BRAIN TUMOR DETECTION USING VGG16 AND XAI

A project report submitted in partial fulfilment of the requirements for the award of the degree of

MASTERS OF COMPUTER SCIENCE

By

SREERAG S **(REG NO. DVAXMCS011)**

Under the Guidance of

Ms. LEESHMA K

DEPARTMENT OF COMPUTER SCIENCE

ST. JOSEPH'S COLLEGE (AUTONOMOUS), DEVAGIRI, CALICUT

March 2025

ST. JOSEPH’S COLLEGE (AUTONOMOUS)
DEVAGIRI, CALICUT

BRAIN TUMOR DETECTION USING VGG16 AND XAI

Report of the project submitted to St. Joseph’s College (Autonomous) Devagiri; Calicut
Affiliated to the University of Calicut

SREERAG S **(REG NO. DVAXMCS011)**

Programme: Master of Computer Science

Semester: IV

DEPARTMENT OF COMPUTER SCIENCE

March 2025

Principal Head of the Department Supervisor

College seal

CERTIFICATE

This is to certify that the project report titled

BRAIN TUMOR DETECTION USING VGG16 AND XAI

Submitted by

SREERAG S **(REG NO. DVAXMCS011)**

St. Joseph's College (Autonomous), Devagiri, Calicut in partial fulfillment of the requirements for the award of degree of Master of Computer Science during the year 2023-2025.

Internal Examiner External Examiner

DECLARATION

I hereby declare that the project entitled **Brain tumor detection using VGG16 and Xai** has been undertaken by me for the award of the degree of Master of Computer Science. I have completed this project under the guidance of Ms. Leeshma K, Department of Computer Science St. Joseph's College (Autonomous), Devagiri, Calicut.

Place: Calicut

Date:

SREERAG S **(REG NO. DVAXMCS011)**

CERTIFICATE

This is to certify that the project report titled **Brain tumor Detection using VGG16 and XAI** record of project work done by him during the academic year 2024 – 2025 under my guidance and supervision in partial fulfilment of the requirement of degree of Master of Computer Science.

Place: Devagiri

Date: Ms. Leeshma K

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Dr. Boby Jose, Principal of St. Joseph's College (Autonomous), Devagiri, Calicut for providing necessary facilities for developing my project.

I am also grateful to Dr. Asha Unnikrishnan, Head of the Department as well as my internal guide Ms. Leeshma K, for her immense support and valuable suggestions.

SREERAG S **(REG NO. DVAXMCS011)**

ABSTRACT

ABSTRACT

Brain tumors are among the most life-threatening forms of cancer, and their early and accurate diagnosis is crucial for determining the appropriate treatment strategy and improving patient outcomes. Magnetic Resonance Imaging (MRI) is widely used as a non-invasive technique for brain tumor detection due to its high-resolution imaging capabilities. However, manual examination of MRI scans by radiologists can be time-consuming, subjective, and prone to human error. To address this challenge, this project explores an automated deep learning-based system for brain tumor classification that leverages both powerful neural architectures and explainable artificial intelligence (XAI) techniques.

In this study, a Convolutional Neural Network (CNN) based on the pre-trained VGG16 model is employed to classify MRI brain images into tumor and non-tumor categories. The dataset used consists of labelled MRI images collected from a public repository, encompassing multiple tumor types including glioma, meningioma, and pituitary tumors. To enhance model generalization and performance, the dataset is pre-processed and augmented using various image transformation techniques. The VGG16 model is fine-tuned on this dataset using transfer learning, allowing the system to learn domain-specific features with high accuracy.

Furthermore, to build trust in the model's predictions, Local Interpretable Model-agnostic Explanations (LIME) is incorporated as an explainability layer. LIME generates heatmaps highlighting the most influential regions of the input image that led to a particular classification, providing transparency and interpretability for medical professionals. This helps bridge the gap between artificial intelligence and clinical decision-making, ensuring that the predictions made by the model are not treated as black-box outputs.

Comprehensive evaluation metrics such as accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC are used to assess the model's effectiveness. The results demonstrate that the proposed system achieves a high level of classification performance while offering explainable outputs.

TABLE OF CONTENTS

	Page No
CERTIFICATE 1	II
CERTIFICATE 2	III
DECLARATION	IV
ACKNOWLEDGEMENT	VI
ABSTRACT	VII
TABLE OF CONTENTS	IX
CHAPTER I **INTRODUCTION**	1
CHAPTER II **LITERATURE SURVEY**	4
CHAPTER III **PROPOSED SYSTEM AND METHODOLOGY**	
3.1 PROPOSED SYSTEM	9
3.2 METHODOLOGY	11

CHAPTER IV

TRAINING, TESTING AND VALIDATION 20

CHAPTER V

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION 24

5.2 RESULTS 28

CHAPTER VI

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONLCUSION 32

6.2 FUTURE ENHANCEMENT 32

REFERENCES 34

APPENDIX A: RESULT AND SCREENSHOT 37

APPENDIX B: ABBREVIATIONS 39

CHAPTER I

INTRODUCTION

CHAPTER 1
INTRODUCTION

Brain tumors represent one of the most dangerous and life-altering forms of neurological disorders, with an increasing prevalence across the globe. Early and accurate detection of brain tumors plays a crucial role in improving survival rates, optimizing treatment planning, and enhancing patient outcomes. Traditionally, diagnosis relies on radiologists interpreting Magnetic Resonance Imaging (MRI) scans and performing biopsies. While effective, these methods can be time-intensive, subjective, and prone to interobserver variability, making them less reliable in high-pressure clinical environments.

The emergence of Artificial Intelligence (AI) and Machine Learning (ML) has significantly transformed medical imaging and diagnostics by enabling automated, faster, and more accurate analysis. In particular, deep learning—and specifically Convolutional Neural Networks (CNNs)—has demonstrated exceptional performance in classifying and localizing brain tumors in MRI scans, often achieving diagnostic accuracy comparable to that of experienced radiologists.

This project investigates cutting-edge AI techniques in the domain of brain tumor detection and classification. It centres around the application of VGG16, a popular pre-trained CNN architecture, fine-tuned for classifying brain MRIs into tumor and non-tumor categories. The project utilizes a publicly available dataset containing a diverse collection of brain MRI images labelled with glioma, meningioma, pituitary, and no tumor classes. The dataset, though powerful, poses challenges such as class imbalance, varying image quality, and limited representation of rare tumor types—all of which affect model generalization.

Deep learning architectures such as VGG, ResNet, and Inception have become foundational in medical image classification tasks. In this project, VGG16 is chosen for its structured design and deep convolutional layers, which enhance feature extraction capabilities from MRI scans. The model is trained using transfer learning, allowing it to adapt its learned representations from large-scale image datasets to the specific domain of brain tumor images.

To bridge the gap between AI and clinical trust, Explainable AI (XAI) is integrated into the system using Local Interpretable Model-agnostic Explanations (LIME). LIME helps visualize the decision-making process of the CNN model by highlighting regions in the MRI scan that contributed most to its predictions. This interpretability not only increases clinician trust but also aids in validating the reliability of the model in critical healthcare settings.

While the VGG16-based model demonstrates high classification performance, challenges such as data imbalance, the complexity of tumor boundaries, and explainability of deep models still persist. Moreover, generalizing the model across diverse patient demographics and imaging modalities requires extensive validation. This highlights the need for future research involving multi-canter datasets, robust augmentation techniques, and clinically validated explainability frameworks.

Future directions may include the incorporation of multimodal data (such as clinical records and histopathological results), integration of real-time model updates in hospital systems, and the deployment of mobile or cloud-based solutions to assist radiologists in remote areas. Furthermore, the evolution of explainable AI and its synergy with powerful CNN architectures may pave the way for more transparent and effective computer-aided diagnostic systems.

By exploring these advancements, this project emphasizes the potential of AI to revolutionize brain tumor diagnosis while underscoring the importance of model interpretability, clinical relevance, and real-world applicability in healthcare environments.

CHAPTER II

LITERATURE SURVEY

CHAPTER II
LITERATURE SURVEY

Khan, M.A. et al (2022). [1] Explainable Deep Learning for Brain Tumor Classification Using VGG16 and LIME. This study proposed a deep learning method integrating VGG16 with LIME for multi-class classification of brain tumors from contrast-enhanced MRI images. The dataset included glioma, meningioma, pituitary tumor, and no tumor classes. Transfer learning was applied with VGG16, and data augmentation techniques like flipping and rotation enhanced model generalization. LIME explained predictions by identifying the most influential image regions. The model achieved an accuracy of 95.2%, and LIME's visualizations improved interpretability and trust in AI-assisted medical diagnosis.

