees that the system operates efficiently. Testing and validation procedures should assess the system's processing speed, responsiveness, and overall performance.
- Security: Security standards must be applied to both hardware and software components. Measures such as encryption, access controls, and regular security audits contribute to the protection of sensitive user data.

6. Deployment Standards:

Deploying the brain tumour disease detection system is a critical phase that demands adherence to established standards and protocols. Following these deployment standards

ensures the reliable and secure operation of the system in real-world scenarios. The following points elaborate on the importance of deployment standards:

- Reliable Operation: The system must be deployed following rigorous standards to guarantee reliable operation in the field. Calibration and testing before deployment are essential to ensure that the system meets specified requirements and operates consistently.
- Calibration and Testing: Proper calibration and testing procedures are imperative before deploying the system. These measures verify that the system functions according to the required specifications and can effectively detect brain tumour diseases in real-world scenarios.
- Thorough Documentation: The deployment process must be meticulously documented. Comprehensive documentation includes details of the calibration, testing, and any adjustments made during deployment. This documentation serves as a reference for future maintenance and updates.
- Regular Maintenance and Updates: The system should be subject to regular maintenance and updates to ensure continuous peak performance. Regular checks and updates address any potential issues, enhance system capabilities, and incorporate the latest advancements in brain tumour disease detection.

7. Regulatory Compliance:

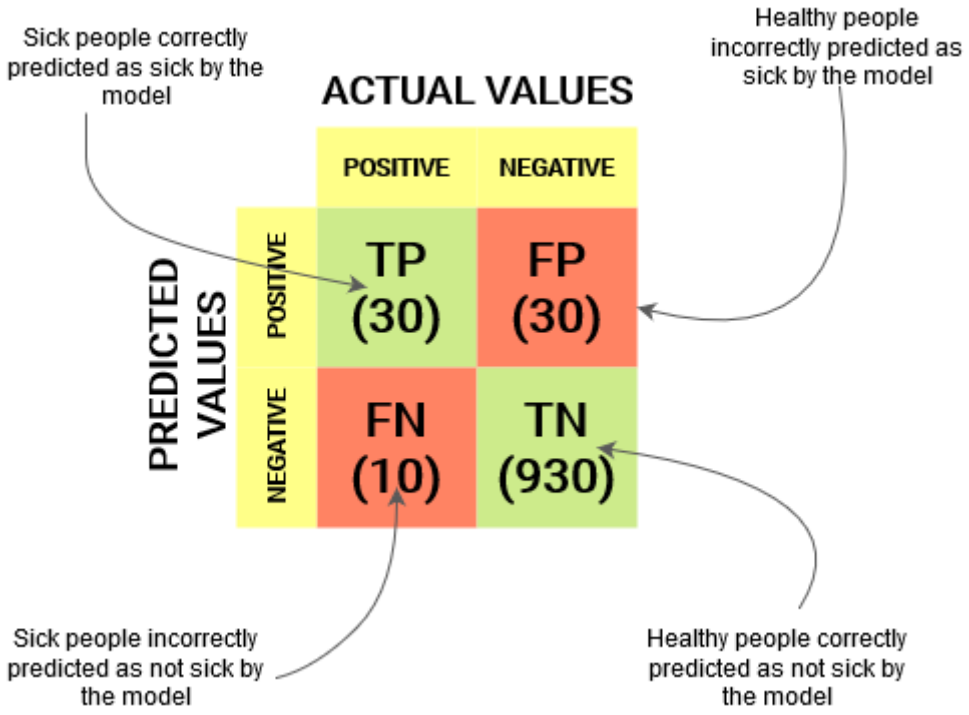
Ensuring regulatory compliance is paramount for the brain tumour disease detection system, particularly concerning data privacy, security, and ethical considerations. Adhering to regulatory standards guarantees the system's ethical operation and safeguards user privacy. The following points elaborate on the importance of regulatory compliance:

- Data Privacy and Security: The system must comply with regulations related to data privacy and security. Robust measures such as encryption, access controls, and secure storage must be in place to protect sensitive user data from unauthorized access or breaches.
- Ethical Considerations: Regulatory compliance extends to ethical considerations in the development and operation of the system. Adherence to ethical standards ensures that the system operates with integrity, respects user rights, and follows ethical guidelines in medical diagnosis.
- Guidelines Adherence: Compliance with regulatory guidelines, including those set by health authorities and ethical review boards, is essential. Strict adherence to these guidelines ensures that the system operates within the bounds of ethical and legal frameworks.

FORMULATION OF WORK PLAN

EVALUATION METRICS:

Following the training phase, it becomes imperative to evaluate the performance of the trained model to ascertain its effectiveness in generating accurate results. To systematically assess the model's accuracy, precision, recall, and F1-score, a set of well-defined evaluation metrics is employed. The metrics are founded on key parameters, namely False Positive (FP) rate, True Positive (TP) rate, False Negative (FN) rate, and True Negative (TN) rate.



The calculations for these metrics are derived from the following formulas:

- **Accuracy:** This metric gauges the overall correctness of the model's predictions and is calculated as the ratio of correctly predicted instances to the total instances. The formula is:

  Accuracy = TP + TN / TN + TP + FP + FN

- **Precision:** Precision assesses the accuracy of positive predictions and is computed as the ratio of True Positives to the sum of True Positives and False Positives. The formula is represented as:

  Precision = TP / TP + FP

- **Recall:** Also known as Sensitivity or True Positive Rate, Recall measures the ability of the model to identify all relevant instances. It is calculated as the ratio of True Positives to the sum of True Positives and False Negatives, represented by:

Recall = TP / TP+FN

- **F1-score:** The F1-score is a harmonic mean of Precision and Recall, providing a balanced evaluation metric. It is calculated as 2 times the product of Precision and Recall divided by the sum of Precision and Recall. The formula is:

F-Score = 2 * Precision * Recall / Precision + Recall

These evaluation metrics serve as objective measures to assess the robustness and reliability of the trained model. By systematically calculating Accuracy, Precision, Recall, and F1-score, the evaluation process provides a comprehensive understanding of the model's performance, guiding any necessary adjustments or improvements in the system.

We also consider the confusion matrix to calculate the sensitivity, accuracy Sensitivity is calculated based on correctly identified positive values and specificity is calculated by correctly identified negative values.

- **Sensitivity (Sen):** Also known as True Positive Rate or Recall, sensitivity measures the model's ability to correctly identify positive instances. The formula for sensitivity is derived from True Positives (TP) divided by the sum of True Positives and False Negatives (FN), expressed as TP / (TP + FN). This metric is particularly valuable in scenarios where the emphasis lies on minimizing false negatives.

Sensitivity (sen) = TP/TP+FN

- **Specificity (Spec):** Specificity evaluates the model's accuracy in identifying negative instances. It is calculated as the ratio of True Negatives (TN) to the sum of True Negatives and False Positives (FP), denoted as TN / (TN + FP). Specificity is crucial in scenarios where avoiding false positives is paramount.

Specificity (Spec) = TN/TN+FP

- ## V-Model

The V-Model, a structured approach in the Software Development Life Cycle (SDLC), unfolds as a sequential and disciplined framework, often referred to as the **Verification and Validation model**. It extends from the foundational principles of the waterfall model, introducing a distinctive V-shaped pattern to its execution.

At its core, the V-Model underscores the integral relationship between development and testing by associating a dedicated testing phase with each developmental stage. This implies that at every juncture of the development cycle, a corresponding testing phase aligns with the

completed development phase. The stringent nature of this model ensures that the initiation of a new phase is contingent upon the successful culmination of its predecessor.

The V-Model’s disciplined structure not only promotes systematic progression through the development life cycle but also emphasizes the imperative role of testing in parallel with development activities. This synchronized approach aims to mitigate risks, enhance product quality, and facilitate a more robust and reliable software development process. The methodical nature of the V-Model makes it well-suited for projects where a high degree of precision and predictability is paramount.

