MODERN EDUCATION SOCIETY'S NESS WADIA COLLEGE OF COMMERCE, PUNE

B.B.A.(C.A.) Sem-V

(CA-505)

A PROJECT REPORT ON

Neuro Vision – Brain Tumor Detection Using MRI Images

2025-2026

BY

AYUSH AMOL BAHIRAT

ROLL NO - 03

&

VIVEK SUDARSHAN SURYAWANSHI

ROLL NO - 66

UNDER THE GUIDANCE OF

PROF. SEEMA PURANDARE

INDEX

1. Introduction

The rapid growth of medical imaging technologies has transformed the healthcare industry by enabling physicians to detect, diagnose, and monitor diseases with a higher degree of accuracy than ever before. Among these medical conditions, brain tumors remain one of the most life-threatening and challenging diseases to diagnose and treat effectively. Brain tumors, whether Glioma, Meningioma, Pituitary, or No Tumor, pose serious risks to patients due to their impact on vital neurological functions. Early detection and accurate classification are crucial for determining the most effective treatment plan and for improving patient survival rates. However, conventional diagnostic methods, which rely heavily on manual inspection of MRI and CT scan images by radiologists, can often be time-consuming, error-prone, and subject to human fatigue.

In response to these challenges, **NeuroVision** has been developed as an artificial intelligence (AI)-based solution designed to automate and enhance the process of brain tumor detection. The primary goal of this project is to leverage the power of machine learning, specifically Convolutional Neural Networks (CNNs), to analyze medical images and identify the presence of tumors with a high level of accuracy. NeuroVision not only classifies brain images into categories such as *normal (no tumor)* and *tumor (with subtypes)* but also highlights the affected areas using visualization techniques such as heatmaps. This added interpretability provides doctors and medical professionals with clear, actionable insights into the patient's condition.

The motivation behind NeuroVision lies in the need for reliable and efficient computer-aided diagnosis systems. With the increasing availability of medical imaging data and the advancements in deep learning algorithms, it is now possible to design systems that can match or even surpass human-level performance in image analysis tasks. By automating the detection and classification process, NeuroVision reduces the reliance on manual inspection, decreases the time required for diagnosis, and minimizes the risk of oversight or human error. This is particularly important in resource-constrained environments where access to highly experienced radiologists may be limited.

Another key aspect of this project is its emphasis on accessibility and practicality. NeuroVision is designed to run on modest hardware, such as a personal computer equipped with a CPU, while also allowing for acceleration using GPU resources for faster training and inference. The use of widely available open-source frameworks like PyTorch, OpenCV, and Scikit-learn makes the system cost-effective and adaptable for academic research, clinical trials, and real-world deployment in hospitals.

The scope of NeuroVision extends beyond simple tumor detection. By incorporating classification and visualization, the system supports doctors in differentiating between various tumor types and in identifying the exact regions of abnormal growth. This functionality is critical in planning treatment options such as surgery, radiation therapy, or chemotherapy. Furthermore, NeuroVision has been designed with extensibility in mind—future improvements can include integration with larger datasets, advanced neural network architectures, and even real-time clinical decision support systems.

In summary, the **NeuroVision project** represents a comprehensive application of artificial intelligence in the domain of medical imaging. It addresses a critical healthcare challenge by providing an automated, accurate, and efficient tool for brain tumor detection and classification. By reducing the workload on doctors and minimizing the chances of diagnostic errors, NeuroVision contributes towards improving patient care and outcomes. This project is a step forward in demonstrating how modern computing techniques can be harnessed to solve real-world medical problems and pave the way for more intelligent and accessible healthcare solutions.

1.1 Problem Definition

“ There is a critical need for an automated, reliable, and efficient system that can detect and classify brain tumors from medical images with high accuracy, while also highlighting affected regions to support clinical decision-making. Such a system should minimize human error, reduce diagnostic time, and improve accessibility to quality healthcare.”

Brain tumors represent a critical medical condition that can severely affect a patient’s neurological functions, overall health, and survival chances. The timely and accurate detection of tumors is therefore essential for effective treatment planning and improved clinical outcomes. Traditionally, radiologists and medical practitioners rely on manual analysis of MRI (Magnetic Resonance Imaging) or CT (Computed Tomography) scans to identify abnormal growths in brain tissues. While this approach has been the gold standard for decades, it presents several fundamental challenges.

Firstly, manual diagnosis is a **time-intensive process**. The volume of medical imaging data in a hospital setting can be overwhelming, and radiologists are required to meticulously analyze each image for signs of abnormality. In high-pressure environments, the time taken to reach a diagnostic decision can delay treatment initiation, which is particularly harmful in the case of aggressive tumor growth.

Secondly, manual interpretation is **prone to human error**. Even experienced specialists may miss small or early-stage tumors due to fatigue, image complexity, or the subtlety of visual cues in scans. Inconsistent lighting, image noise, and differences in imaging equipment further increase the risk of misinterpretation. These errors may result in delayed treatment or misdiagnosis, potentially endangering patient lives.

Thirdly, the **classification of tumors into different types** poses another challenge. Distinguishing between normal brain tissue, Glioma, Meningioma, Pituitary, or No Tumor, requires a high level of expertise. Each category carries distinct treatment protocols and prognoses, making accurate classification an integral part of the diagnostic process. Inconsistencies in classification between specialists can affect treatment strategies and outcomes.

In addition to accuracy concerns, the **availability of expert radiologists** is limited, especially in rural or resource-constrained regions. This shortage contributes to disparities in healthcare access and outcomes, leaving many patients without timely diagnosis and care.

With the advent of advanced computing techniques, there is a clear need for a **computer-aided system** that can address these issues by:

- Automating the detection of tumors in brain scans.
- Accurately classifying the type of tumor when present.
- Providing visual cues, such as heatmaps or highlighted regions, to support interpretability and assist doctors in decision-making.
- Reducing diagnosis time while maintaining or improving accuracy.

By addressing this problem, NeuroVision aims to bridge the gap between the growing demand for accurate brain tumor diagnosis and the limitations of manual interpretation, ultimately contributing to improved patient outcomes and healthcare efficiency.

1.2 Need for the New System

The diagnosis and treatment of brain tumors demand both precision and timeliness. In conventional medical practice, radiologists manually examine MRI and CT images to detect the presence of abnormal tissues. While this traditional method has been effective, it is increasingly insufficient to meet the demands of modern healthcare. Several limitations in the existing diagnostic process highlight the urgent need for a new, technology-driven system such as **NeuroVision**.

1. Increasing Complexity of Medical Data

The amount of medical imaging data generated every day has grown significantly. Hospitals and diagnostic centers handle thousands of scans daily, making it difficult for radiologists to analyze each image with the same level of focus and detail. The sheer volume of data creates delays in reporting and increases the chances of oversight. A new system is required to **process large datasets automatically and consistently**, ensuring that no critical details are missed.

2. Minimizing Human Error

Human error is an unavoidable factor in manual image analysis. Even highly skilled radiologists may overlook small abnormalities due to fatigue, stress, or the limitations of the human eye in detecting subtle variations in pixel intensity. This can lead to misdiagnosis or delayed diagnosis. An AI-based system reduces this risk by **detecting tumors with consistency and objectivity**, unaffected by human limitations.

3. Need for Faster Diagnosis

Time is a crucial factor in treating brain tumors. Tumors can progress rapidly, and delays in detection or treatment can negatively impact patient outcomes. Manual diagnosis often involves multiple steps and expert consultations, which prolongs the process. The new system accelerates diagnosis by providing **real-time or near real-time results**, enabling doctors to initiate treatment sooner.