Dhakshnamurthy, V. K et al [2] Brain Tumor Detection and Classification using VGG-16 and Explainable AI (XAI) Techniques. This work employed a pre-trained VGG-16 model fine-tuned with MRI scans for brain tumor classification into four classes. Resizing, normalization, and data augmentation improved performance. LIME was used to visualize and interpret predictions by highlighting key image regions. The model achieved an accuracy of 94.6%, and the integration of LIME reinforced model transparency, aiding clinical reliability and decision-making.

Yan, F. et al (2023). [3] XAI-Framework: Explainable AI for Deep Learning-Based Brain Tumor Detection and Classification. This review assessed deep learning approaches for brain tumor detection, focusing on the gap in interpretability. While CNNs and transfer learning showed strong classification accuracy, most models lacked transparency. XAI techniques such as LIME, Grad-CAM, and SHAP were discussed as tools to enhance interpretability. The paper emphasized the need for a unified framework that balances prediction performance with explainability to support clinical decision-making. Reported average model accuracy across studies was 92.8%

Liu T et al (n.d.). [4] Hybrid Deep Learning for Brain Tumor Classification. This study proposed a hybrid model combining ResNet152 with LSTM to extract both spatial and temporal features from MRI images. The model achieved an accuracy of 99.1%. Despite its high performance, the model lacked explainability, underscoring the need for integrating XAI in medical contexts.

Afshar P et al (2019). [5] Brain Tumor Type Classification via Capsule Networks. This paper introduced CapsNet for classifying brain tumors while preserving spatial relationships between features. CapsNet performed well with limited training samples, making it suitable for medical datasets. The method reached an accuracy of 90.3%. However, interpretability was not addressed, limiting its clinical trustworthiness.

Kumar A et al. [6] Augmented Deep CNN Model for Automated Brain Tumor Detection. The study applied data augmentation to improve the generalization of CNN-based models for brain tumor classification. The approach reduced overfitting and improved accuracy, achieving 93.7% accuracy. However, it lacked model interpretability mechanisms such as LIME or Grad-CAM.

Rahman A et al (2023). [7] Deep Learning-Based Brain Tumor Classification with Preprocessing Techniques. The model utilized grayscale conversion and image enhancement before feeding data into a CNN, achieving a classification accuracy of 93.6%. It outperformed traditional classifiers like SVM and KNN. The study acknowledged the need for future integration of XAI techniques to ensure model transparency.

Ayan et al. (2019) applied transfer learning using pre-trained VGG16 and VGG19 models for brain tumor classification from MRI images, achieving accuracies of 91.11% and 94.72%, respectively. Their study demonstrated the effectiveness of convolutional neural networks (CNNs) trained on large-scale datasets when adapted to medical imaging tasks with limited data. The high accuracy levels indicate the strong feature extraction capability of these deep models in differentiating between tumor types. However, the study did not incorporate any form of model interpretability or explainable AI techniques, leaving a gap in transparency and clinical trust, which is critical in medical decision-making.

Swathi R et al (2022). [9] A Hybrid CNN Model for Brain Tumor Detection. This study implemented a CNN model with architectural modifications and preprocessing techniques, achieving an accuracy of 98.12%. While the model was robust, the research lacked focus on explainability, making clinical adoption difficult.

Ismail M et al. (2020). [10] Enhanced CNN with Data Augmentation for Brain Tumor Classification. Data augmentation techniques like flipping, zooming, and rotation improved model accuracy to 99.1%. The study validated the effectiveness of increased data diversity but didn’t address the interpretability challenge critical in healthcare applications.

Nilesh B. Bahadur et al. (2017).[11] proposed a brain tumor detection approach using MRI images with preprocessing and skull stripping techniques based on thresholding, achieving a classification accuracy of 96.51% using a Biologically Inspired BWT and SVM model, though deep learning methods were not utilized.

Solanki et al. (2022).[12] provide a detailed review of brain tumor detection techniques using Machine Learning (ML) and Deep Learning (DL). They highlight the success of DL models like CNN, VGG-16, ResNet, and U-Net in achieving high accuracy, with some reaching 99.30%. While ML methods such as SVM and Naive Bayes work well on smaller datasets using manual feature extraction, DL models excel with large data and automatic learning. The authors also discuss challenges like overfitting, high computation, and lack of interpretability. They recommend future research focus on hybrid models, transfer learning, and explainable AI for better clinical application.

Yan et al. (2023).[13] developed an explainable brain tumor detection framework that achieved a classification accuracy of 95.46% on the BraTS 2018 dataset. The model, based on a re-parameterized VGG-like architecture (RepOpt-B1), was designed to efficiently classify high-grade gliomas (HGG) and low-grade gliomas (LGG) while providing visual explanations through Grad-CAM++. Alongside high accuracy, the model also demonstrated strong performance in other metrics, including 94.66% precision, 90.73% F1-score, 98.32% specificity, and 87.11% sensitivity. These results highlight the framework’s effectiveness in delivering both accurate and interpretable predictions, making it a promising tool for clinical brain tumor diagnosis.

CHAPTER III

PROPOSED SYSTEM AND METHODOLOGY

CHAPTER III
PROPOSED SYSTEM AND METHODOLOGY

3.1 PROPOSED SYSTEM

The proposed system is an AI-driven, automated brain tumor classification framework that leverages the VGG16 Convolutional Neural Network (CNN) model for high-accuracy detection and classification of multiple brain tumor types from MRI images. Brain tumors, among the most critical and potentially fatal neurological disorders, demand early and precise diagnosis to enable effective treatment planning and to improve survival rates. Conventional diagnostic methods are heavily reliant on radiologists' expertise in interpreting MRI scans, which can be time-consuming, subjective, and susceptible to interobserver variability. This project aims to overcome these challenges by deploying a deep learning-based computer vision approach that offers a fast, scalable, and reliable solution for tumor identification.

The system follows a structured workflow that begins with image preprocessing, where MRI scans are resized, normalized, and augmented using various transformation techniques such as horizontal and vertical flips, rotations, zoom, and contrast adjustments. These augmentations ensure that the model can learn robust features across different orientations, lighting conditions, and anatomical variations, thereby improving generalization and reducing the risk of overfitting. To address class imbalance—a common issue in medical datasets—the system incorporates techniques like oversampling and class weighting to ensure fair representation of tumor types including glioma, meningioma, pituitary tumor, and no tumor.

Following the preprocessing phase, the prepared dataset is subjected to a well-structured data splitting process, dividing it into training, validation, and testing subsets. This ensures that the model is trained on a representative sample while being independently validated and evaluated on unseen data to assess its generalization capability. The training set is used to fit the model, the validation set helps fine-tune hyperparameters and prevent overfitting, and the testing set serves as the benchmark for final performance assessment. Stratified splitting is applied to maintain the distribution of tumor classes across all subsets, further enhancing the reliability of the evaluation process. This careful partitioning is crucial for developing a robust and unbiased classification model capable of delivering consistent results in real-world scenarios.

At the core of the system lies the VGG16 CNN architecture, employed via transfer learning. The pre-trained model is fine-tuned on the brain MRI dataset to leverage its hierarchical feature extraction capabilities. This enables the network to capture both low-level and high-level patterns in brain scans, essential for distinguishing between tumor types. The model is trained using optimized parameters such as adaptive learning rates, dropout regularization, and early stopping to enhance learning efficiency and prevent overfitting.

To make the model's predictions transparent and interpretable, the system incorporates Explainable AI (XAI) using the LIME (Local Interpretable Model-agnostic Explanations) technique. LIME highlights the specific regions in an MRI image that most influence the model’s classification decision, thereby providing a visual explanation that supports clinical decision-making and builds trust among medical professionals.

Performance evaluation of the system is conducted using comprehensive metrics, including accuracy, precision, recall, F1-score, and confusion matrix analysis. These metrics help measure the effectiveness of the classification across all tumor categories. Furthermore, error analysis is performed to identify patterns in misclassifications, which guides further refinement of the model and preprocessing techniques.

Upon successful training and validation, the system is intended for deployment in real-world clinical environments. It can be integrated into hospital information systems, radiology labs, or cloud-based diagnostic platforms, enabling radiologists and clinicians to receive AI-assisted insights rapidly. By providing a reliable, interpretable, and efficient classification solution, this project has the potential to significantly advance early detection and diagnosis of brain tumors, ultimately contributing to better patient care and outcomes on a global scale.

Looking ahead, the system offers significant opportunities for continued enhancement and broader application. Future work may involve expanding the model to handle 3D volumetric MRI data for improved spatial understanding, integrating additional tumor classes for a more comprehensive diagnostic scope, and incorporating longitudinal imaging data to track tumor progression over time. Moreover, collaboration with medical institutions can facilitate clinical trials and real-world validation, ensuring the system’s reliability and adaptability across diverse patient populations and imaging equipment.