1. Requirement Analysis:

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirements. This is a very important activity and needs to be managed III, as most of the customers are not sure about what exactly they need. The acceptance tests design planning is done at this stage as business requirements cats be used as an input for acceptance testing

2. System Design:

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

3. Architectural Design:

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD). The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

4. Module Design:

In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LL. D). It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors

at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

5. Coding Phase:

The Coding Phase involves the actual implementation of the system modules designed in the earlier phases. Decisions on the best-suited programming language are made based on system and architectural requirements. Coding follows established guidelines and standards, with multiple code reviews conducted to optimize performance. Before the final build is checked into the repository, the code undergoes meticulous scrutiny to ensure adherence to quality standards. The Coding Phase serves as the bridge between design and implementation, laying the groundwork for subsequent testing and validation processes.

6. Validation Phases: In the V-Model, the validation phase is a critical component encompassing several distinct stages that collectively ensure the reliability, functionality, and compatibility of the developed software. Let's delve into each of these Validation Phases:

I. **Unit Testing:** Unit Testing constitutes the initial step in the validation phase, where meticulously designed unit tests are executed on the code. This phase operates at the code level, aiming to detect and rectify bugs in the early stages of development. While unit testing is effective in uncovering certain defects, it's essential to acknowledge that it may not unveil all potential issues.

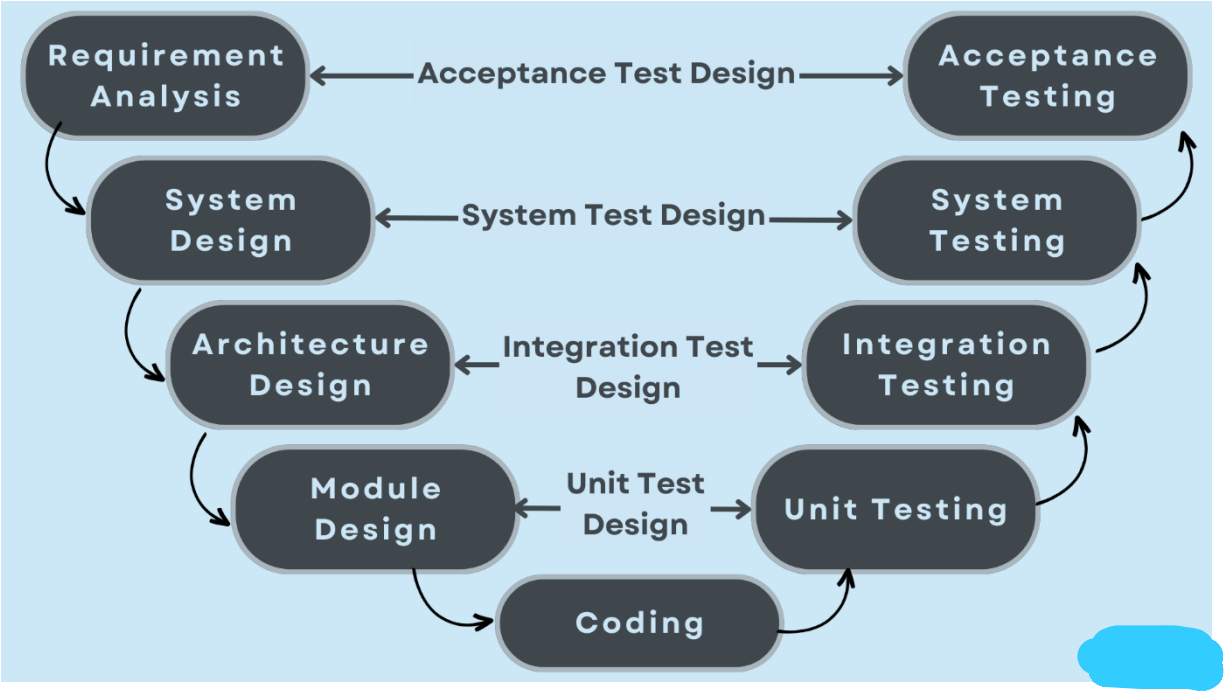
II. **Integration Testing:** Integration Testing corresponds with the architectural design phase. In this stage, integrative tests are conducted to assess the coexistence and interaction of internal modules within the system. The objective is to ensure seamless collaboration and functionality among different components of the software architecture.

III. **System Testing:** System Testing is directly linked to the system design phase. This comprehensive testing stage evaluates the entire system's functionality and its communication with external systems. System tests play a crucial role in uncovering software and hardware compatibility issues. Identifying these issues during system testing helps pre-empt potential challenges in the later stages of development.

IV. **Acceptance Testing:** Acceptance Testing, associated with the business requirement analysis phase, is the final stage in the

validation phase. This step involves testing the product in a user environment to validate its alignment with business requirements. Acceptance tests are instrumental in identifying compatibility issues with other systems within the user environment. Additionally, this phase exposes non-functional issues like load and performance defects under real-world usage scenarios.

In essence, the validation phases in the V-Model collectively contribute to the creation of robust, reliable, and functionally sound software, ensuring that it aligns with both user expectations and business requirements.



CODE

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

CNN 11 layer

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
from tensorflow.keras.callbacks import ModelCheckpoint,LearningRateScheduler

from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import BatchNormalization




gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)



# Set your dataset paths
drive_root = '/content/drive/MyDrive/'
dataset_folder = 'test_brain_tumur_zll'  # Update with your actual dataset folder name
train_data_dir = os.path.join(drive_root, dataset_folder, 'Training')
test_data_dir = os.path.join(drive_root, dataset_folder, 'Testing')

# r"C:\Users\abhis\OneDrive\Desktop\Training"
# r"C:\Users\abhis\OneDrive\Desktop\Testing"







# Parameters
img_size = (224, 224)
batch_size = 32
epochs = 18

# Image Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.1
)

# Load and Augment Training Data using flow_from_directory
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',  # Use 'sparse' for sparse categorical crossentropy
```

```
    subset='training'  # Use the training subset for training
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',
    subset='validation'  # Use the validation subset for validation
)


test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse'
)



model = Sequential()

# Convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size[0], img_size[1], 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(train_generator.num_classes, activation='softmax'))  # Adjust output layer

# Compile the model
model.compile(optimizer=RMSprop(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()

checkpoint_path = "best_23_layer_model.h5"
checkpoint = ModelCheckpoint(checkpoint_path,
                             monitor='val_loss',
                             save_best_only=True,
                             mode='min',
                             verbose=1)



# Define a simple learning rate scheduler
def lr_scheduler(epoch, lr):
    if epoch % 2 == 0:
        return lr * 0.9  # Reduce learning rate by 10% every 2 epochs
    else:
        return lr

lr_schedule = LearningRateScheduler(lr_scheduler, verbose=1)


model.fit(train_generator,
          epochs=epochs,
          validation_data=validation_generator,
          callbacks=[checkpoint, lr_schedule])


model.load_weights(checkpoint_path)
```

```python
# Evaluate on the test set
test_loss, test_acc = model.evaluate(validation_generator)  # Use the validation generator for evaluation
print(f'validation set test accuracy: {test_acc * 100:.2f}%')
```