4. Classification of Tumor Types

Detecting a tumor is only part of the challenge. Determining whether the tumor is Glioma, Meningioma, Pituitary, or No Tumor, and identifying its type, is equally important in deciding the treatment plan. Existing systems often require additional manual analysis or biopsies to classify tumors. NeuroVision incorporates **classification capabilities within its workflow**, offering both detection and categorization in one integrated platform.

5. Accessibility and Resource Constraints

In many regions, especially rural and underdeveloped areas, access to experienced radiologists is limited. Patients may have to travel long distances or wait extended periods for expert review of their scans. The new system addresses this issue by providing **accessible, automated diagnostic support** that can be deployed even in smaller hospitals or clinics with limited resources. This democratizes access to quality healthcare by reducing dependency on a small pool of specialists.

6. Support for Decision-Making

Radiologists and doctors require not only diagnostic results but also **visual interpretability** of those results. By highlighting the affected regions through heatmaps or segmentation masks, the system offers an additional layer of support. This improves confidence in the diagnosis and provides medical professionals with clear insights that can be directly correlated with clinical knowledge.

7. Cost-Effectiveness

The proposed system leverages **open-source tools and frameworks** such as PyTorch, OpenCV, and Scikit-learn. This ensures that the solution remains cost-effective and does not impose additional financial burdens on healthcare institutions. In contrast, manual methods require repeated consultations, specialized manpower, and longer hospital stays, which increase treatment costs.

In summary, the need for the new system arises from the **limitations of manual diagnosis** and the **growing demand for accurate, efficient, and accessible healthcare solutions**. NeuroVision directly addresses these challenges by offering a reliable, automated platform for brain tumor detection and classification. It reduces diagnostic delays, minimizes errors, supports medical decision-making, and makes advanced diagnostic tools accessible across diverse healthcare environments.

1.3 Project Scope

The **NeuroVision Brain Tumor Detection System** is developed to address the critical need for an automated, efficient, and accurate solution for detecting and classifying brain tumors using MRI images. The primary focus of the project is to identify whether a brain tumor is present and classify it into specific categories. This system is designed to assist medical professionals, such as radiologists and clinicians, by providing rapid and reliable classification results, thereby improving diagnostic efficiency and reducing the risk of human error.

The scope of the project defines the boundaries, objectives, and functionalities of the system. By clearly outlining what the system will achieve, it ensures that the project remains focused and aligned with its intended goals. The NeuroVision system is aimed at providing an accessible and user-friendly platform that allows users to obtain meaningful diagnostic insights from MRI scans without requiring extensive technical expertise.

Objectives

The main objectives of the NeuroVision Brain Tumor Detection System are as follows:

1. **Automated Tumor Classification:**
   The system is designed to classify MRI scans into four predefined categories:
   - No Tumor
   - Glioma
   - Meningioma
   - Pituitary Tumor

2. This classification ensures that medical professionals can quickly and accurately identify the presence of a tumor and understand its type. By automating this process, the system reduces the need for manual interpretation of MRI scans, which can be time-consuming and prone to error.

3. **Enhanced Diagnostic Efficiency:**
   The system aims to reduce the overall time required for tumor detection by providing rapid classification results. This allows clinicians to prioritize cases more effectively and focus on treatment planning for patients who require urgent attention.

4. **High Accuracy and Reliability:**
   Leveraging advanced machine learning and deep learning algorithms, the

system is trained to recognize patterns in MRI images corresponding to different tumor types. The goal is to achieve high levels of accuracy and minimize false positives and false negatives, ensuring that the results are dependable and clinically useful.

5. **Ease of Use and Accessibility:**
   The system includes a simple and intuitive interface that allows users to upload MRI scans and receive immediate classification results. By focusing on user-friendly design, the system ensures accessibility for medical professionals regardless of their technical background, enabling a broader adoption in hospitals, diagnostic centers, and research institutions.

Functional Scope

The functional scope defines the features and capabilities of the NeuroVision system. These functionalities are specifically included in the current version of the project:

1. **Image Preprocessing:**
   MRI scans often contain noise or inconsistencies that can affect classification accuracy. The system preprocesses each image by performing operations such as noise reduction, normalization, and resizing. This ensures that the input data is standardized and optimized for accurate classification.

2. **Feature Extraction and Classification:**
   The core functionality of the system is its ability to extract relevant features from the MRI images and use these features to classify the scans. Advanced machine learning and deep learning models analyze patterns and characteristics within the images to determine the tumor type or the absence of a tumor.

3. **Prediction Output:**
   Once processing is complete, the system outputs the classification result clearly, indicating the predicted tumor type or confirming that no tumor is present. This output is designed to be concise, easily interpretable, and actionable for medical professionals.

4. **Performance Evaluation:**
   To ensure reliability, the system incorporates mechanisms to evaluate its own performance. Metrics such as accuracy, precision, recall, and processing time are considered to validate the effectiveness of the classification model. Regular assessment ensures that the system maintains consistent and dependable

performance during use.

5. **Support for Clinical Decision-Making:**
   While the system does not provide treatment recommendations or malignancy predictions, it serves as a valuable diagnostic support tool. By providing quick and accurate classification results, the system helps clinicians make informed decisions regarding further investigations or interventions.

6. **Scalability and Future Integration:**
   The project scope includes the possibility of future enhancements, such as incorporating additional tumor types, expanding the dataset, or integrating the system with clinical workflows. These future developments are beyond the current scope but are considered during design to allow scalability and adaptability over time.

Conclusion

The NeuroVision Brain Tumor Detection System is scoped to provide a robust and reliable solution for automated brain tumor classification using MRI images. By focusing on high accuracy, user accessibility, and efficient processing, the system addresses the challenges associated with manual tumor detection. It provides a foundation for medical professionals to enhance their diagnostic capabilities, improve patient care, and streamline clinical workflows.

While the current system focuses on classification into four categories and does not include tumor visualization or malignancy assessment, it establishes a solid framework for future improvements. This focused scope ensures that the project remains achievable, targeted, and effective in meeting its primary objectives.

2. Analysis

The analysis phase of the NeuroVision Brain Tumor Detection System focuses on evaluating the overall feasibility and practicality of developing an automated brain tumor classification tool using MRI images. This step ensures that the project is viable and aligns with the objectives of providing an efficient, accurate, and user-friendly solution for medical professionals.

The proposed system aims to classify brain MRI scans into four categories: No Tumor, Glioma, Meningioma, and Pituitary Tumor. Achieving this objective requires careful consideration of the availability of data, computational resources, and appropriate algorithms. Publicly available MRI datasets, along with open-source deep learning frameworks such as TensorFlow, Keras, or PyTorch, will be utilized for model training and testing. Image preprocessing techniques, including noise reduction, normalization, and resizing, will ensure consistent input quality for the classification model. GPU-enabled systems will facilitate the training process, while the final predictions can be executed efficiently on standard computing setups, making the system accessible for clinical use without requiring specialized hardware for everyday operation.

The development of NeuroVision is also practical from an economic perspective. Since the system relies heavily on open-source tools and publicly available datasets, development costs are significantly reduced. There is no need for database integration or long-term storage, which further minimizes operational expenses. The benefits provided by the system, such as faster diagnostic support, reduced human error, and improved access to reliable brain tumor classification, justify the investment and make the project cost-effective for medical institutions and research centers.