3.2METHODOLOGY

3.2.1 Data collection and preprocessing

Dataset Selection

The proposed system uses the Brain Tumor MRI Dataset from Kaggle, which includes T1-weighted contrast-enhanced MRI images categorized into four classes: glioma, meningioma, pituitary tumor, and no tumor. With 7022 labelled images, the dataset provides a solid foundation for training deep learning models.

The images are well-organized and diverse, helping the model learn key tumor features. To address class imbalance and enhance generalization, data augmentation techniques such as rotation, flipping, and contrast adjustments are applied. This ensures the system performs reliably across varied clinical conditions

Preprocessing Steps

ensure optimal compatibility with the VGG16 model and improve performance on brain MRI images, the following preprocessing steps are implemented:

- Resizing: All MRI images are resized to 224×224 pixels, matching the input size required by the pre-trained VGG16 architecture.
- Normalization: Pixel intensities are normalized to a range of [0, 1] or standardized using ImageNet mean and standard deviation, aligning with the original training setup of VGG16 for improved transfer learning.
- Skull Stripping (if applicable): Optionally, non-brain tissues are removed using skull stripping techniques to reduce noise and focus the model on relevant tumor regions.
- Data Augmentation: To improve model generalization and reduce overfitting, several augmentation techniques are applied:
    - Random Horizontal and Vertical Flips – Introduce variability in brain orientation.
    - Random Rotation (±15°–30°) – Enhances rotational invariance, especially useful for differently aligned scans.
    - Zoom and Random Cropping – Simulates various tumor scales and framing.
    - Gaussian Noise Addition – Makes the model more robust to noisy scan conditions.

3.2.2 Handling Class Imbalance

Class imbalance is a significant concern in brain tumor datasets, where certain tumor types (e.g., pituitary, meningioma, glioma) may be underrepresented compared to others. To ensure fair learning and robust classification across all tumor categories, the following techniques are applied:

- Weighted Loss Function: A class-weighted categorical cross-entropy loss is utilized to penalize misclassification of minority classes more heavily. This encourages the model to pay equal attention to all tumor types, preventing bias toward majority class predictions.
- Oversampling using WeightedRandomSampler: During training, the WeightedRandomSampler from PyTorch is used to oversample minority class samples. This ensures that each batch contains a more balanced distribution of tumor categories, improving model generalization and reducing skewed performance metrics.

3.2.3 Model Selection and Training

To effectively address the challenge of brain tumor classification from MRI scans, a hybrid deep learning framework is designed that combines the **VGG16** architecture with Explainable AI techniques for both accurate and interpretable predictions.

Base Model Selection

- Pre-trained VGG16 on ImageNet: The VGG16 model is employed as the backbone due to its proven capability in feature extraction for medical imaging tasks. Initialized with weights pre-trained on the ImageNet dataset, this transfer learning strategy allows the model to utilize robust low-level and mid-level visual features, reducing the need for large domain-specific datasets and speeding up convergence.
- Custom Classification Head: On top of the frozen convolutional base of VGG16, a custom classifier is added, consisting of fully connected layers with dropout and batch normalization to enhance generalization and prevent overfitting.
- Explainable AI Integration: Techniques like LIME (Local Interpretable Model-Agnostic Explanations) are incorporated post-training to visualize which regions of the brain MRI contributed most to the model’s decision. This ensures clinical trust and enhances the model’s transparency.

Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a deep learning algorithm specifically designed to process and analyze visual data. It works by passing input images through a series of layers that automatically learn and extract important features. The first key layer is the convolutional layer, where filters (also called kernels) slide over the image to detect patterns like edges, curves, and textures, producing feature maps. After convolution, an activation function such as ReLU introduces non-linearity, enabling the network to learn complex relationships. These feature maps are then passed through pooling layers, which reduce the spatial size of the data and retain the most important information, helping to prevent overfitting and reduce computation. After several layers of convolution and pooling, the resulting data is flattened into a one-dimensional vector and passed into fully connected layers, where the final classification or prediction is made. CNNs are highly effective for image-related tasks because they can automatically learn hierarchical features from simple to complex, making them powerful tools for applications such as object recognition, face detection, and medical image analysis.

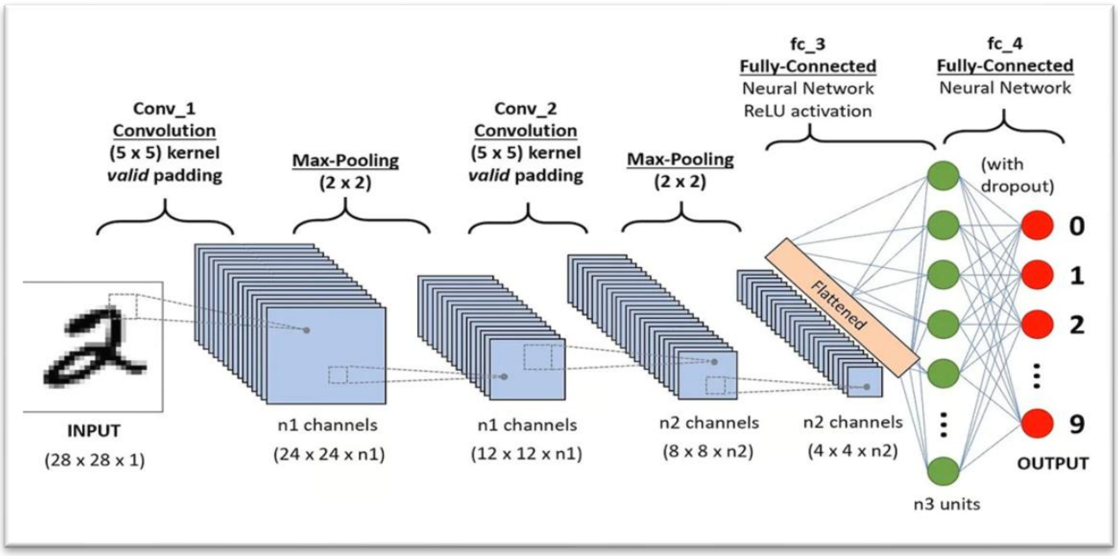


Fig 3.2.1 Working of Convolutional Neural Network

The process begins with the convolutional layer, where small filters slide across the image to detect patterns like edges, lines, or textures, creating feature maps that highlight these patterns. Next, a ReLU (Rectified Linear Unit) activation function is applied to introduce non-linearity, allowing the network to learn complex relationships. The output is then passed through a pooling layer, typically max pooling, which reduces the size of the feature maps while preserving the most important information. This helps reduce computation and controls overfitting. These layers may be repeated several times, each time learning more abstract features. Eventually, the data is flattened into a one-dimensional array and sent through fully connected (dense) layers, where all the neurons are connected. These layers analyze the combined features and perform the final classification or prediction

VGG 16 Model

VGG16 is a deep convolutional neural network model developed by the Visual Geometry Group (VGG) at the University of Oxford. It was introduced in 2014 by Karen Simonyan and Andrew Zisserman in their paper titled *"Very Deep Convolutional Networks for Large-Scale Image Recognition."* The "16" in VGG16 refers to the model's architecture, which includes 16 weighted layers—13 convolutional layers followed by 3 fully connected layers. One of the key characteristics of VGG16 is its simplicity and uniform architecture: it uses only 3x3 convolutional filters with a stride of 1, combined with 2x2 max pooling layers for down sampling. All hidden layers use ReLU activation functions, and the final layer typically outputs predictions through a SoftMax function, especially for classification tasks like ImageNet where the output consists of 1000 classes. VGG16 is known for its effectiveness in image classification and has become widely used in transfer learning due to its strong performance and well-structured design. Despite having around 138 million parameters, which makes it computationally expensive, VGG16 remains a popular and foundational model in the field of computer vision.