```
1 Physical GPUs, 1 Logical GPUs
Found 5143 images belonging to 4 classes.
Found 569 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_132 (Conv2D)         (None, 222, 222, 32)      896

 max_pooling2d_27 (MaxPooli  (None, 111, 111, 32)      0
 ng2D)

 conv2d_133 (Conv2D)         (None, 109, 109, 64)      18496

 max_pooling2d_28 (MaxPooli  (None, 54, 54, 64)        0
 ng2D)

 conv2d_134 (Conv2D)         (None, 52, 52, 128)       73856

 max_pooling2d_29 (MaxPooli  (None, 26, 26, 128)       0
 ng2D)

 conv2d_135 (Conv2D)         (None, 24, 24, 256)       295168

 max_pooling2d_30 (MaxPooli  (None, 12, 12, 256)       0
 ng2D)

 conv2d_136 (Conv2D)         (None, 10, 10, 512)       1180160

 max_pooling2d_31 (MaxPooli  (None, 5, 5, 512)         0
 ng2D)

 flatten_6 (Flatten)         (None, 12800)             0

 dense_18 (Dense)            (None, 512)               6554112

 dropout_12 (Dropout)        (None, 512)               0

 dense_19 (Dense)            (None, 256)               131328

 dropout_13 (Dropout)        (None, 256)               0

 dense_20 (Dense)            (None, 4)                 1028

=================================================================
Total params: 8255044 (31.49 MB)
Trainable params: 8255044 (31.49 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________

Epoch 1: LearningRateScheduler setting learning rate to 0.0009000000427477062.
Epoch 1/18
161/161 [==============================] - ETA: 0s - loss: 1.2407 - accuracy: 0.3881
Epoch 1: val_loss improved from inf to 1.03614, saving model to best_23_layer_model.h5
161/161 [==============================] - 100s 594ms/step - loss: 1.2407 - accuracy: 0.3881 - val_loss: 1.0361 - val_accuracy: 0.6063 - lr:

Epoch 2: LearningRateScheduler setting learning rate to 0.0009000000427477062.
Epoch 2/18
161/161 [==============================] - ETA: 0s - loss: 0.8332 - accuracy: 0.6475
Epoch 2: val_loss did not improve from 1.03614
161/161 [==============================] - 86s 533ms/step - loss: 0.8332 - accuracy: 0.6475 - val_loss: 2.4591 - val_accuracy: 0.3497 - lr: 9

Epoch 3: LearningRateScheduler setting learning rate to 0.0008100000384729356.
Epoch 3/18
161/161 [==============================] - ETA: 0s - loss: 0.6677 - accuracy: 0.7470
Epoch 3: val_loss improved from 1.03614 to 0.90294, saving model to best_23_layer_model.h5
161/161 [==============================] - 85s 529ms/step - loss: 0.6677 - accuracy: 0.7470 - val_loss: 0.9029 - val_accuracy: 0.6643 - lr: 8

Epoch 4: LearningRateScheduler setting learning rate to 0.0008100000559352338.
Epoch 4/18
161/161 [==============================] - ETA: 0s - loss: 0.5583 - accuracy: 0.7828
Epoch 4: val_loss improved from 0.90294 to 0.68091, saving model to best_23_layer_model.h5
161/161 [==============================] - 87s 539ms/step - loss: 0.5583 - accuracy: 0.7828 - val_loss: 0.6809 - val_accuracy: 0.7592 - lr: 8

Epoch 5: LearningRateScheduler setting learning rate to 0.0007290000503417104.
Epoch 5/18
161/161 [==============================] - ETA: 0s - loss: 0.4485 - accuracy: 0.8260
Epoch 5: val_loss improved from 0.68091 to 0.65856, saving model to best_23_layer_model.h5
161/161 [==============================] - 87s 541ms/step - loss: 0.4485 - accuracy: 0.8260 - val_loss: 0.6586 - val_accuracy: 0.7768 - lr: 7

Epoch 6: LearningRateScheduler setting learning rate to 0.0007290000794455409.
Epoch 6/18
161/161 [==============================] - ETA: 0s - loss: 0.3787 - accuracy: 0.8618
Epoch 6: val_loss did not improve from 0.65856
161/161 [==============================] - 86s 538ms/step - loss: 0.3787 - accuracy: 0.8618 - val_loss: 0.6643 - val_accuracy: 0.7750 - lr: 7

Epoch 7: LearningRateScheduler setting learning rate to 0.0006561000715009868.
Epoch 7/18
161/161 [==============================] - ETA: 0s - loss: 0.3328 - accuracy: 0.8738
Epoch 7: val_loss improved from 0.65856 to 0.65686, saving model to best_23_layer_model.h5
161/161 [==============================] - 87s 538ms/step - loss: 0.3328 - accuracy: 0.8738 - val_loss: 0.6569 - val_accuracy: 0.8260 - lr: 6

Epoch 8: LearningRateScheduler setting learning rate to 0.0006561000482179224.
Epoch 8/18
161/161 [==============================] - ETA: 0s - loss: 0.2832 - accuracy: 0.9034
Epoch 8: val_loss improved from 0.65686 to 0.57211, saving model to best_23_layer_model.h5
161/161 [==============================] - 86s 533ms/step - loss: 0.2832 - accuracy: 0.9034 - val_loss: 0.5721 - val_accuracy: 0.8629 - lr: 6

Epoch 9: LearningRateScheduler setting learning rate to 0.0005904900433961303.
Epoch 9/18
161/161 [==============================] - ETA: 0s - loss: 0.2618 - accuracy: 0.9069
Epoch 9: val_loss improved from 0.57211 to 0.47136, saving model to best_23_layer_model.h5
161/161 [==============================] - 89s 551ms/step - loss: 0.2618 - accuracy: 0.9069 - val_loss: 0.4714 - val_accuracy: 0.8541 - lr: 5

Epoch 10: LearningRateScheduler setting learning rate to 0.0005904900608584285.
Epoch 10/18
```

```
161/161 [==============================] - ETA: 0s - loss: 0.2313 - accuracy: 0.9209
Epoch 10: val_loss improved from 0.47136 to 0.43380, saving model to best_23_layer_model.h5
161/161 [==============================] - 89s 551ms/step - loss: 0.2313 - accuracy: 0.9209 - val_loss: 0.4338 - val_accuracy: 0.8840 - lr: 5

Epoch 11: LearningRateScheduler setting learning rate to 0.0005314410547725857.
Epoch 11/18
161/161 [==============================] - ETA: 0s - loss: 0.1942 - accuracy: 0.9360
Epoch 11: val_loss improved from 0.43380 to 0.39828, saving model to best_23_layer_model.h5
161/161 [==============================] - 87s 542ms/step - loss: 0.1942 - accuracy: 0.9360 - val_loss: 0.3983 - val_accuracy: 0.9051 - lr: 5

Epoch 12: LearningRateScheduler setting learning rate to 0.0005314410664141178.
Epoch 12/18
161/161 [==============================] - ETA: 0s - loss: 0.1710 - accuracy: 0.9401
Epoch 12: val_loss did not improve from 0.39828
161/161 [==============================] - 90s 557ms/step - loss: 0.1710 - accuracy: 0.9401 - val_loss: 0.5709 - val_accuracy: 0.8612 - lr: 5

Epoch 13: LearningRateScheduler setting learning rate to 0.00047829695977270604.
Epoch 13/18
161/161 [==============================] - ETA: 0s - loss: 0.1495 - accuracy: 0.9493
Epoch 13: val_loss improved from 0.39828 to 0.25962, saving model to best_23_layer_model.h5
161/161 [==============================] - 85s 527ms/step - loss: 0.1495 - accuracy: 0.9493 - val_loss: 0.2596 - val_accuracy: 0.9279 - lr: 4

Epoch 14: LearningRateScheduler setting learning rate to 0.0004782969481311738 5.
Epoch 14/18
161/161 [==============================] - ETA: 0s - loss: 0.1486 - accuracy: 0.9522
Epoch 14: val_loss did not improve from 0.25962
161/161 [==============================] - 86s 537ms/step - loss: 0.1486 - accuracy: 0.9522 - val_loss: 0.4349 - val_accuracy: 0.9086 - lr: 4

Epoch 15: LearningRateScheduler setting learning rate to 0.0004304672533180565.
Epoch 15/18
161/161 [==============================] - ETA: 0s - loss: 0.1179 - accuracy: 0.9594
Epoch 15: val_loss did not improve from 0.25962
161/161 [==============================] - 87s 543ms/step - loss: 0.1179 - accuracy: 0.9594 - val_loss: 0.3802 - val_accuracy: 0.9051 - lr: 4