From an operational standpoint, NeuroVision is designed to be user-friendly and easily adoptable in real-world clinical settings. Medical professionals can interact with the system through a simple interface, uploading MRI scans and receiving rapid classification results without extensive technical training. The system does not replace the clinician’s judgment but serves as a supportive tool to enhance diagnostic efficiency. Maintenance requirements are minimal, and the design allows for future scalability, such as the addition of more tumor types or improved algorithms, without affecting existing functionality.

In terms of hardware and software requirements, the system can be implemented using standard computing infrastructure. An Intel i5 or AMD Ryzen 5 processor with at least 8 GB of RAM is sufficient, although 16 GB is recommended for smoother processing. A GPU with CUDA support is required for model training, while storage of around 500 GB is needed for datasets and project files. On the software side, Python 3.x will be used

along with libraries such as OpenCV and PIL for image processing, and frameworks like TensorFlow, Keras, or PyTorch for deep learning model development. Development and testing can be carried out in IDEs such as Jupyter Notebook, PyCharm, or VS Code.

Overall, the analysis indicates that the NeuroVision Brain Tumor Detection System is both feasible and practical. With the available technology, resources, and accessible datasets, the system can be successfully developed to provide accurate and reliable classification of brain tumors. It has the potential to improve diagnostic workflows, assist clinicians in decision-making, and provide a scalable foundation for future enhancements in automated medical imaging.

2.1 Feasibility Study

The feasibility study evaluates the viability of developing and implementing the **NeuroVision Brain Tumor Detection System**. It examines whether the project can be carried out successfully by analyzing technical, economic, and operational aspects. The objective of this study is to ensure that the system is practical, cost-effective, and suitable for integration into clinical workflows.

a. Technical Feasibility

Technical feasibility assesses whether the necessary technology, tools, and expertise are available to develop the system efficiently. The NeuroVision system relies on machine learning and deep learning algorithms to classify MRI scans into four categories: No Tumor, Glioma, Meningioma, and Pituitary Tumor.

A critical component for technical feasibility is the availability of MRI datasets. Publicly available datasets or collaborations with hospitals provide sufficient data for model training and testing. Preprocessing of these images, including noise reduction, normalization, and resizing, ensures that the input data is consistent and of high quality for accurate classification.

Deep learning frameworks such as TensorFlow, Keras, and PyTorch will be used to design and implement the classification model. Python, with libraries like OpenCV and PIL, allows for efficient handling of MRI images during preprocessing and feature extraction. GPU-enabled systems are essential during the training phase to reduce computation time, while predictions can be performed on standard computers, making the system accessible to clinicians without specialized hardware.

The technical feasibility is further reinforced by the simplicity of the system's architecture. Since NeuroVision does not require database management or long-term storage, the design remains streamlined and easier to implement. All required tools, technologies, and expertise are readily available, confirming that the project is technically feasible.

b. Economical Feasibility

Economic feasibility evaluates whether the system can be developed and implemented within a reasonable budget while providing sufficient benefits. The NeuroVision system is designed to be cost-effective by utilizing open-source software, frameworks, and publicly available datasets. This significantly reduces development costs compared to proprietary solutions.

Hardware requirements, including GPU-enabled systems for model training and standard computing devices for predictions, are modest and manageable within typical institutional budgets. The absence of database requirements further reduces infrastructure costs.

The system provides substantial benefits, including faster diagnosis, minimized risk of human error, and improved access to accurate brain tumor classification for clinicians and researchers. These benefits justify the initial investment in hardware, software, and human resources for model development and testing. The cost-effectiveness and potential impact on clinical workflow make the project economically viable for hospitals, diagnostic centers, and research institutions.

c. Operational Feasibility

Operational feasibility examines whether the system can be implemented effectively in real-world clinical settings and whether it can be adopted by end-users efficiently. NeuroVision is designed with a simple and intuitive interface, allowing medical professionals to upload MRI scans and obtain classification results quickly without the need for technical expertise.

The system does not replace the role of the clinician; instead, it acts as an auxiliary diagnostic tool to support faster and more reliable decision-making. By automating the classification process, NeuroVision reduces the manual workload on radiologists, enabling them to focus on patient care and complex diagnostic tasks.

Maintenance requirements for the system are minimal. Since it processes images in real-time without storing patient data, there is no need for database management or continuous backend maintenance. The modular design of the system allows for future upgrades, such as adding additional tumor types or improving classification algorithms, without disrupting existing functionality.

The operational feasibility is also enhanced by the flexibility of the system. It can be deployed in hospitals, diagnostic labs, and research institutions with minimal adaptation to existing workflows. The system’s simplicity, efficiency, and ease of use confirm that it is operationally feasible and ready for practical implementation.

Conclusion

The feasibility study confirms that the NeuroVision Brain Tumor Detection System is viable across technical, economic, and operational dimensions. The availability of required technologies, accessible datasets, and cost-effective development options ensures that the project can be successfully implemented. The system provides a

practical and efficient solution for automated brain tumor classification, enhancing diagnostic workflows and supporting medical professionals in delivering accurate and timely patient care. Its modular and scalable design also ensures potential for future improvements and expanded functionality, making it a robust foundation for ongoing advancements in medical imaging and AI-assisted diagnostics.

2.2 Hardware and Software Requirements

The successful implementation of the **NeuroVision Brain Tumor Detection System** depends on specific hardware and software components that ensure smooth development, training, and deployment of the classification model. The following sections outline the necessary resources.

Hardware Requirements

The system requires hardware capable of handling image processing and deep learning tasks efficiently:

- **Processor (CPU):** Intel i3 (minimum) / i5 or higher (recommended) or AMD Ryzen 3 (minimum) / Ryzen 5 or higher for general computations
- **RAM:** Minimum 8 GB (16 GB recommended for smoother model training and processing)
- **GPU (Optional):** NVIDIA GPU with CUDA support (such as GTX 1060 or higher) for faster deep learning model training. If a GPU is not available, training can still be performed on CPU with increased time.
- **Storage:** At least 500 GB HDD/SSD for storing datasets, project files, and temporary processing data
- **Monitor & Input Devices:** Standard monitor, keyboard, and mouse for user interaction and development

These specifications ensure that the system can preprocess MRI images, train models, and perform predictions efficiently without significant delays.

Software Requirements

The NeuroVision system is implemented using Python and several supporting libraries and modules that facilitate image processing, model development, and evaluation. The software requirements are as follows:

Operating System:

- Windows 10/11 or Linux (Ubuntu preferred) for development and deployment

Programming Language:

- **Python 3.x** is used for all modules due to its flexibility, extensive library support, and ease of integration with machine learning frameworks

Libraries and Frameworks:

1. **TensorFlow:** For building and training deep learning models for tumor classification
2. **Keras:** High-level API for TensorFlow, used for rapid model prototyping and training
3. **PyTorch (optional):** Alternative deep learning framework for experimentation and model implementation
4. **OpenCV:** For image processing tasks such as resizing, normalization, and noise reduction
5. **PIL (Python Imaging Library):** For image loading, manipulation, and preprocessing
6. **NumPy:** For numerical computations, array manipulations, and matrix operations
7. **Pandas:** For dataset management, organization, and analysis
8. **Matplotlib / Seaborn:** For visualization of data distribution, model metrics, and performance graphs
9. **Scikit-learn:** For preprocessing utilities, splitting datasets, evaluation metrics (accuracy, precision, recall), and optional model experiments
10. **OS and Shutil modules:** For file handling, directory management, and batch processing of images

IDE / Editor:

- **Jupyter Notebook:** For interactive development and testing of machine learning models
- **PyCharm or VS Code:** For coding, debugging, and managing the project structure

Summary:

The combination of these hardware and software resources ensures that the NeuroVision system can efficiently preprocess MRI images, train deep learning models, and provide accurate tumor classification results. While a GPU is optional, having one will significantly speed up the training process. The chosen modules and libraries cover all critical tasks, including image handling, model development, data management, and evaluation, allowing the system to operate smoothly and effectively in both development and clinical environments.