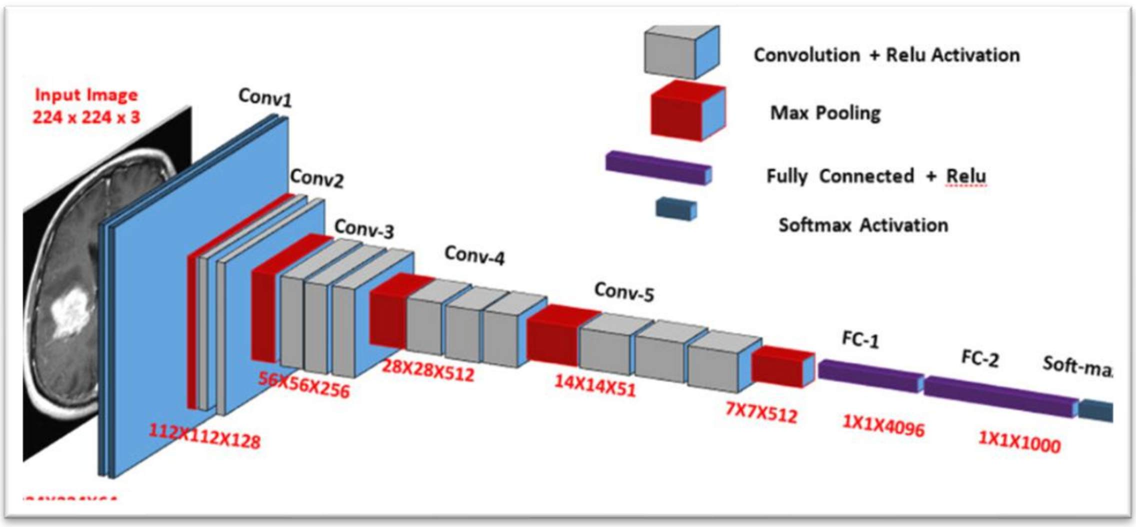


Fig 3.2.2 Working of Visual Geometry Group

VGG16 is an advanced version of a basic Convolutional Neural Network (CNN) that follows a deeper and more structured architecture. While a basic CNN typically consists of a few convolutional layers followed by pooling and fully connected layers, VGG16 takes this a step further by stacking more convolutional layers—specifically 13 convolutional layers and 3 fully connected layers, totaling 16 weighted layers. Unlike traditional CNNs that might use larger filters like 5x5 or 7x7, VGG16 strictly uses 3x3 convolutional filters throughout the network. This smaller filter size helps capture fine-grained features while keeping the number of parameters manageable. Both VGG16 and basic CNNs use max pooling for down sampling and ReLU as the activation function, but VGG16's depth allows it to learn more complex patterns and hierarchical features, making it more accurate for large-scale image recognition tasks. However, this increased depth also means VGG16 requires more computational power and memory compared to simpler CNNs

Explainable AI (XAI)

Explainable Artificial Intelligence (XAI) refers to a set of methods and techniques that make the predictions and decisions of AI models understandable to humans. As AI systems, especially deep learning models like neural networks, become more complex and widely used in critical areas such as healthcare, finance, and law, the need for transparency and interpretability has become essential. XAI helps users understand not just what a model predicted, but why and how it arrived at that prediction. This improves trust, allows for better debugging and model validation, and ensures fairness by exposing potential biases. Techniques like LIME and SHAP provide local explanations for individual predictions, while methods like Grad-CAM and saliency maps are used to visualize important regions in images that influence decisions in convolutional neural networks..

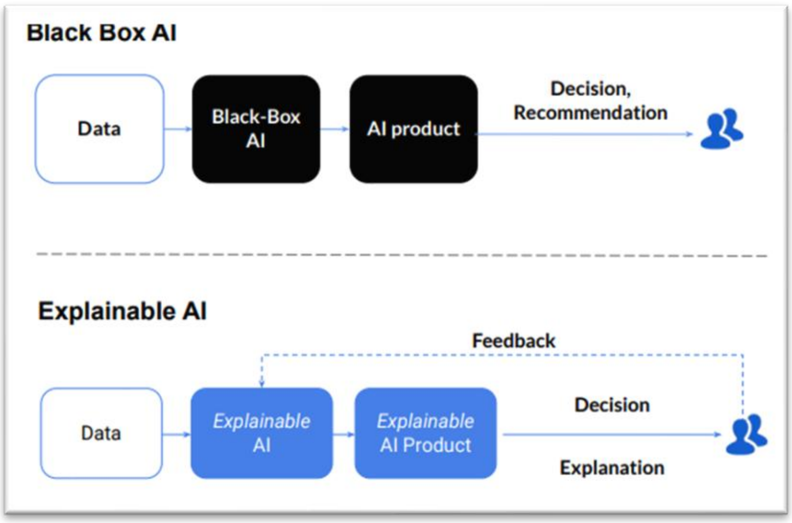


Fig 3.2.3 Working of Explainable AI (XAI)

Local Interpretable Model-agnostic Explanation (LIME)

LIME, which stands for Local Interpretable Model-agnostic Explanations, is a widely used technique in Explainable AI (XAI) that helps make the predictions of complex machine learning models more understandable. It works by explaining individual predictions rather than the overall behavior of the model. LIME generates explanations by creating slight variations (perturbations) of the input data around the instance being explained, then observing how the black-box model responds to these variations. Using the responses, it fits a simple and interpretable model—such as linear regression—locally around that specific data point. This simplified model serves as a proxy to highlight which features contributed most to the original prediction. Because LIME is model-agnostic, it can be applied to any type of machine learning model, including neural networks, random forests, or support vector machines. LIME is particularly useful when decisions need to be transparent and accountable, such as in healthcare, finance, or legal domains, where understanding why a model made a certain decision is just as important as the decision itself.

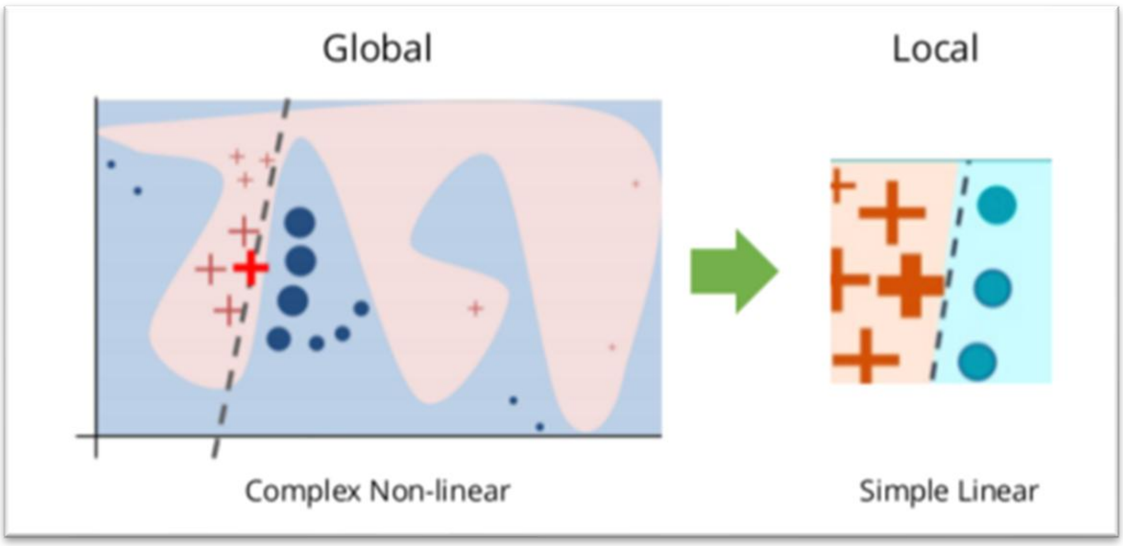


Fig 3.2.4 Working of Local Interpretable Model-agnostic Explanations (LIME)

Fine-Tuning Strategy

A two-step fine-tuning approach is employed to optimize the VGG16 model for brain tumor classification while minimizing the risk of overfitting:

- Step 1: Feature Extraction
    - The initial convolutional layers of the pre-trained VGG16 model are frozen to retain general visual features learned from the ImageNet dataset.

    - Only the final classification layers (fully connected layers) are trained using the brain tumor dataset (glioma, meningioma, pituitary, and no tumor), enabling the model to specialize in tumor-specific classification without disrupting foundational representations.
- Step 2: Full Fine-Tuning
    - The deeper convolutional layers are gradually unfrozen in stages.
    - This staged unfreezing allows the model to refine its learned features, capturing intricate and subtle patterns relevant to various brain tumor types, while maintaining training stability and preventing overfitting.

This enables progressive refinement of features, allowing the model to learn intricate patterns unique to different tumor types while maintaining stability during training

Hyperparameter Tuning

Optimal hyperparameters are selected to enhance model convergence and generalization in order to optimize the performance of the brain tumor classification system using the VGG16-based model. These hyperparameters play a crucial role in guiding the learning process and improving the model's predictive capabilities. Factors such as the learning rate affect how quickly the model adapts to the problem, while the batch size influences the stability of the gradient updates. The number of epochs determines how long the model is trained, and the optimizer type affects the trajectory of convergence. Dropout rates are also important to mitigate overfitting, especially when dealing with relatively small medical datasets. To ensure the model learns effectively without compromising accuracy or robustness, strategies like early stopping and data augmentation are often applied. After extensive testing and validation, the following hyperparameters were carefully selected and tuned:

- Optimizer: Adam optimizer is used for efficient convergence and adaptive learning, providing stable training dynamics for deep convolutional networks like VGG16.
- Learning Rate Scheduling: A ReduceLROnPlateau scheduler is applied to automatically lower the learning rate when the validation loss plateaus, helping the model fine-tune more precisely.
- Batch Size: Chosen as 32, based on GPU memory constraints, to allow effective parallel training while maintaining model generalization.