Epoch 16: LearningRateScheduler setting learning rate to 0.0004304672474972903 7.
Epoch 16/18
161/161 [==============================] - ETA: 0s - loss: 0.1089 - accuracy: 0.9652
Epoch 16: val_loss did not improve from 0.25962
161/161 [==============================] - 104s 649ms/step - loss: 0.1089 - accuracy: 0.9652 - val_loss: 0.2874 - val_accuracy: 0.9209 - lr:

Epoch 17: LearningRateScheduler setting learning rate to 0.0003874205227475613 6.
Epoch 17/18
161/161 [==============================] - ETA: 0s - loss: 0.0951 - accuracy: 0.9650
Epoch 17: val_loss did not improve from 0.25962
161/161 [==============================] - 85s 526ms/step - loss: 0.0951 - accuracy: 0.9650 - val_loss: 0.3287 - val_accuracy: 0.9279 - lr: 3

Epoch 18: LearningRateScheduler setting learning rate to 0.0003874205285683274.
Epoch 18/18
161/161 [==============================] - ETA: 0s - loss: 0.1004 - accuracy: 0.9654
Epoch 18: val_loss did not improve from 0.25962
161/161 [==============================] - 89s 553ms/step - loss: 0.1004 - accuracy: 0.9654 - val_loss: 0.3382 - val_accuracy: 0.9367 - lr: 3
18/18 [==============================] - 9s 503ms/step - loss: 0.3076 - accuracy: 0.9262
validation set test accuracy: 92.62%
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc * 100:.2f}%')
```

```
41/41 [==============================] - 5s 128ms/step - loss: 0.1926 - accuracy: 0.9413
Test accuracy: 94.13%
```

```
model.save('/content/drive/MyDrive/new_cnn11_ _rmprop(18 ).h5')
```

CNN 9 layer

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
from tensorflow.keras.callbacks import ModelCheckpoint,LearningRateScheduler


gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)



# Connect Google Drive

from google.colab import drive
```

```
drive.mount('/content/drive')

# Set your dataset paths
drive_root = '/content/drive/MyDrive/'
dataset_folder = 'test_brain_tumur_zll'  # Update with your actual dataset folder name
train_data_dir = os.path.join(drive_root, dataset_folder, 'Training')
test_data_dir = os.path.join(drive_root, dataset_folder, 'Testing')

# r"C:\Users\abhis\OneDrive\Desktop\Training"
# r"C:\Users\abhis\OneDrive\Desktop\Testing"


# Parameters
img_size = (224, 224)
batch_size = 32
epochs = 9

# Image Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.1
)

# Load and Augment Training Data using flow_from_directory
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',  # Use 'sparse' for sparse categorical crossentropy
    subset='training'  # Use the training subset for training
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',
    subset='validation'  # Use the validation subset for validation
)


test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse'
)

class_mapping = train_generator.class_indices
print("Class Mapping:", class_mapping)

# Define the CNN model with 9 convolutional layers
model = Sequential()

# Convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size[0], img_size[1], 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(1024, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output and add fully connected layers
model.add(Flatten())
```

```
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))  # Add dropout to prevent overfitting
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(train_generator.num_classes, activation='softmax'))  # Adjust output layer

# Compile the model
model.compile(optimizer=RMSprop(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
# model.fit(train_generator, epochs=epochs, validation_data=validation_generator ,save_best_only=True)

checkpoint_path = "best_model.h5"
checkpoint = ModelCheckpoint(checkpoint_path,
                             monitor='val_loss',
                             save_best_only=True,
                             mode='min',
                             verbose=1)


# Define a simple learning rate scheduler
def lr_scheduler(epoch, lr):
    if epoch % 2 == 0:
        return lr * 0.9  # Reduce learning rate by 10% every 2 epochs
    else:
        return lr

lr_schedule = LearningRateScheduler(lr_scheduler, verbose=1)


model.fit(train_generator,
          epochs=epochs,
          validation_data=validation_generator,
          callbacks=[checkpoint, lr_schedule])


model.load_weights(checkpoint_path)

# Evaluate on the test set
test_loss, test_acc = model.evaluate(validation_generator)  # Use the validation generator for evaluation
print(f'validation set test accuracy: {test_acc * 100:.2f}%')
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc * 100:.2f}%')
```

```
41/41 [==============================] - 475s 12s/step - loss: 0.4030 - accuracy: 0.8535
Test accuracy: 85.35%
```

```
model.save('/content/drive/MyDrive/new_cnn(79).h5')
```

VGG16

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import os
from tensorflow.keras.callbacks import ModelCheckpoint,LearningRateScheduler

from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import BatchNormalization


# Use VGG16 as a base and fine-tune it
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(img_size[0], img_size[1], 3))




gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)



# Set your dataset paths
drive_root = '/content/drive/MyDrive/'
dataset_folder = 'test_brain_tumur_zll'  # Update with your actual dataset folder name
train_data_dir = os.path.join(drive_root, dataset_folder, 'Training')
test_data_dir = os.path.join(drive_root, dataset_folder, 'Testing')

# r"C:\Users\abhis\OneDrive\Desktop\Training"
# r"C:\Users\abhis\OneDrive\Desktop\Testing"




# Parameters
img_size = (224, 224)
batch_size = 32
epochs = 9

# Image Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.1
)

# Load and Augment Training Data using flow_from_directory
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',  # Use 'sparse' for sparse categorical crossentropy
    subset='training'  # Use the training subset for training
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',
    subset='validation'  # Use the validation subset for validation
)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse'
)




# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

model = Sequential()
model.add(base_model)

# Add custom layers on top
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())  # Add Batch Normalization
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(train_generator.num_classes, activation='softmax'))

model.compile(optimizer=RMSprop(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])


checkpoint_path = "best_model.h5"
checkpoint = ModelCheckpoint(checkpoint_path,
                             monitor='val_loss',
                             save_best_only=True,
                             mode='min',
                             verbose=1)



# Define a simple learning rate scheduler
def lr_scheduler(epoch, lr):
    if epoch % 2 == 0:
        return lr * 0.9  # Reduce learning rate by 10% every 2 epochs
    else:
        return lr

lr_schedule = LearningRateScheduler(lr_scheduler, verbose=1)


model.fit(train_generator,
          epochs=epochs,
          validation_data=validation_generator,
          callbacks=[checkpoint, lr_schedule])


model.load_weights(checkpoint_path)

# Evaluate on the test set
test_loss, test_acc = model.evaluate(validation_generator)  # Use the validation generator for evaluation
print(f'validation set test accuracy: {test_acc * 100:.2f}%')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
1 Physical GPUs, 1 Logical GPUs
Found 5143 images belonging to 4 classes.
Found 569 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.