3 Design

The design phase of the **NeuroVision Brain Tumor Detection System** focuses on representing the structure and workflow of the system. This includes visualizing data flow and the interactions between the user and the system, ensuring clarity before implementation. The system is primarily designed to classify MRI images into tumor types efficiently and accurately.

3.1 ER Diagram

The **Entity-Relationship (ER) diagram** represents the core entities involved in the NeuroVision Brain Tumor Detection System and the relationships between them. Although no database is used, the ER diagram provides a conceptual understanding of how data and processes interact at runtime.

Entities:

1. **User:** Represents the clinician or researcher interacting with the system.
2. **MRI Image:** Represents the brain scan uploaded by the user for analysis.
3. **Model:** Represents the deep learning model responsible for analyzing MRI images and generating predictions.
4. **Prediction:** Represents the output generated by the model, indicating the type of tumor (No Tumor, Glioma, Meningioma, Pituitary).

Relationships:

1. **User uploads MRI Images:**
    - A **user** can upload **one or many MRI images**.
    - Each **MRI image** is uploaded by **exactly one user**.
    - Relationship type: **1 : Many** (User → MRI Images)

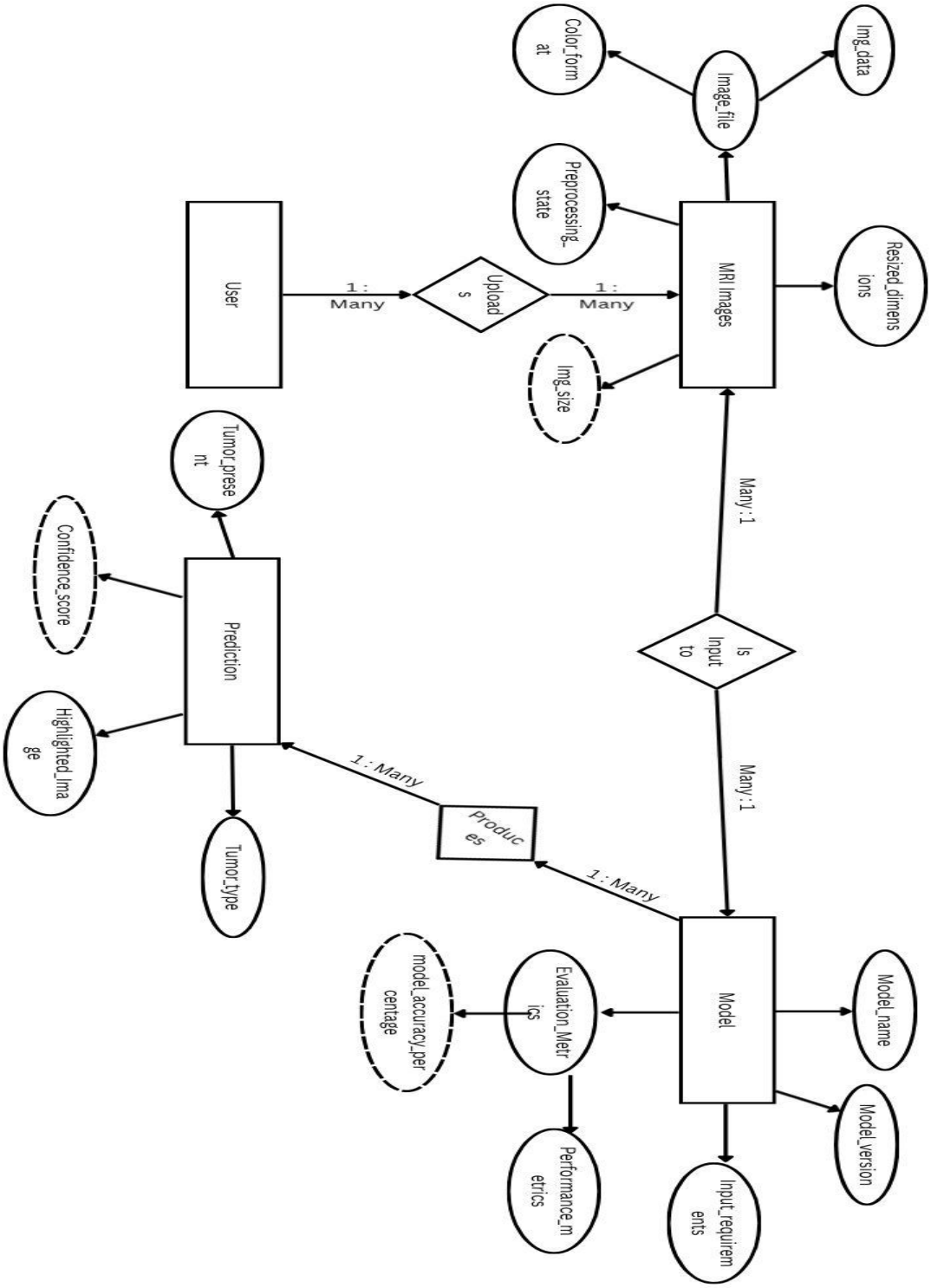
2. **MRI Image is input to Model:**
    - Each **MRI image** serves as input to the **model** for analysis.
    - Each **model** can process **many MRI images**.
    - Relationship type: **Many : 1** (MRI Images → Model)

3. **Model produces Predictions:**
    - The **model** generates **one or many predictions** for MRI images.
    - Each **prediction** is generated by **exactly one model**.
    - Relationship type: **1 : Many** (Model → Predictions)

Summary of ER Flow:

- **User → MRI Image → Model → Prediction**
- Users upload images, which are processed by the model to produce predictions, forming a clear conceptual flow of data.

ER Diagram:


Color_format
Image_file
Img_data
Preprocessing_state
MRI Images
Resized_dimensions
User
1 : Many
Upload s
1 : Many
Img_size
Tumor_present
Confidence_score
Prediction
Highlighted_Image
Tumor_type
Many : 1
Is Input to
Many : 1
1 : Many
Produces
1 : Many
Model
Model_name
Model_version
Evaluation_Metrics
model_accuracy_percentage
Performance_metrics
Input_requirements

3.2 DFD Diagram

The **Data Flow Diagram (DFD)** represents how data moves through the system and how processes interact:

Processes:

1. **Upload MRI Image:** The user uploads a brain scan to the system.
2. **Preprocessing:** The system normalizes, resizes, and prepares the image for analysis.
3. **Tumor Classification:** The deep learning model predicts the tumor type based on the preprocessed image.
4. **Display Result:** The classification result is returned to the user.

External Entities:

- **User:** Clinician or researcher interacting with the system.