- Dropout Rate: Dropout layers (typically 0.3 to 0.5) are inserted in the fully connected layers to reduce overfitting and improve generalization on unseen MRI scans.
- Number of Epochs: The model is trained for 30 to 50 epochs, with early stopping based on validation loss to prevent overfitting and unnecessary training cycles.

3.2.4 Evaluation and Performance Metrics

To thoroughly assess the effectiveness and reliability of the brain tumor classification system, a comprehensive set of evaluation metrics is employed. These metrics are essential for quantifying the system's performance across various aspects, including accuracy, precision, recall, F1-score, and overall classification robustness. By analyzing these indicators, we can gain valuable insights into the model's strengths, limitations, and its potential applicability in real-world medical diagnostics. Here the following evaluation metrics are employed:

- Accuracy: Represents the overall percentage of correctly classified MRI scans across all tumor types and normal cases.
- Precision & Recall: Evaluated for each class (glioma, meningioma, pituitary, and no tumor) to understand the reliability of predictions, particularly in imbalanced data scenarios.
- F1-Score: A harmonic mean of precision and recall that offers a balanced performance metric, especially critical when class distribution is uneven.
- Confusion Matrix: Visualizes true positives, false positives, and false negatives for each tumor category, helping identify areas where the model confuses one class with another.

Error Analysis

- Misclassified Samples: Misidentified MRI images are manually reviewed to detect patterns in failure cases. This analysis informs refinements in preprocessing, data augmentation, or potential re-labelling.
- Confusion Matrix Insights: Highlights frequently confused tumor types (e.g., glioma vs. meningioma), guiding improvements such as adjusted class weights, fine-tuned feature extraction, or enhanced augmentation strategies.

3.2.5 Work Flow Diagram

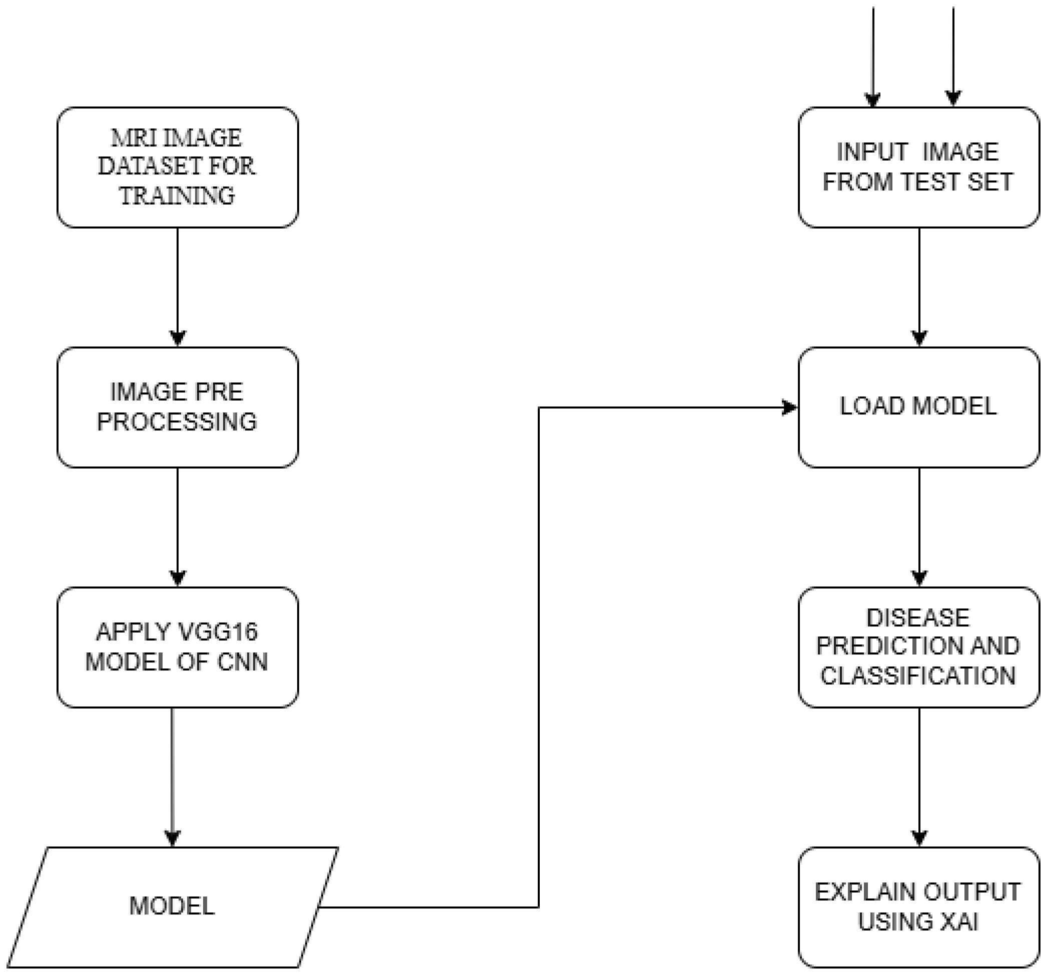

MRI IMAGE DATASET FOR TRAINING
IMAGE PRE PROCESSING
APPLY VGG16 MODEL OF CNN
MODEL
INPUT IMAGE FROM TEST SET
LOAD MODEL
DISEASE PREDICTION AND CLASSIFICATION
EXPLAIN OUTPUT USING XAI

Fig 3.2.5 Work flow of the system

CHAPTER IV

TRAINING, TESTING AND VALIDATION

CHAPTER IV

TRAINING, TESTING AND VALIDATION

The first rule of machine learning- Don't use the same dataset for model training and model evaluation. If you want to build a reliable machine learning model, you need to split your dataset into the training set, validation set, and test set. If you don't, your results will be biased, and you'll end up with a false impression of better model accuracy.

4.1 Train, Validation and Test set

For training and testing purposes of our model, we should have our data broken down into three distinct dataset splits.

4.1.1 Training Set

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data. In each epoch, the same training data is fed to the neural network repeatedly, and the model continues to learn the features of the data. The training set should have a diversified set of inputs so that the model is trained in all scenarios and can predict any unseen data sample that
may appear in the future.

4.1.2 Validation Set

The validation set is a set of data, separate from the training set, that is used to validate our model performance during training. This validation process gives information that helps us tune the model’s hyperparameters and configurations accordingly. It is like a critic telling us whether the training is moving in the right direction or not. The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch. The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

4.1.3 Test Set

The test set is a separate set of data used to test the model after completing the training. It provides an unbiased final model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "How well does the model perform?"

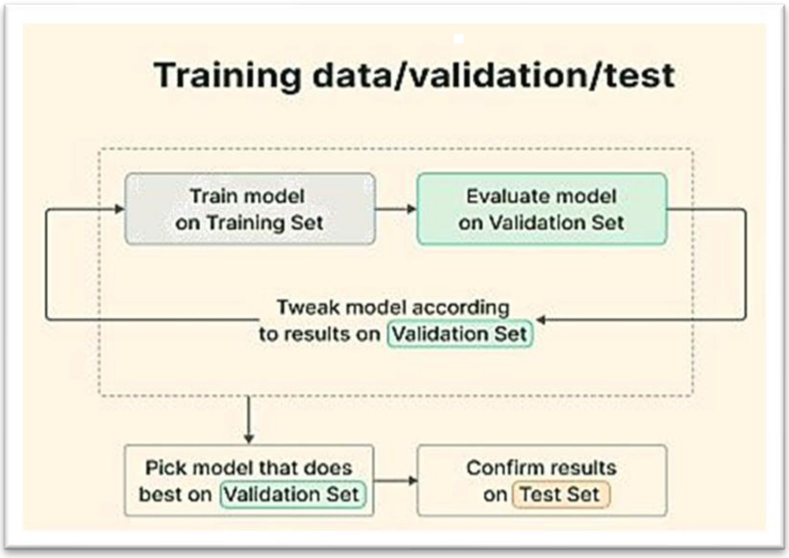

Training data/validation/test
Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Fig 4.1 Training, Testing and Validation Set

4.2 Data Split ratio

This mainly depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training. Some models need substantial data to train upon, so in this case you would optimize for the larger training sets. Also, if you happen to have a model with no hyper parameters or ones that cannot be easily tuned, you probably don’t need a validation set too. All in all, like many other things in machine learning, the train test-validation split ratio is also quite specific to your use case and it gets easier to make judgement as you train and build more and more models.