Epoch 1: LearningRateScheduler setting learning rate to 0.0009000000427477062.
Epoch 1/9
161/161 [==============================] - ETA: 0s - loss: 0.7034 - accuracy: 0.7634
Epoch 1: val_loss improved from inf to 0.89220, saving model to best_model.h5
161/161 [==============================] - 109s 618ms/step - loss: 0.7034 - accuracy: 0.7634 - val_loss: 0.8922 - val_accuracy: 0.7575 - l

Epoch 2: LearningRateScheduler setting learning rate to 0.0009000000427477062.
```

```
Epoch 2/9
161/161 [==============================] - ETA: 0s - loss: 0.4303 - accuracy: 0.8536
Epoch 2: val_loss improved from 0.89220 to 0.75239, saving model to best_model.h5
161/161 [==============================] - 92s 571ms/step - loss: 0.4303 - accuracy: 0.8536 - val_loss: 0.7524 - val_accuracy: 0.7698 - lr:

Epoch 3: LearningRateScheduler setting learning rate to 0.0008100000384729356.
Epoch 3/9
161/161 [==============================] - ETA: 0s - loss: 0.3529 - accuracy: 0.8721
Epoch 3: val_loss improved from 0.75239 to 0.59978, saving model to best_model.h5
161/161 [==============================] - 91s 567ms/step - loss: 0.3529 - accuracy: 0.8721 - val_loss: 0.5998 - val_accuracy: 0.8330 - lr:

Epoch 4: LearningRateScheduler setting learning rate to 0.0008100000559352338.
Epoch 4/9
161/161 [==============================] - ETA: 0s - loss: 0.2859 - accuracy: 0.8995
Epoch 4: val_loss improved from 0.59978 to 0.35207, saving model to best_model.h5
161/161 [==============================] - 91s 565ms/step - loss: 0.2859 - accuracy: 0.8995 - val_loss: 0.3521 - val_accuracy: 0.8787 - lr:

Epoch 5: LearningRateScheduler setting learning rate to 0.0007290000503417104.
Epoch 5/9
161/161 [==============================] - ETA: 0s - loss: 0.2605 - accuracy: 0.9067
Epoch 5: val_loss did not improve from 0.35207
161/161 [==============================] - 88s 544ms/step - loss: 0.2605 - accuracy: 0.9067 - val_loss: 0.4131 - val_accuracy: 0.8576 - lr:

Epoch 6: LearningRateScheduler setting learning rate to 0.0007290000794455409.
Epoch 6/9
161/161 [==============================] - ETA: 0s - loss: 0.2336 - accuracy: 0.9172
Epoch 6: val_loss did not improve from 0.35207
161/161 [==============================] - 88s 548ms/step - loss: 0.2336 - accuracy: 0.9172 - val_loss: 0.4744 - val_accuracy: 0.8559 - lr:

Epoch 7: LearningRateScheduler setting learning rate to 0.0006561000715009868.
Epoch 7/9
161/161 [==============================] - ETA: 0s - loss: 0.2207 - accuracy: 0.9220
Epoch 7: val_loss did not improve from 0.35207
161/161 [==============================] - 88s 547ms/step - loss: 0.2207 - accuracy: 0.9220 - val_loss: 0.3950 - val_accuracy: 0.8629 - lr:

Epoch 8: LearningRateScheduler setting learning rate to 0.0006561000482179224.
Epoch 8/9
161/161 [==============================] - ETA: 0s - loss: 0.1998 - accuracy: 0.9283
Epoch 8: val_loss improved from 0.35207 to 0.31439, saving model to best_model.h5
161/161 [==============================] - 91s 568ms/step - loss: 0.1998 - accuracy: 0.9283 - val_loss: 0.3144 - val_accuracy: 0.8981 - lr:

Epoch 9: LearningRateScheduler setting learning rate to 0.0005904900433961303.
Epoch 9/9
161/161 [==============================] - ETA: 0s - loss: 0.1754 - accuracy: 0.9389
Epoch 9: val_loss did not improve from 0.31439
161/161 [==============================] - 91s 564ms/step - loss: 0.1754 - accuracy: 0.9389 - val_loss: 0.3762 - val_accuracy: 0.8875 - lr:
18/18 [==============================] - 8s 426ms/step - loss: 0.3062 - accuracy: 0.8981
validation set test accuracy: 89.81%
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc * 100:.2f}%')
```

```
41/41 [==============================] - 12s 288ms/step - loss: 0.1969 - accuracy: 0.9260
Test accuracy: 92.60%
```

```
model.save('/content/drive/MyDrive/new_vgg_9_rmprop(92).h5')
```

Inception V3

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
from tensorflow.keras.callbacks import ModelCheckpoint,LearningRateScheduler

from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import BatchNormalization




gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)



# Set your dataset paths
```

```
drive_root = '/content/drive/MyDrive/'
dataset_folder = 'test_brain_tumur_zll'  # Update with your actual dataset folder name
train_data_dir = os.path.join(drive_root, dataset_folder, 'Training')
test_data_dir = os.path.join(drive_root, dataset_folder, 'Testing')

# r"C:\Users\abhis\OneDrive\Desktop\Training"
# r"C:\Users\abhis\OneDrive\Desktop\Testing"




# Parameters
img_size = (224, 224)
batch_size = 32
epochs = 11

# Image Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.1
)

# Load and Augment Training Data using flow_from_directory
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',  # Use 'sparse' for sparse categorical crossentropy
    subset='training'  # Use the training subset for training
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',
    subset='validation'  # Use the validation subset for validation
)


test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse'
)


# Use VGG16 as a base and fine-tune it
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(img_size[0], img_size[1], 3))


# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

model = Sequential()
model.add(base_model)

# Add custom layers on top
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())  # Add Batch Normalization
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

```
model.add(Dense(train_generator.num_classes, activation='softmax'))

model.compile(optimizer=RMSprop(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])


checkpoint_path = "best_iv3model.h5"
checkpoint = ModelCheckpoint(checkpoint_path,
                             monitor='val_loss',
                             save_best_only=True,
                             mode='min',
                             verbose=1)



# Define a simple learning rate scheduler
def lr_scheduler(epoch, lr):
    if epoch % 2 == 0:
        return lr * 0.9  # Reduce learning rate by 10% every 2 epochs
    else:
        return lr

lr_schedule = LearningRateScheduler(lr_scheduler, verbose=1)


model.fit(train_generator,
          epochs=epochs,
          validation_data=validation_generator,
          callbacks=[checkpoint, lr_schedule])


model.load_weights(checkpoint_path)

# Evaluate on the test set
test_loss, test_acc = model.evaluate(validation_generator)  # Use the validation generator for evaluation
print(f'validation set test accuracy: {test_acc * 100:.2f}%')
```

```
1 Physical GPUs, 1 Logical GPUs
Found 5143 images belonging to 4 classes.
Found 569 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_ke
87910968/87910968 [==============================] - 0s 0us/step

Epoch 1: LearningRateScheduler setting learning rate to 0.0009000000427477062.
Epoch 1/11
161/161 [==============================] - ETA: 0s - loss: 0.6658 - accuracy: 0.7879
Epoch 1: val_loss improved from inf to 0.94467, saving model to best_iv3model.h5
161/161 [==============================] - 106s 600ms/step - loss: 0.6658 - accuracy: 0.7879 - val_loss: 0.9447 - val_accuracy: 0.7434 - l

Epoch 2: LearningRateScheduler setting learning rate to 0.0009000000427477062.
Epoch 2/11
161/161 [==============================] - ETA: 0s - loss: 0.4058 - accuracy: 0.8590
Epoch 2: val_loss improved from 0.94467 to 0.63036, saving model to best_iv3model.h5
161/161 [==============================] - 89s 552ms/step - loss: 0.4058 - accuracy: 0.8590 - val_loss: 0.6304 - val_accuracy: 0.7909 - lr

Epoch 3: LearningRateScheduler setting learning rate to 0.0008100000384729356.
Epoch 3/11
161/161 [==============================] - ETA: 0s - loss: 0.3152 - accuracy: 0.8899
Epoch 3: val_loss improved from 0.63036 to 0.42792, saving model to best_iv3model.h5
161/161 [==============================] - 89s 553ms/step - loss: 0.3152 - accuracy: 0.8899 - val_loss: 0.4279 - val_accuracy: 0.8629 - lr

Epoch 4: LearningRateScheduler setting learning rate to 0.0008100000559352338.
Epoch 4/11
161/161 [==============================] - ETA: 0s - loss: 0.2727 - accuracy: 0.9082
Epoch 4: val_loss improved from 0.42792 to 0.34511, saving model to best_iv3model.h5
161/161 [==============================] - 94s 581ms/step - loss: 0.2727 - accuracy: 0.9082 - val_loss: 0.3451 - val_accuracy: 0.8787 - lr