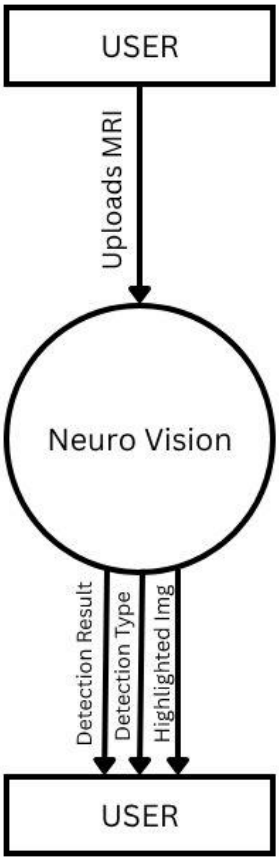
Data Flows:

- MRI Image → Preprocessing → Classification → Display Result

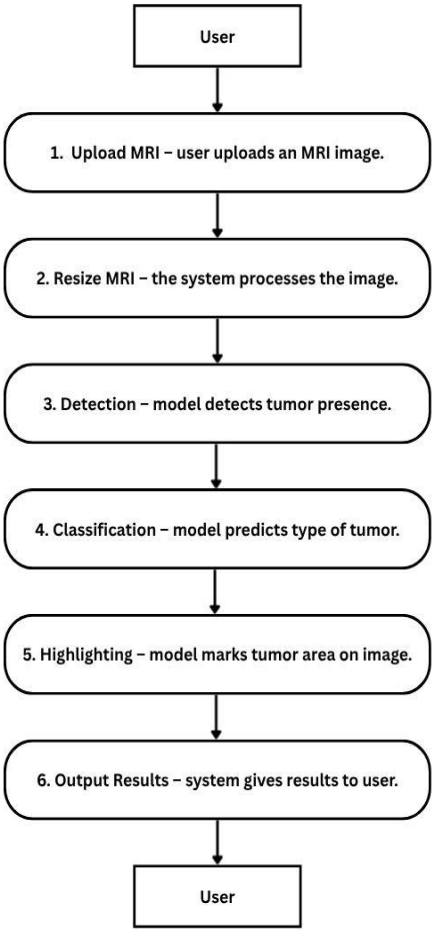
Data Stores:

- No persistent storage is used; all data processing occurs in memory during runtime.

Context Level DFD :

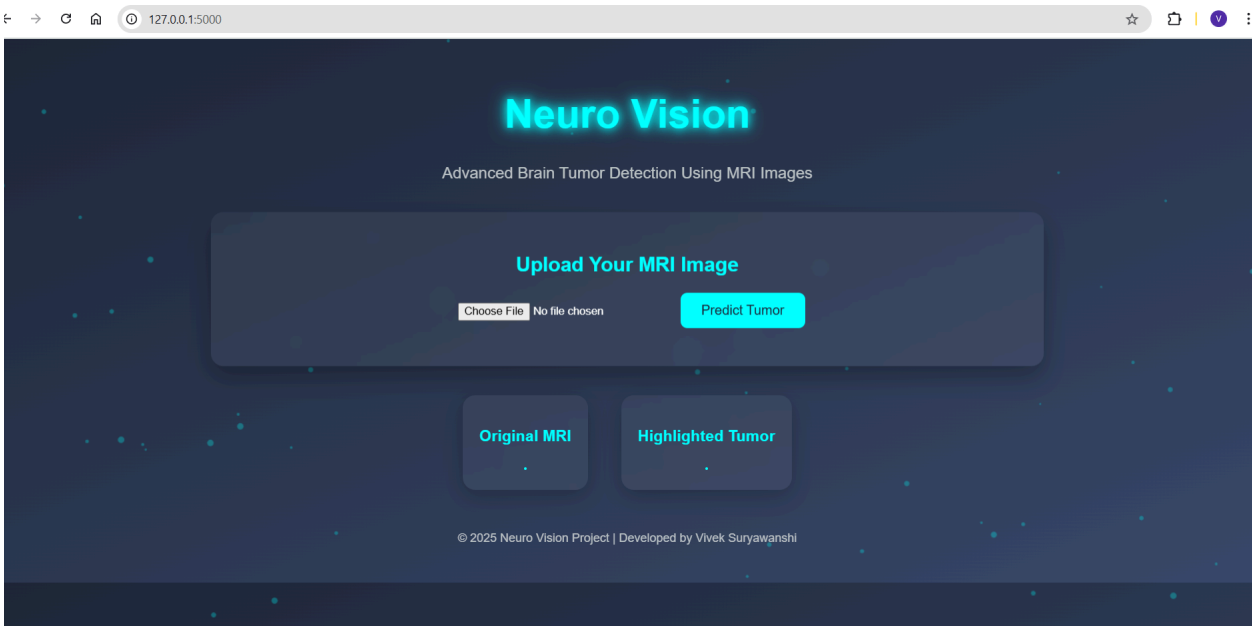


First Level DFD :

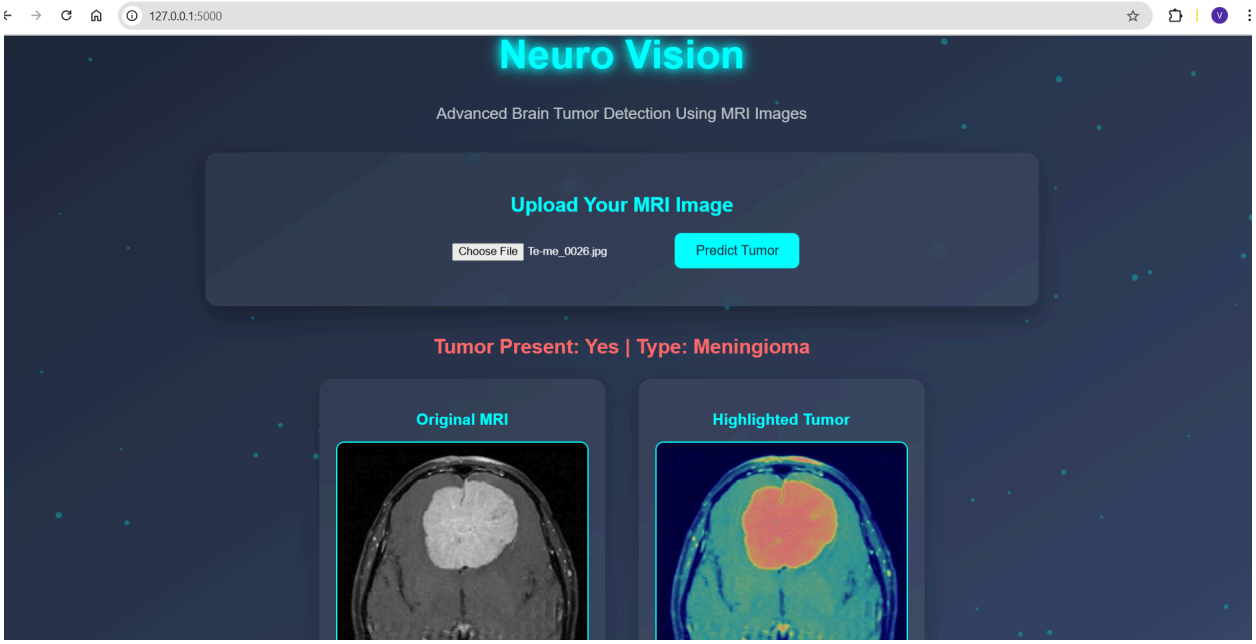


4. Screenshots and Code

Screenshots:



Result:



Code:

Structure:

```
neuro_vision/
│
├── brain_tumor/              # Main Python package
│   ├── __init__.py
│   ├── app.py                # Flask backend (routes and main app)
│   ├── models.py             # CNN model definitions (PyTorch)
│   ├── train.py              # Training scripts for the CNN
│   ├── dataset.py            # Dataset loader class
│   ├── utils.py              # Utility functions (image preprocessing, transforms)
│   ├── templates/            # HTML templates for Flask
│   │   └── index.html        # Home page (image upload)
│   └── static/               # Static files for frontend
│       ├── style.css         # Styling
│       └── script.js         # Frontend JavaScript
│
├── data/                     # Dataset folder
│   ├── training/
│   │   ├── glioma/           # Training images for Glioma
│   │   ├── meningioma/       # Training images for Meningioma
│   │   ├── pituitary/        # Training images for Pituitary tumor
│   │   └── notumor/          # Training images for No Tumor
│   └── testing/
│       ├── glioma/           # Testing images for Glioma
│       ├── meningioma/       # Testing images for Meningioma
│       ├── pituitary/        # Testing images for Pituitary tumor
│       └── notumor/          # Testing images for No Tumor
│
├── models/                   # Trained model weights
│   └── cnn.pth               # Pre-trained CNN model
│
├── requirements.txt          # Python dependencies
└── predict.py                # Main file to run the application
```

➤ **neuro_vision/brain_tumor/app.py** *# Flask backend (routes and main app)*

```
from flask import Flask, request, jsonify, render_template
from werkzeug.utils import secure_filename
import os
from predict import predict

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

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

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

    file = request.files['file']
    filename = secure_filename(file.filename)
    filepath = os.path.join(UPLOAD_FOLDER, filename)
    file.save(filepath)

    tumor_present, tumor_type, highlighted_url = predict(filepath)

    return jsonify({
        'tumor_present': tumor_present,
        'tumor_type': tumor_type,
        'highlighted_image': highlighted_url
    })

if __name__ == '__main__':
    app.run(debug=True)
```

➤ **neuro_vision/brain_tumor/models.py** *# CNN model definitions (PyTorch)*

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Phase1CNN(nn.Module):
    def __init__(self):
        super(Phase1CNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2,2),
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2,2)
        )
        # Compute flattened feature size for 128x128 input
        self.classifier = nn.Sequential(
            nn.Linear(128*16*16, 256),  # 128x128 input -> 16x16 feature map after 3 poolings
            nn.ReLU(),
            nn.Linear(256, 4)  # 4 classes
        )

    def forward(self, x):
        x = self.features(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```

➤ **neuro_vision/brain_tumor/train.py** *# Training scripts for the CNN*

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, random_split
from torchvision import transforms
from brain_tumor.dataset import BrainTumor4ClassDataset
from brain_tumor.models import Phase1CNN
from sklearn.metrics import precision_score, recall_score, f1_score
import os

# Device
device = torch.device("cuda" if torch.cuda.is_available() else "mps" if
torch.backends.mps.is_available() else "cpu")
print(f"Using device: {device}")

# Hyperparameters
batch_size = 32
epochs = 10
lr = 1e-3

# Transform (resizes all images to 128x128)
transform = transforms.Compose([
    transforms.Resize((128,128)),
    transforms.ToTensor()
])

# Dataset
dataset = BrainTumor4ClassDataset(root_dir='data/Training', transform=transform)
train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size
train_dataset, val_dataset = random_split(dataset, [train_size, val_size])
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

# Model, criterion, optimizer
model = Phase1CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=lr)

# Training loop
for epoch in range(epochs):
    model.train()
```

```
    running_loss = 0.0
    correct = 0
    total = 0
    for imgs, labels in train_loader:
        imgs, labels = imgs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(imgs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * imgs.size(0)
        _, predicted = torch.max(outputs.data,1)
        total += labels.size(0)
        correct += (predicted==labels).sum().item()
    train_loss = running_loss / total
    train_acc = correct / total

    # Validation
    model.eval()
    val_loss = 0.0
    val_correct = 0
    val_total = 0
    all_preds, all_labels = [], []
    with torch.no_grad():
        for imgs, labels in val_loader:
            imgs, labels = imgs.to(device), labels.to(device)
            outputs = model(imgs)
            loss = criterion(outputs, labels)
            val_loss += loss.item() * imgs.size(0)
            _, predicted = torch.max(outputs.data,1)
            val_total += labels.size(0)
            val_correct += (predicted==labels).sum().item()
            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())
    val_loss /= val_total
    val_acc = val_correct / val_total
    precision = precision_score(all_labels, all_preds, average='macro')
    recall = recall_score(all_labels, all_preds, average='macro')
    f1 = f1_score(all_labels, all_preds, average='macro')

    print(f"Epoch {epoch+1}/{epochs} - Train Loss: {train_loss:.4f}, Val Loss: {val_loss:.4f}, "
          f"Train Acc: {train_acc:.4f}, Val Acc: {val_acc:.4f}, Precision: {precision:.4f}, Recall:
{recall:.4f}, F1: {f1:.4f}")
```

```
# Save model
os.makedirs('models', exist_ok=True)
torch.save(model.state_dict(), 'models/phase1cnn.pth')
print("Model saved to models/phase1cnn.pth")
```

➤ **neuro_vision/brain_tumor/dataset.py** *# Dataset loader class*

```
import os
from PIL import Image
import torch
from torch.utils.data import Dataset

class BrainTumor4ClassDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        self.image_paths = []
        self.labels = []
        self.class_map = {'Glioma':0, 'Meningioma':1, 'Pituitary':2, 'NoTumor':3}

        for class_name, label in self.class_map.items():
            class_dir = os.path.join(root_dir, class_name)
            if not os.path.exists(class_dir):
                continue
            for filename in os.listdir(class_dir):
                if filename.endswith(('.png','.jpg','.jpeg')):
                    self.image_paths.append(os.path.join(class_dir, filename))
                    self.labels.append(label)

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        image = Image.open(img_path).convert('RGB')
        if self.transform:
            image = self.transform(image)
        label = torch.tensor(self.labels[idx])
        return image, label
```

➤ **neuro_vision/brain_tumor/utils.py** *# Utility functions (image preprocessing, transforms)*

```
import torch
import torch.nn.functional as F
import numpy as np
import cv2
from torchvision import transforms
from PIL import Image

class GradCAM:
    def __init__(self, model, target_layer):
        self.model = model
        self.model.eval()
        self.target_layer = target_layer
        self.gradients = None
        self.activations = None
        self.hook_layers()

    def hook_layers(self):
        def forward_hook(module, input, output):
            self.activations = output.detach()
        def backward_hook(module, grad_in, grad_out):
            self.gradients = grad_out[0].detach()
        self.target_layer.register_forward_hook(forward_hook)
        self.target_layer.register_backward_hook(backward_hook)

    def generate(self, input_tensor, class_idx=None):
        self.model.zero_grad()
        output = self.model(input_tensor)
        if class_idx is None:
            class_idx = output.argmax().item()
        target = output[0, class_idx]
        target.backward()
        weights = self.gradients.mean(dim=(2,3), keepdim=True)
        cam = F.relu((weights * self.activations).sum(1, keepdim=True))
        cam = cam.squeeze().cpu().numpy()
        cam = cv2.resize(cam, (input_tensor.size(3), input_tensor.size(2)))
        cam = (cam - cam.min()) / (cam.max() - cam.min() + 1e-8)
        return cam

def overlay_cam_on_image(img_path, cam):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
heatmap = cv2.applyColorMap(np.uint8(255*cam), cv2.COLORMAP_JET)
heatmap = cv2.cvtColor(heatmap, cv2.COLOR_BGR2RGB)
overlay = heatmap * 0.4 + img * 0.6
overlay = np.uint8(overlay)
return overlay
```

➤ **neuro_vision/predict.py** *# Main file to run the application*