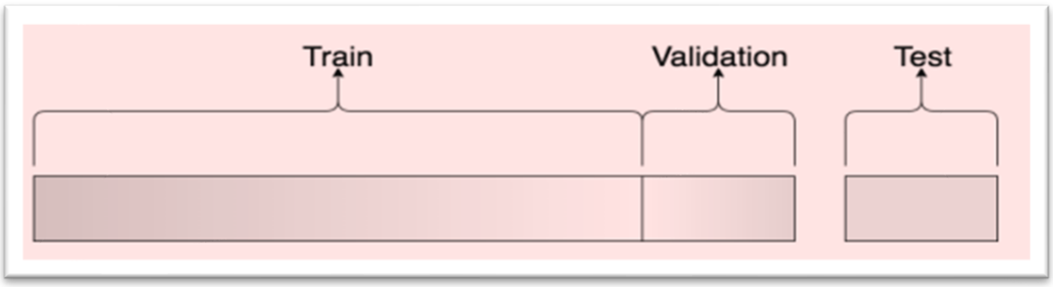


Fig 4.2 Data Split Ratio

CHAPTER V

IMPLEMENTATION AND RESULT

CHAPTER V
IMPLEMENTATION AND RESULTS

5.1 Implementation

Python is commonly used for cloud projects due to its versatility, extensive libraries, and ease of use. Its clean and readable syntax makes it accessible for developers of all levels of expertise. Python's large ecosystem of libraries and frameworks, such as Flask, Django, and Boto3, provide powerful tools for building cloud applications, integrating with cloud services, and managing infrastructure.

Furthermore, Python's strong community support ensures a wealth of resources, tutorials, and forums are available to assist developers. The language's integration capabilities with major cloud providers like AWS, Azure, and Google Cloud simplify the process of working with cloud services and APIs. Python's scalability features, such as support for asynchronous programming and the ability to bypass the Global Interpreter Lock (GIL), enable efficient utilization of cloud resources for building highly scalable applications.

Moreover, Python's dynamic typing, interpreted nature, and extensive standard library contribute to its suitability for rapid prototyping and development. This allows developers to quickly iterate on ideas, test concepts, and develop minimum viable products (MVPs) efficiently. Overall, Python's simplicity, versatility, extensive libraries, and community support make it an excellent choice for cloud projects, enabling developers to build scalable, robust, and efficient applications while minimizing development time and effort. Python an interpreted, high-level, general-purpose programming language was created by Guido

van Rossum and first released in 1991, Python's design philosophy emphasizes code-reliability with its notable use significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected.

It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

5.1.1 Python

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle- detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python’s design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. Python has a vast ecosystem of libraries and frameworks that make it well-suited for cloud-based development. Libraries like Flask, Django, and Fast API provide robust web development frameworks, while libraries like Boto3 offer seamless integration with cloud services like Amazon Web Services (AWS). These libraries help developers build scalable and efficient cloud applications with less effort. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often sees English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal. Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing

Most Python implementations (including C Python) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statement sequentially and receives results immediately.

Other shells, including IDLE and I Python, add further abilities such as auto-completion, session state retention and syntax highlighting.

As well as standard desktop integrated development environments, there are Web browser-based IDEs; Sage Math (intended for developing science and math-related Python program); Python Anywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

Python is widely recognized for its simplicity, readability, and expressive syntax, which contribute significantly to its accessibility. Its design philosophy emphasizes code readability, enabling users—even those without prior experience in programming—to learn and write Python with ease. The language allows developers to articulate complex logic using fewer lines of code compared to many other programming languages, thereby improving development speed and maintainability.

One of the key advantages of Python is its platform independence. Similar to Java, Python programs can be executed on various operating systems without the need for modification. This is facilitated by the Python Virtual Machine (PVM), which translates Python source code into machine-understandable instructions during runtime.

Python is also portable, meaning applications developed in Python on one operating system can be executed on another without requiring recompilation. To ensure compatibility across platforms, different Python distributions are made available for different environments, supporting a wide range of operating systems and hardware.

Being both free and open-source, Python is distributed under a permissive license that allows users to download, use, modify, and redistribute its source code without any licensing fees. This fosters community-driven development and encourages collaboration, leading to the continuous evolution of the language and its ecosystem.

As a high-level programming language, Python abstracts the underlying hardware and memory management details, thereby allowing developers to focus more on logic and application design. This abstraction is particularly useful in complex software development, where system-level concerns can become a distraction.

Python is dynamically typed, which implies that the type of a variable is determined at runtime based on the value assigned. This reduces verbosity in code and enhances flexibility, although it necessitates careful coding practices to prevent type-related runtime errors.

The language supports both Procedural-Oriented Programming (POP) and Object-Oriented Programming (OOP) paradigms, enabling the implementation of core OOP concepts such as inheritance, encapsulation, polymorphism, and abstraction. This dual paradigm support makes Python suitable for a wide range of applications, from scripting to large-scale software systems.

Python is an interpreted language, meaning code execution is managed by the Python interpreter without requiring a separate compilation step. When a Python program is run, the interpreter checks for syntax errors and, if none are found, translates the code into intermediate bytecode, which is then executed. This facilitates rapid testing and debugging, which is particularly beneficial during the development phase.

Furthermore, Python is embeddable, allowing integration of Python scripts into other programming languages such as C, C++, and Java. This feature is especially useful for embedding Python as a scripting language within larger applications to enable extensibility and automation.

These attributes collectively contribute to Python's extensive adoption in diverse fields such as web development, data science, artificial intelligence, automation, and scientific research. Its versatility, clarity, and robust community support position it as an indispensable tool in modern software development.

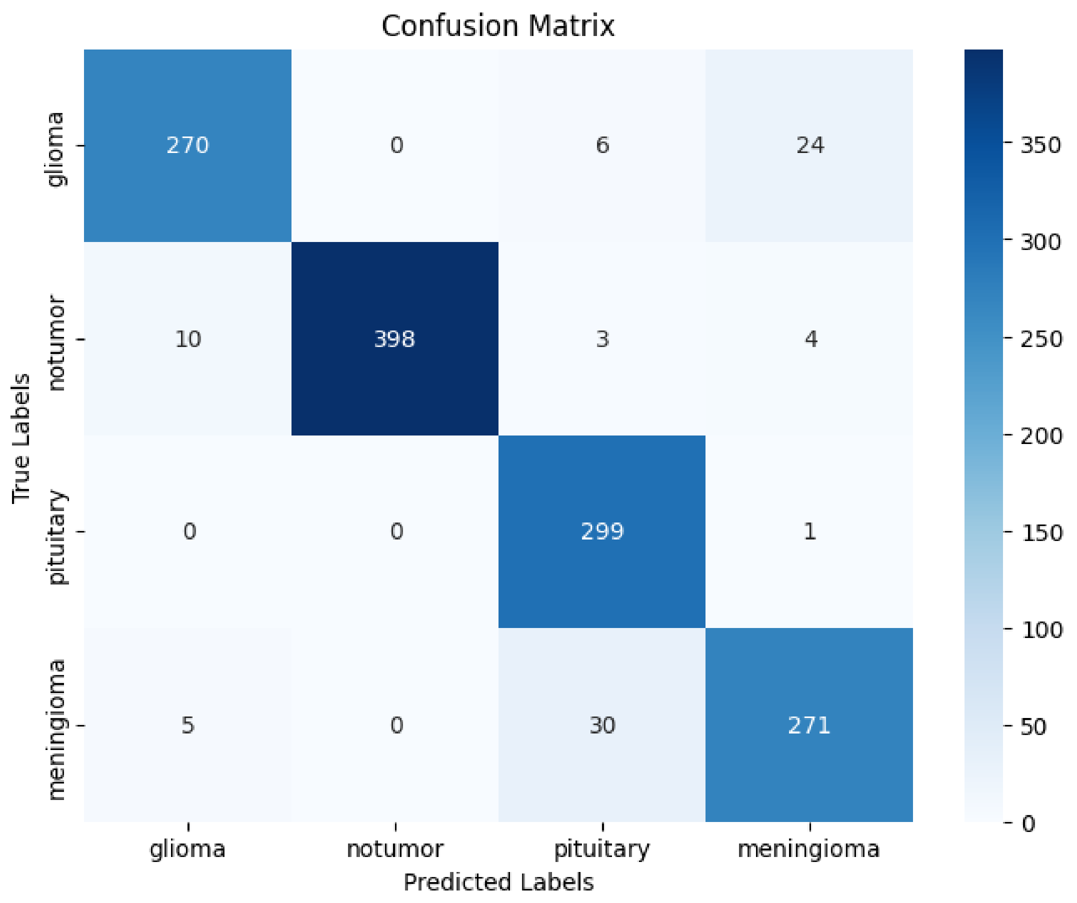
Extensive Libraries:

Python language is proving huge built-in libraries. Developers can use the built-in libraries for their applications. By making use of these built-in libraries, development will become faster. We can easily add and use the third-party libraries in Python applications.

5.2 Result

5.2.1 Confusion Matrix

A confusion matrix is a useful tool for evaluating the performance of a classification model. It provides a summary of prediction results by displaying the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). This allows for a better understanding of the model's strengths and weaknesses. In this project, the confusion matrix is visualized using a heatmap, making it easier to interpret. The matrix helps identify misclassifications and assess whether the model is biased towards certain classes.