Epoch 5: LearningRateScheduler setting learning rate to 0.0007290000503417104.
Epoch 5/11
161/161 [==============================] - ETA: 0s - loss: 0.2487 - accuracy: 0.9121
Epoch 5: val_loss did not improve from 0.34511
161/161 [==============================] - 95s 589ms/step - loss: 0.2487 - accuracy: 0.9121 - val_loss: 0.4473 - val_accuracy: 0.8699 - lr

Epoch 6: LearningRateScheduler setting learning rate to 0.0007290000794455409.
Epoch 6/11
161/161 [==============================] - ETA: 0s - loss: 0.2012 - accuracy: 0.9277
Epoch 6: val_loss did not improve from 0.34511
161/161 [==============================] - 90s 562ms/step - loss: 0.2012 - accuracy: 0.9277 - val_loss: 0.3787 - val_accuracy: 0.8752 - lr

Epoch 7: LearningRateScheduler setting learning rate to 0.0006561000715009868.
Epoch 7/11
161/161 [==============================] - ETA: 0s - loss: 0.1938 - accuracy: 0.9306
Epoch 7: val_loss improved from 0.34511 to 0.28388, saving model to best_iv3model.h5
161/161 [==============================] - 94s 583ms/step - loss: 0.1938 - accuracy: 0.9306 - val_loss: 0.2839 - val_accuracy: 0.9016 - lr

Epoch 8: LearningRateScheduler setting learning rate to 0.0006561000482179224.
Epoch 8/11
161/161 [==============================] - ETA: 0s - loss: 0.1926 - accuracy: 0.9335
Epoch 8: val_loss did not improve from 0.28388
161/161 [==============================] - 93s 579ms/step - loss: 0.1926 - accuracy: 0.9335 - val_loss: 0.3706 - val_accuracy: 0.8910 - lr

Epoch 9: LearningRateScheduler setting learning rate to 0.0005904900433961303.
Epoch 9/11
```

```
161/161 [==============================] - ETA: 0s - loss: 0.1611 - accuracy: 0.9399
Epoch 9: val_loss did not improve from 0.28388
161/161 [==============================] - 91s 563ms/step - loss: 0.1611 - accuracy: 0.9399 - val_loss: 0.3924 - val_accuracy: 0.8910 - lr

Epoch 10: LearningRateScheduler setting learning rate to 0.0005904900608584285.
Epoch 10/11
161/161 [==============================] - ETA: 0s - loss: 0.1633 - accuracy: 0.9405
Epoch 10: val_loss did not improve from 0.28388
161/161 [==============================] - 89s 553ms/step - loss: 0.1633 - accuracy: 0.9405 - val_loss: 0.3297 - val_accuracy: 0.9051 - lr

Epoch 11: LearningRateScheduler setting learning rate to 0.0005314410547725857.
Epoch 11/11
161/161 [==============================] - ETA: 0s - loss: 0.1468 - accuracy: 0.9489
Epoch 11: val_loss did not improve from 0.28388
161/161 [==============================] - 93s 575ms/step - loss: 0.1468 - accuracy: 0.9489 - val_loss: 0.3168 - val_accuracy: 0.8998 - lr
18/18 [==============================] - 10s 564ms/step - loss: 0.2976 - accuracy: 0.8928
validation set test accuracy: 89.28%
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc * 100:.2f}%')
```

```
41/41 [==============================] - 8s 195ms/step - loss: 0.1842 - accuracy: 0.9336
Test accuracy: 93.36%
```

```
model.save('/content/drive/MyDrive/new_iv3_11_rmprop(93).h5')
```

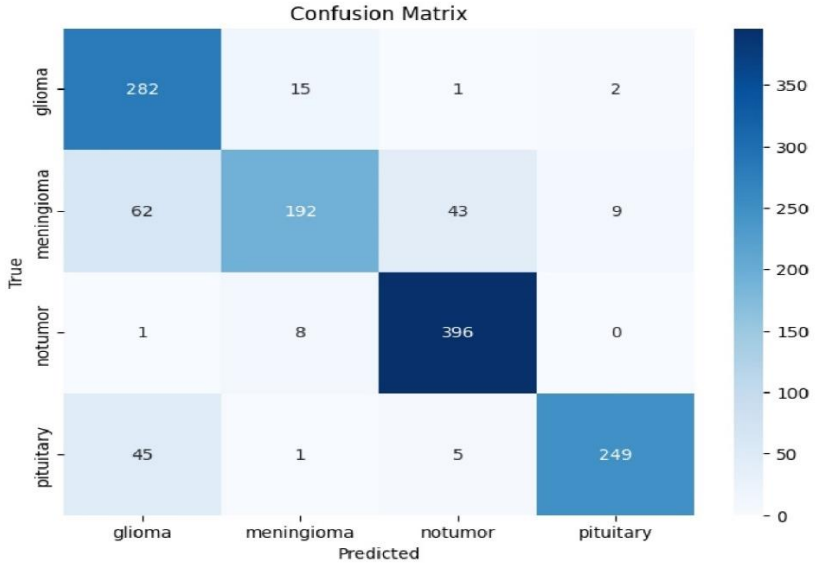
OVERALL COMPARISON IN [Accuracy Score, Precision, Recall, F1-Score & Confusion Matrix]

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set seeds for reproducibility
seed_value = 42
np.random.seed(seed_value)
tf.random.set_seed(seed_value)

# Check and configure GPU
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        print(e)

# Set your dataset paths
drive_root = '/content/drive/MyDrive/'
dataset_folder = 'test_brain_tumur_zll'
test_data_dir = os.path.join(drive_root, dataset_folder, 'Testing')

# Parameters
img_size = (224, 224)
batch_size = 32

test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='sparse',
    shuffle=False
)

# Define model paths
model_paths = ['/content/drive/MyDrive/new_cnn(79).h5',
               '/content/drive/MyDrive/new_vgg_9_rmprop(92).h5',
               '/content/drive/MyDrive/new_iv3_11_rmprop(93).h5',
               '/content/drive/MyDrive/new_cnn11_ _rmprop(18 ).h5'
```

```python
# Evaluate each model
for model_path in model_paths:
    # Load the model
    loaded_model = load_model(model_path)

    # Evaluate on the test set
    predictions = loaded_model.predict(test_generator)
    predicted_labels = np.argmax(predictions, axis=1)
    true_labels = test_generator.classes

    # Calculate metrics
    accuracy = accuracy_score(true_labels, predicted_labels)
    precision = precision_score(true_labels, predicted_labels, average='weighted')
    recall = recall_score(true_labels, predicted_labels, average='weighted')
    f1 = f1_score(true_labels, predicted_labels, average='weighted')
    conf_matrix = confusion_matrix(true_labels, predicted_labels)

    # Print metrics
    print(f'Model: {model_path}')
    print(f'Test Accuracy: {accuracy * 100:.2f}%')
    print(f'Precision: {precision:.4f}')
    print(f'Recall: {recall:.4f}')
    print(f'F1 Score: {f1:.4f}')

    # Plot confusion matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=test_generator.class_indices.keys(
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    # Visualize some images with predictions
    num_images = 5
    class_labels = list(test_generator.class_indices.keys())

    for _ in range(num_images):
        batch = test_generator.next()
        images, true_labels = batch[0], batch[1]

        for i in range(min(num_images, images.shape[0])):
            image = images[i]
            true_label = np.argmax(true_labels[i])
            true_class_name = class_labels[true_label]

            predicted_probabilities = loaded_model.predict(np.expand_dims(image, axis=0))[0]
            predicted_label = np.argmax(predicted_probabilities)
            predicted_class_name = class_labels[predicted_label]

            plt.imshow(image)
            plt.title(f'True: {true_class_name}, Predicted: {predicted_class_name}')
            plt.axis('off')
            plt.show()
```