```
import torch
from torchvision import transforms
from PIL import Image
import numpy as np
import cv2
import os
from brain_tumor.models import Phase1CNN

# Device
device = torch.device("cuda" if torch.cuda.is_available() else
                      "mps" if torch.backends.mps.is_available() else "cpu")

# Load trained model
model = Phase1CNN()
model.load_state_dict(torch.load("models/phase1cnn.pth", map_location=device))
model.to(device)
model.eval()

# Preprocessing transform
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
])

def predict(image_path):
    # Load image
    img = Image.open(image_path).convert("RGB")
    img_tensor = transform(img).unsqueeze(0).to(device)
    img_tensor.requires_grad = True

    # Forward pass
    outputs = model(img_tensor)
    pred_class_idx = torch.argmax(outputs, dim=1).item()
    classes = ['Glioma', 'Meningioma', 'Pituitary', 'NoTumor']
    pred_class = classes[pred_class_idx]
    tumor_present = "No" if pred_class == "NoTumor" else "Yes"

    # Grad-CAM
    model.zero_grad()
    outputs[0, pred_class_idx].backward(retain_graph=True)
    gradients = img_tensor.grad.detach().cpu().numpy()[0]
    activations = img_tensor.detach().cpu().numpy()[0]
```

```
weights = np.mean(gradients, axis=(1, 2))
cam = np.zeros(activations.shape[1:], dtype=np.float32)
for i, w in enumerate(weights):
    cam += w * activations[i]

cam = np.maximum(cam, 0)
cam = cv2.resize(cam, (img.width, img.height))
cam = cam - np.min(cam)
cam = cam / (np.max(cam) + 1e-8)

# Create RGB heatmap overlay
heatmap = cv2.applyColorMap(np.uint8(255 * cam), cv2.COLORMAP_JET)
heatmap = cv2.cvtColor(heatmap, cv2.COLOR_BGR2RGB)
original = np.array(img)
overlay = np.uint8(0.6 * original + 0.4 * heatmap)

# Save highlighted image in static folder so browser can access it
save_dir = 'brain_tumor/static/highlighted'
os.makedirs(save_dir, exist_ok=True)
base_name = os.path.basename(image_path)
highlighted_filename = f"highlighted_{base_name}"
highlighted_path = os.path.join(save_dir, highlighted_filename)
Image.fromarray(overlay).save(highlighted_path)

# Return URL relative to static folder
highlighted_url = 'static/highlighted/' + highlighted_filename
return tumor_present, pred_class, highlighted_url
```

➤ **neuro_vision/brain_tumor/templates/index.html** *# Home page (image upload)*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Neuro Vision - Brain Tumor Detection</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>

<body>
    <!-- Neural Network Background -->
    <canvas id="neuron-canvas"></canvas>

    <!-- Full-screen Processing Overlay -->
    <div id="overlay">
        <div class="overlay-content">
            <h2>Analyzing MRI...</h2>
            <div class="loader"></div>
        </div>
    </div>

    <div class="container">
        <!-- Header -->
        <header>
            <h1 class="animated-text">Neuro Vision</h1>
            <p class="subtitle">Advanced Brain Tumor Detection Using MRI Images</p>
        </header>

        <!-- Upload Section -->
        <div class="upload-card">
            <h2>Upload Your MRI Image</h2>
            <input type="file" id="fileInput" accept="image/*">
            <button onclick="upload()">Predict Tumor</button>
        </div>

        <!-- Prediction Result -->
        <div id="result-section">
            <h2 id="result"></h2>
        </div>

        <!-- Images Section -->
        <div id="images">
```

```
            <div class="image-container">
                <h3>Original MRI</h3>
                <img id="original" src="" alt="">
            </div>
            <div class="image-container">
                <h3>Highlighted Tumor</h3>
                <img id="highlighted" src="" alt="">
            </div>
        </div>

        <!-- Footer -->
        <footer>
            <p>© 2025 Neuro Vision Project | Developed by Vivek Suryawanshi</p>
        </footer>
    </div>

    <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>

</html>
```

➤ **neuro_vision/brain_tumor/static/css/style.css** *# Styling*

```
/* Body + Dark Background */
body {
    margin: 0;
    font-family: 'Arial', sans-serif;
    background: linear-gradient(to bottom right, #1f2739, #3a4a6b);
    color: #fff;
    overflow-x: hidden;
    position: relative;
}

/* Container */
.container {
    max-width: 1000px;
    margin: auto;
    padding: 30px;
    position: relative;
    z-index:1;
}

/* Header */
header {
    text-align: center;
    margin-bottom: 40px;
}

header h1 {
    font-size: 3rem;
    color: #00ffff;
    animation: glow 2s ease-in-out infinite alternate;
}

header .subtitle {
    font-size: 1.2rem;
    color: #bdc3c7;
}

@keyframes glow {
    from { text-shadow: 0 0 10px #00ffff, 0 0 20px #00ffff; }
    to { text-shadow: 0 0 20px #00ffff, 0 0 40px #00ffff; }
}

/* Upload Card */
```

```
.upload-card {
    background: rgba(255,255,255,0.05);
    backdrop-filter: blur(10px);
    padding: 30px;
    border-radius: 15px;
    text-align: center;
    margin-bottom: 30px;
    transition: transform 0.3s, box-shadow 0.3s;
    box-shadow: 0 8px 25px rgba(0,0,0,0.2);
}

.upload-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 15px 30px rgba(0,0,0,0.3);
}

.upload-card h2 {
    margin-bottom: 20px;
    color: #00ffff;
}

.upload-card input[type="file"] {
    padding: 10px;
    margin-bottom: 15px;
    border-radius: 8px;
}

.upload-card button {
    padding: 12px 25px;
    background-color: #00ffff;
    color: #1f2739;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-size: 16px;
    transition: 0.3s;
}

.upload-card button:hover {
    background-color: #00e0e0;
}

/* Images Section */
#images {
```

```
    display: flex;
    justify-content: center;
    gap: 40px;
    flex-wrap: wrap;
}

.image-container {
    text-align: center;
    background: rgba(255,255,255,0.05);
    backdrop-filter: blur(6px);
    padding: 20px;
    border-radius: 15px;
    box-shadow: 0 8px 25px rgba(0,0,0,0.2);
    transition: transform 0.3s;
}

.image-container:hover {
    transform: scale(1.03);
}

.image-container h3 {
    margin-bottom: 15px;
    color: #00ffff;
}

.image-container img {
    max-width: 300px;
    border-radius: 10px;
    border: 2px solid #00ffff;
    animation: fadeIn 0.5s ease-in-out;
}

/* Result Section */
#result-section {
    text-align: center;
    margin: 25px 0;
}

#result {
    font-size: 1.5rem;
    font-weight: bold;
    color: #ff6b6b;
    animation: fadeIn 1s ease-in-out;
}
```

```
/* Footer */
footer {
    text-align: center;
    margin-top: 50px;
    color: #bdc3c7;
    font-size: 0.9rem;
}

/* Overlay */
#overlay {
    position: fixed;
    top:0;
    left:0;
    width:100%;
    height:100%;
    background: rgba(0,0,0,0.7);
    backdrop-filter: blur(8px);
    display: none;  /* hidden by default */
    align-items: center;
    justify-content: center;
    z-index: 9999;
}

.overlay-content {
    text-align: center;
    color: #00ffff;
}

.loader {
    border: 5px solid #f3f3f3;
    border-top: 5px solid #00ffff;
    border-radius: 50%;
    width: 50px;
    height: 50px;
    margin: 20px auto 0;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg);}
    100% { transform: rotate(360deg);}
}
```

```
/* Neuron canvas */
#neuron-canvas {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: 0;
}

/* Fade-in */
@keyframes fadeIn {
    from { opacity: 0;}
    to { opacity: 1;}
}
```