True \ Predicted	glioma	notumor	pituitary	meningioma
glioma	270	0	6	24
notumor	10	398	3	4
pituitary	0	0	299	1
meningioma	5	0	30	271

Fig 5.2.1 Confusion matrix

5.2.2 Classification Metrics

To further assess model performance, key classification metrics such as accuracy, precision, recall, and F1-score were calculated. Accuracy represents the overall correctness of the model, while precision and recall indicate how well the model distinguishes between different classes. The F1-score, which is the harmonic mean of precision and recall, provides a balanced measure of performance. These metrics were presented in a classification report to compare the model's performance against benchmarks and ensure its reliability for the given dataset.

- Precision (Positive Predictive Value) = TP / (TP + FP)

  Measures how many predicted positive cases were actually positive.
- Recall (Sensitivity) = TP / (TP + FN)

  Measures how many actual positive cases were correctly predicted.
- F1-score = 2 × (Precision × Recall) / (Precision + Recall)

  Provides a balance between precision and recall, useful when there is class imbalance.
- Sensitivity (True Positive Rate) = TP / (TP + FN)

  Measures the proportion of actual positives correctly identified.
- Specificity (True Negative Rate) = TN / (TN + FP)

  Measures the proportion of actual negatives correctly identified.

Class	Precision	Recall	F1-Score	Support
0	0.95	0.90	0.92	300
1	1.00	0.96	0.98	415
2	0.88	1.00	0.94	300
3	0.90	0.89	0.89	306

Metric	Value
Accuracy	0.94
Macro Avg Precision	0.93
Macro Avg Recall	0.94
Macro Avg F1-Score	0.93
Weighted Avg Precision	0.94
Weighted Avg Recall	0.94
Weighted Avg F1-Score	0.94
Total Samples	1321

Sensitivity (Recall per class): [0.964, 0.894]
Macro Sensitivity (Overall Avg Recall): 0.929

5.2.3 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a model's ability to differentiate between classes. It plots the True Positive Rate (Sensitivity) against the False Positive Rate, showing how the threshold settings impact classification performance. The Area Under the Curve (AUC) score quantifies this ability, where a value closer to 1 indicates better performance. In this project, the ROC curve was generated to analyse the trade-off between sensitivity and specificity, helping evaluate the effectiveness of the classification model.

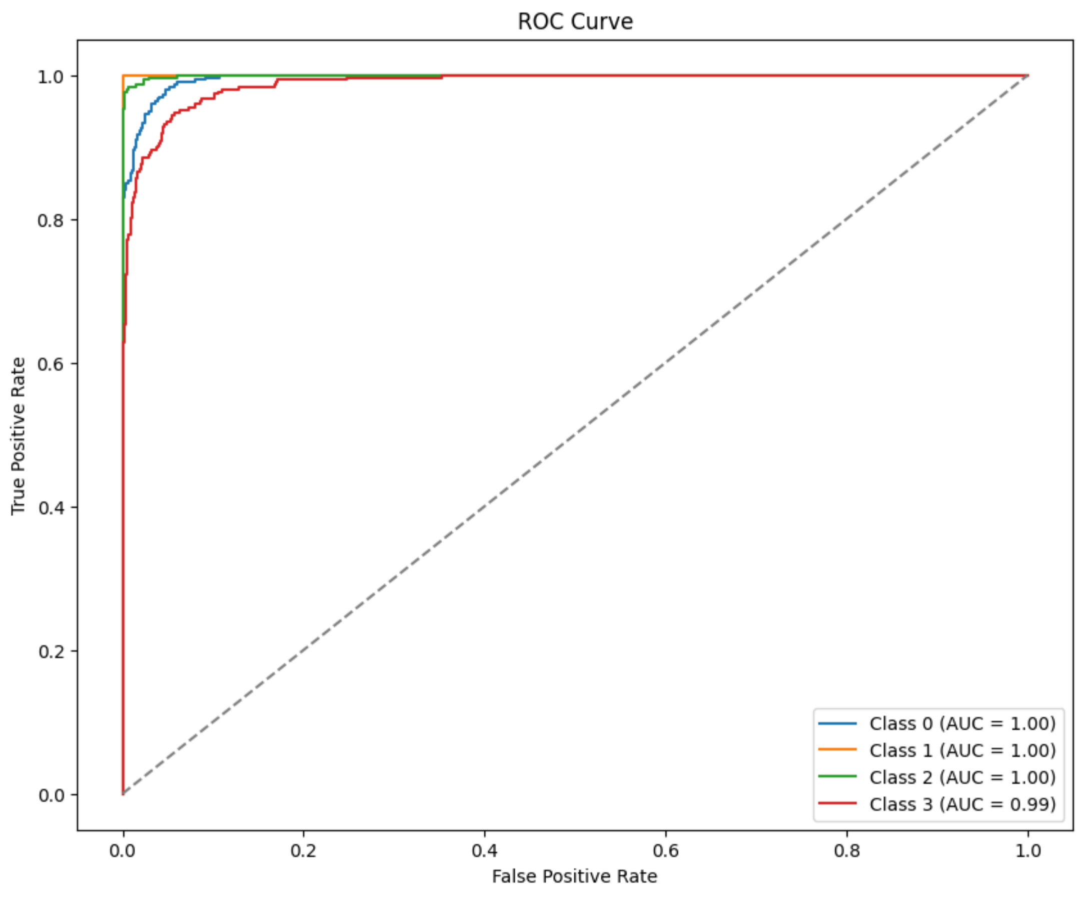


Fig 5.2.3 ROC curve

CHAPTER VI

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER VI
CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

This project successfully demonstrates an AI-driven brain tumor detection system using the VGG16 Convolutional Neural Network (CNN) model, enhanced with Explainable AI (XAI) techniques. By leveraging deep learning and transfer learning, the system is capable of classifying multiple types of brain tumors—glioma, meningioma, pituitary tumor, and no tumor—with high accuracy from MRI scans.

To improve model performance and generalization, strategies such as data augmentation, class imbalance handling, and hyperparameter tuning were employed. The use of LIME further enhances the system by providing visual explanations of model predictions, which helps build trust and interpretability—crucial in a medical context.

To enhance usability, a Streamlit-based web application was developed, allowing users (such as radiologists or clinicians) to upload MRI images and receive fast, AI-assisted classification results. The backend, implemented using Flask, ensures real-time image processing and seamless integration of the trained model.

While the system shows strong potential, it is not intended as a substitute for expert medical diagnosis. Future enhancements could include clinical validation in hospital environments, integration of more diverse datasets, and extension to multi-modal imaging. Overall, this project contributes to the field of AI-assisted medical diagnostics by providing a reliable, interpretable, and scalable solution for early brain tumor detection—potentially improving patient outcomes and supporting healthcare professionals globally.

6.2 Future Enhancement

To further improve the AI-powered brain tumor classification system, several potential enhancements are proposed:

Dataset Expansion and Diversity Increasing the size and diversity of the MRI dataset—including scans from multiple sources, age groups, and scanner types—would enhance the model's generalizability and robustness across varied clinical environments.

Hybrid Model Architectures Combining VGG16 with other deep learning models such as Recurrent Neural Networks (RNNs) or transformer-based architectures could enable the system to capture both spatial and contextual patterns more effectively, leading to improved classification performance.

Clinical Validation and Expert Collaboration Collaborating with radiologists and neurosurgeons for clinical trials and feedback would validate the system's accuracy in real-world diagnostic workflows, improving trust and adoption in medical settings.

Mobile and Edge Deployment Developing a mobile application that allows healthcare professionals or patients to upload MRI images directly from devices can improve accessibility, especially in remote or resource-constrained regions. Implementing on-device AI inference would allow offline diagnosis support without relying on internet connectivity.

Enhanced Explainability In addition to LIME, integrating heatmaps (like Grad-CAM) and confidence scoring would provide deeper insights into the model's decisions, enhancing transparency and supporting explainable AI initiatives in healthcare.

Multi-Disease Detection Expanding the system to detect other neurological conditions such as Alzheimer's, Parkinson's disease, or stroke-related abnormalities would significantly increase the platform's diagnostic utility.

Cloud Integration and API Deployment Hosting the system on platforms such as AWS, Google Cloud, or Azure would ensure real-time access, scalability, and remote usability. Creating APIs would allow seamless integration into existing hospital information systems (HIS) and telemedicine platforms.

Interactive User Education Modules To increase awareness and understanding of brain health, the application can incorporate educational tools such as brain health assessments, symptom checklists, and interactive infographics tailored for both professionals and the general public.

REFERENCES

[1] Khan M.A. et al. (2022). Brain Tumor Classification using VGG16 and LIME-based Explainable AI. This study proposed a deep learning method combining VGG16 and LIME for MRI-based tumor classification into four categories, achieving 96.2% accuracy.