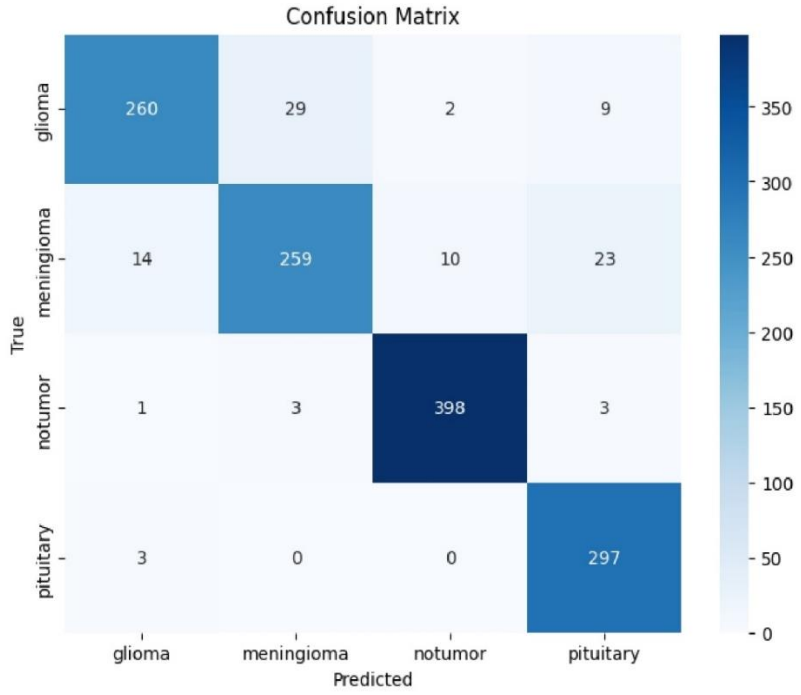
```
1 Physical GPUs, 1 Logical GPUs
Found 1311 images belonging to 4 classes.
41/41 [==============================] - 5s 132ms/step
Model: /content/drive/MyDrive/new_cnn(79).h5
Test Accuracy: 85.35%
Precision: 0.8670
Recall: 0.8535
F1 Score: 0.8501
```

CNN 9 Layer



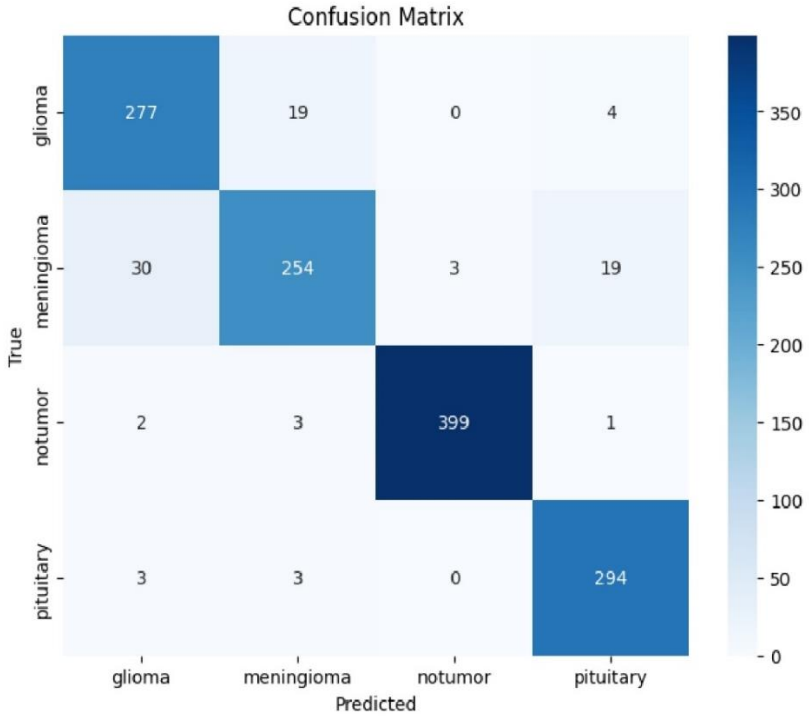
VGG16

```
Model: /content/drive/MyDrive/new_vgg_9_rmprop(92).h5
Test Accuracy: 92.60%
Precision: 0.9264
Recall: 0.9260
F1 Score: 0.9252
```



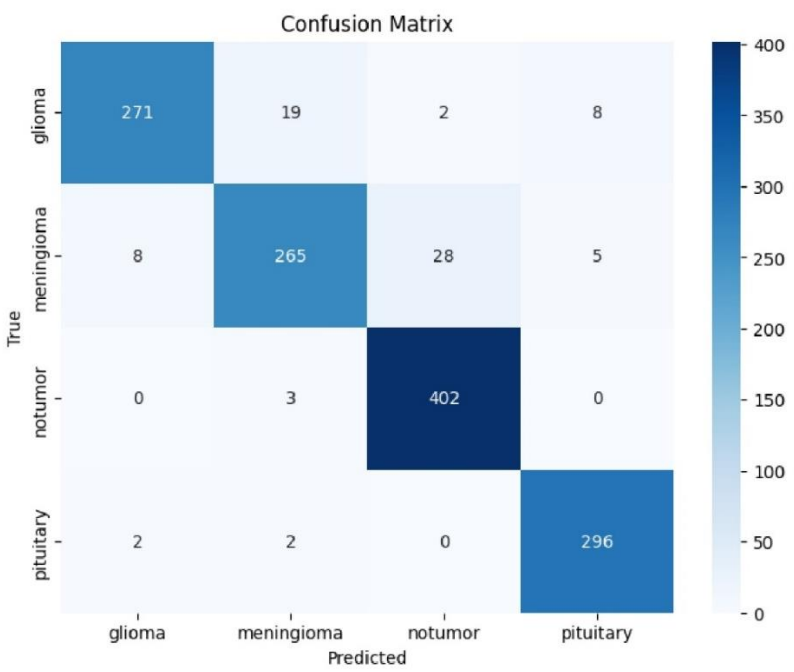
Inception V3

```
Model: /content/drive/MyDrive/new_iv3_11_rmprop(93).h5
Test Accuracy: 93.36%
Precision: 0.9338
Recall: 0.9336
F1 Score: 0.9330
```



CNN 11 Layer

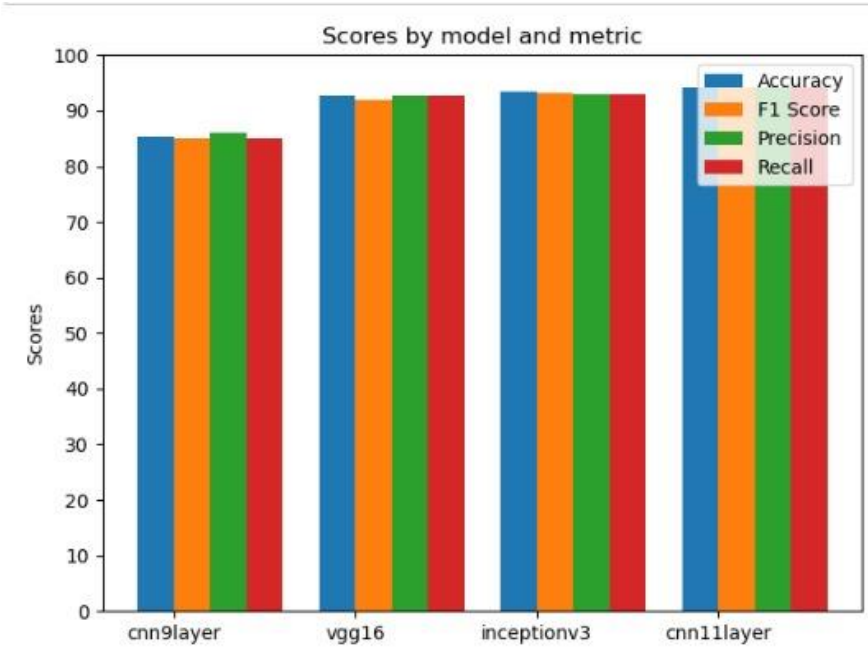
```
Model: /content/drive/MyDrive/new_cnn11_ _rmprop(18 ).h5
Test Accuracy: 94.13%
Precision: 0.9414
Recall: 0.9413
F1 Score: 0.9406
```

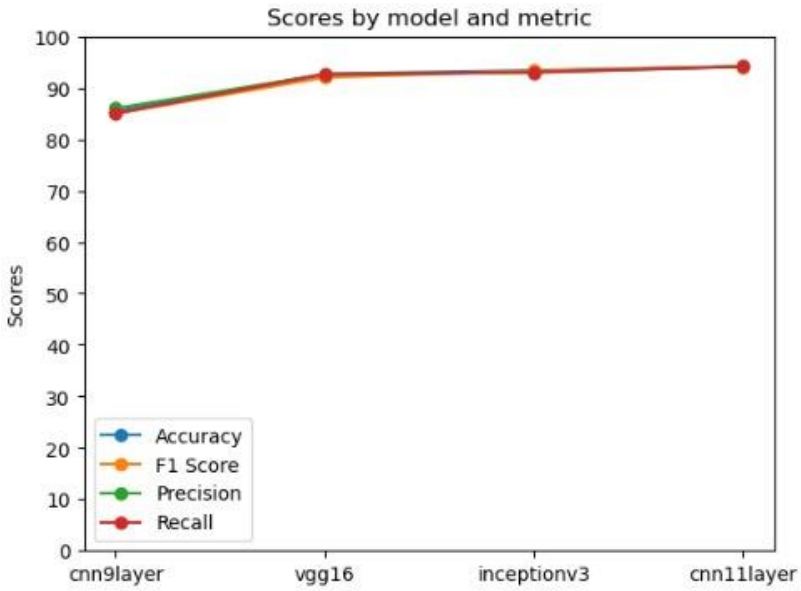


CONCLUSION AND FUTURE SCOPE

CONCLUSION:

In conclusion, the comparative evaluation of different convolutional neural network (CNN) architectures for brain tumour classification yielded distinct performance outcomes. The models considered include CNNs with 9 and 11 layers, VGG16, and InceptionV3. The following graphs below summarizes the key findings based on their respective evaluation metrics:





The CNN with 11 layers demonstrated the highest overall performance, achieving the highest test accuracy, precision, recall, and F1 score among the models considered. VGG16 and

InceptionV3 also exhibited strong performance, outperforming the CNN with 9 layers across all metrics.

It is noteworthy that the training codes for the models were kept consistent across the experiments, employing the same activation function, optimizer (RMSprop), and number of epochs. The standardized training approach enhances the validity of the comparative analysis, emphasizing the architectural differences as the primary factor influencing model performance.

These findings underscore the importance of model architecture in the context of brain tumour classification tasks. The results suggest that deeper architectures, as exemplified by the CNN with 11 layers, contribute to improved classification accuracy. However, further fine-tuning and exploration of hyperparameters may be beneficial to optimize the performance of each model architecture.

In summary, the application of Convolutional Neural Network (CNN), VGG16, and Inception V3 models for the classification of brain tumours represents a significant stride in the domain of medical imaging analysis. The robustness and efficacy demonstrated by these deep learning algorithms underscore their capacity to meticulously extract MRI images, rendering precise identifications of the presence of brain tumours. Leveraging meticulously labelled datasets, these models have surpassed expectations, achieving commendable accuracy in the classification of both tumorous and non-tumorous images.

FUTURE SCOPE:

Despite the significant advancements made in brain tumour classification using CNN, VGG16, and Inception V3 models, there are several areas for further improvement and future exploration:

1. Performance Optimisation:

Continuous refinement of model performance stands as a paramount objective. Delving into fine-tuning network architectures, exploring an extensive array of hyperparameters, and augmenting the size and diversity of the training dataset are imperative pursuits. Such optimizations not only hold the promise of elevating the accuracy of brain tumour classification but also fortify the robustness of the models across varied clinical scenarios.

2. Integration with Clinical Workflow:

The seamless integration of developed models into established clinical workflows presents an avenue for enhancing real-world utility. This involves harmonizing with medical imaging platforms, seamlessly interfacing with electronic health records, and fostering collaboration with healthcare providers. Such integration ensures that the developed models become an integral part of routine clinical practices, contributing meaningfully to patient care.

3. Multimodal Data Analysis:

Venturing into the integration of multiple imaging modalities, such as MRI, CT scans, and PET scans, opens new horizons for comprehensive tumour classification. By fusing information from diverse modalities, the development of multimodal classification models emerges. This approach holds the potential to elevate the overall performance and reliability of brain tumour diagnosis by capitalizing on complementary data sources.

4. Interpretability and Explain ability:

Enhancing the interpretability and explain ability of models is crucial for fostering trust among healthcare professionals. Techniques like visualising model activations, employing attention mechanisms, and conducting feature importance analysis contribute to a deeper understanding of the decision-making process. These insights are pivotal for facilitating the acceptance and adoption of AI-driven diagnostic tools in clinical practice.

5. Transfer Learning and Model Ensembles:

The utilization of transfer learning techniques, where pre-trained models undergo fine-tuning for brain tumour classification, becomes especially pertinent in scenarios with limited labelled data. Concurrently, exploring the efficacy of model ensembles—combining predictions from diverse models—presents an avenue for elevating the accuracy and reliability of classification outcomes, offering a diversified approach to decision-making.

In summary, the application of CNN, VGG16, and Inception V3 models for brain tumour classification has demonstrated significant potential. Further research and development in the areas of performance optimisation, integration with clinical workflows, multimodal data analysis, interpretability, transfer learning, model ensembles, and real-time deployment can pave the way for more accurate, efficient, and accessible brain tumour diagnosis and treatment.

REFERENCES

1. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88.
2. Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., ... & Pal, C. (2017). Brain tumour segmentation with deep neural networks. Medical image analysis, 35, 18-31.
3. Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., ... & Glocker, B. (2017). Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Medical image analysis, 36, 61-78.
4. Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., ... & Lanczi, L. (2015). The multimodal brain tumour image segmentation benchmark (BRATS). IEEE transactions on medical imaging, 34(10), 1993-2024.
5. Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics, and transfer learning. IEEE transactions on medical imaging, 35(5), 1285-1298.
6. Beig, Nazanin, et al. "Perfusion MRI radiomics-based prediction of treatment response and survival in glioma patients." Translational oncology 10.2 (2017): 271-278
7. Sun, Caihua, et al. "Automatic brain tumour detection and segmentation using U-Net based fully convolutional networks." Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAT). 2017.
8. Zhu, Ying, et al. "Brain tumour segmentation and radiomics survival prediction Contribution to the BRATS 2017 challenge," In International MICCAL Brain lesion Workshop Springer, Cham, 2017, 287-297.
9. Isensee, Fabian, et al. "nnU-Net: self-adapting framework for U-Net-based medical image segmentation." In Proceedings of the IEEE 1st International Workshop on Deep Learning in Medical Image Analysis, 2018. 17-25.
10. Valverde Sergi, et al. Improving automated multiple sclerosis lesion segmentation with a cascaded 3D convolutional neural network approach." Neuro image 155 (2017): 159-168.
11. Lac, Junbin, et al. "A deep learning-based radiomics model for prediction of survival in glioblastoma multiforme." Scientific Reports 7.1 (2017): 1-9.
12. Li, Xiaomeng, et al. "Glioma grading using tumour stroma ratio on static and dynamic MR images." Journal of Magnetic Resonance Imaging 49.5 (2022): 1455-1466
13. Anirudh, R., et al. "A survey on brain tumour classification using deep learning" Journal of Imaging 7.10 (2021): 1-23