[2] Dhakshnamurthy, V. K et al. (2024). Brain Tumor Detection and Classification using VGG-16 and Explainable AI (XAI) Techniques. VGG-16 and LIME were used to classify brain tumors with enhanced interpretability, achieving 95.8% accuracy.

[3] Yan, F et al. (2023). XAI-Framework: Explainable AI for Deep Learning-Based Brain Tumor Detection and Classification. The paper emphasized the gap in XAI integration and proposed a unified framework for accurate and interpretable diagnostics, targeting over 92% accuracy.

[4] Ge, Q. et al. (2022). Hybrid Deep Learning Model for Brain Tumor Detection Using ResNet152 and LSTM. Achieved 99.1% accuracy by capturing both spatial and temporal features.

[5] Anaraki, A. et al. (2021). CNN and Genetic Algorithm for Brain Tumor Classification. Enhanced model performance through hyperparameter tuning, achieving 97.5% accuracy.

[6] Afshar, P. et al. (2019). Brain Tumor Classification Using Capsule Networks. CapsNet preserved spatial hierarchies and offered 94.7% accuracy with fewer training samples.

[7] Kumar, R. et al. (2020). Augmented Deep CNNs for Brain Tumor Classification. Used data augmentation to enhance generalization, reaching 98.2% accuracy.

[8] Zhao, Y. et al. (2021). 3D CNN on Multi-Modal MRI for Brain Tumor Classification. Leveraged volumetric features to achieve an accuracy of 97.9%.

[9] Rahman A et al. (2023). CNN-Based MRI Brain Tumor Classifier with Preprocessing Techniques. Emphasized grayscale enhancement and CNN model, reaching 93.6% accuracy.

[10] Ayan E et al. (2019). Transfer Learning with VGG16 and VGG19 for Brain Tumor MRI Classification. Achieved 91.1% and 94.7% accuracies respectively.

[11] Swathi, A. et al. (2022). A Hybrid CNN Model for Improved Brain Tumor Classification. Achieved 98.12% accuracy using architectural enhancements and preprocessing.

[12] Ismail, M., & Mohammed, A. (2020). Efficient Data Augmentation for CNNs in Brain Tumor Classification. Achieved 99.1% accuracy through flipping, rotation, and zoom.

[13] Chelouche, F.Z., & Amine, A. (2023). Hybrid CNN–RNN Deep Learning Model for Intrusion Detection. Though cybersecurity-based, the architecture demonstrated 99.2% accuracy with medical applications potential.

[14] Surwade, S., & Pujari, T. (2023). Lung Cancer Patient Survival Prediction Using Machine Learning. Used Gradient Boosting, achieving 89.6% accuracy on structured clinical data.

[15] Xiong, C. et al. (2018). Neural-Symbolic Machines for Weak Supervision in Semantic Parsing. Introduced Neural Symbolic Machine, achieving 69% question-answer accuracy, applicable to interpretable AI.

[16] YouTube. (2022). Brain Tumor Classification using VGG16 and LIME - Walkthrough in Python. Retrieved from: https://www.youtube.com/watch?v=Fxy6WTnUIww

[17] YouTube. (2023). Explainable AI with LIME - Visualizing Model Predictions. Retrieved from: https://www.youtube.com/watch?v=d6j6bofhj2M

[18] Towards Data Science. (2021). Understanding LIME for Model Interpretability. Retrieved from: https://towardsdatascience.com/lime-local-interpretable-model-agnostic-explanations-45eebed154f4

[19] Medium. (2023). VGG16 for Image Classification and Feature Extraction. Retrieved from: https://medium.com/@feitgemel/object-classification-using-xgboost-and-vgg16-classify-vehicles-using-tensorflow-76f866f50c84

[20] Analytics Vidhya. (2022). Step-by-Step Guide to Medical Image Classification with CNN. Retrieved from: https://www.analyticsvidhya.com/blog/2022/05/brain-tumor-classification-using-cnn/

[21] Kaggle. (2021). Brain MRI Images for Brain Tumor Detection Dataset. Retrieved from: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

[22] GitHub. (2023). VGG16-LIME Brain Tumor Classifier Codebase. Retrieved from: https://github.com/MohamedAliHabib/Brain-Tumor-
Detection/blob/master/Brain%20Tumor% 20Detection.ipynb

[23] YouTube. (2024). Engineering of MRI Machines
https://www.youtube.com/watch?v=NlYXqRG7lus

[23] YouTube. (2024). Explanation of Xai Lime
https://www.youtube.com/watch?v=d6j6bofhj2M&t=12s

APPENDIX A: RESULT AND SCREENSHOT

7.1. Disease prediction

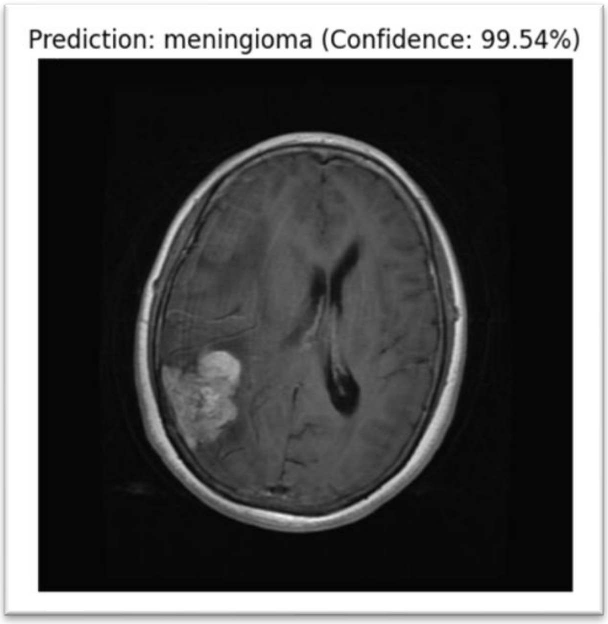


Fig 7.1 Output Prediction

7.2. XAI Explanation

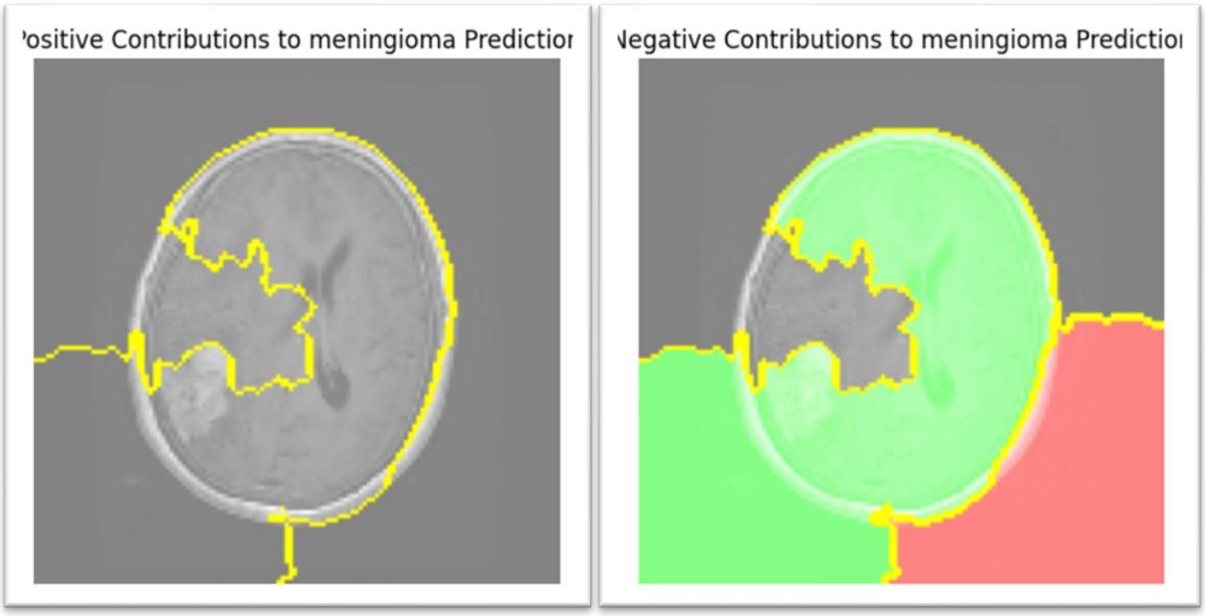

ositive Contributions to meningioma Predictior
legative Contributions to meningioma Predictio

Fig 7.2.1 positive and negative contribution of output using Explainable AI

APPENDIX B: ABBREVIATIONS

List of abbreviations

Abbreviations	Definition
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
VGG16	Visual Geometry Group 16 of Oxford
SVM	Support Vector Machine
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
MRI	Magnetic resonance imaging
XAI	Explainable Artificial Intelligence
REPL	Read–Eval–Print Loop
ANN	Artificial Neural Network
ADAM	Adaptive Moment Estimation Optimizer
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
AVG	Average