➤ **neuro_vision/brain_tumor/static/script.js** *# Frontend JavaScript*

```
// Neural network canvas animation
const canvas = document.getElementById("neuron-canvas");
const ctx = canvas.getContext("2d");
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;

const neurons = [];
for (let i = 0; i < 60; i++) {
  neurons.push({
    x: Math.random() * canvas.width,
    y: Math.random() * canvas.height,
    radius: Math.random() * 3 + 1,
    dx: (Math.random() - 0.5) * 0.5,
    dy: (Math.random() - 0.5) * 0.5,
  });
}

function animateNeurons() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  for (let n of neurons) {
    n.x += n.dx;
    n.y += n.dy;
    if (n.x < 0 || n.x > canvas.width) n.dx *= -1;
    if (n.y < 0 || n.y > canvas.height) n.dy *= -1;
    ctx.beginPath();
    ctx.arc(n.x, n.y, n.radius, 0, Math.PI * 2);
    ctx.fillStyle = "rgba(0,255,255,0.3)";
    ctx.fill();
  }
  requestAnimationFrame(animateNeurons);
}
animateNeurons();

// Function to show in-page warning
function showWarning(message) {
  let warningDiv = document.getElementById("warning");
  if (!warningDiv) {
    warningDiv = document.createElement("div");
    warningDiv.id = "warning";
    warningDiv.style.color = "#ff4d4d";
    warningDiv.style.fontWeight = "bold";
    warningDiv.style.marginTop = "10px";
```

```
    warningDiv.style.opacity = 0;
    warningDiv.style.transition = "opacity 0.5s";
    document.querySelector(".upload-card").appendChild(warningDiv);
  }
  warningDiv.innerText = message;
  warningDiv.style.opacity = 1;
  setTimeout(() => {
    warningDiv.style.opacity = 0;
  }, 3000);
}

// Upload & Predict function
function upload() {
  const fileInput = document.getElementById("fileInput");
  const file = fileInput.files[0];

  if (!file) {
    showWarning("Please select an MRI image to analyze – Neuro Vision");
    return; // Stop execution
  }

  // Show processing overlay
  const overlay = document.getElementById("overlay");
  overlay.style.display = "flex";

  const formData = new FormData();
  formData.append("file", file);

  // Display original MRI immediately
  document.getElementById("original").src = URL.createObjectURL(file);

  fetch("/predict", { method: "POST", body: formData })
    .then((resp) => resp.json())
    .then((data) => {
      // Display highlighted tumor image
      document.getElementById("highlighted").src =
        data.highlighted_image + "?t=" + new Date().getTime();

      // Show result text after short delay for smooth transition
      setTimeout(() => {
        document.getElementById("result").innerText =
          "Tumor Present: " +
          data.tumor_present +
          " | Type: " +
```

```
          data.tumor_type;
        overlay.style.display = "none"; // Hide overlay
      }, 500);
    })
    .catch((err) => {
      overlay.style.display = "none";
      console.error(err);
      showWarning("Error analyzing the image – Neuro Vision");
    });
}

// Resize canvas on window resize
window.addEventListener("resize", () => {
  canvas.width = window.innerWidth;
  canvas.height = window.innerHeight;
});
```

➤ **neuro_vision/requirements.txt** *# Python dependencies*

Flask
torch
torchvision
Pillow
opencv-python
numpy

5. Limitations

While the **NeuroVision Brain Tumor Detection System** provides an automated and efficient solution for classifying brain tumors using MRI images, there are certain limitations that need to be acknowledged. These limitations primarily relate to the scope, technical constraints, and practical applicability of the current system.

1. **Limited Tumor Types:**
   The system currently classifies MRI scans into only four categories: No Tumor, Glioma, Meningioma, and Pituitary Tumor. It does not identify other rare or complex tumor types, limiting its use in broader clinical scenarios.

2. **No Tumor Region Highlighting:**
   The system provides the classification result but does not highlight the exact location or region of the tumor within the MRI scan. This limits the ability of clinicians to visually interpret affected areas directly from the system.

3. **No Malignancy or Severity Assessment:**
   The current implementation does not predict whether a tumor is benign or malignant, nor does it provide information on tumor size, stage, or severity. Such information is critical for treatment planning.

4. **Dependence on Image Quality:**
   The accuracy of classification depends heavily on the quality and resolution of the MRI scans. Poorly captured, noisy, or inconsistent images may reduce the model's performance and reliability.

5. **No Database Integration:**
   As the system processes images on-the-fly without a database, it does not store patient data or past results. This means longitudinal tracking, historical comparisons, or large-scale data analysis is not possible.

6. **Hardware and Processing Limitations:**
   While GPU acceleration is optional, training the deep learning model on a CPU may be time-consuming. Users without high-performance computing resources may face slower processing times during model training.

7. **Model Generalization:**
   The system is trained on specific datasets and may not generalize well to all types of MRI machines, protocols, or populations. Variations in scanning parameters or patient demographics could affect accuracy.

8. **No Clinical Decision Support:**
   The system is designed purely for classification and does not provide treatment

suggestions or medical advice. Final diagnostic and treatment decisions must always be made by qualified clinicians.

Summary:

Despite its capabilities in automated brain tumor classification, NeuroVision has limitations related to scope, image analysis, and clinical utility. Future enhancements could address these limitations, such as adding tumor region highlighting, expanding tumor types, integrating databases, or including malignancy and severity prediction.

6. Future Enhancements

While the **NeuroVision Brain Tumor Detection System** provides an effective solution for automated classification of brain tumors from MRI scans, there are several opportunities to enhance its functionality, usability, and clinical relevance in the future. These enhancements aim to overcome current limitations and expand the system's capabilities to support advanced diagnostic processes.

1. **Tumor Region Highlighting:**
   One of the primary future improvements would be incorporating **tumor segmentation**, which allows the system to highlight the exact location and boundaries of the tumor within the MRI scan. This will help clinicians visually interpret affected regions, facilitating better treatment planning and reducing diagnostic ambiguity.

2. **Malignancy and Severity Prediction:**
   Extending the system to predict whether a tumor is **benign or malignant**, and estimating its **size, stage, and severity**, would significantly enhance clinical utility. Such predictions can provide critical information for treatment planning and patient management.

3. **Support for Additional Tumor Types:**
   Expanding the dataset and training the model on **more tumor categories** can increase the system's applicability in diverse clinical scenarios. This includes rare tumors or mixed-type tumors, improving generalization across a broader range of cases.

4. **Database Integration:**
   Incorporating a **secure database** to store MRI scans, predictions, and patient information can facilitate **longitudinal analysis**, historical comparisons, and model retraining using real-world data. This would also allow better tracking of patient progress and trends over time.

5. **Multi-Modality Imaging Support:**
   Future versions of the system could integrate data from multiple imaging modalities, such as **CT scans, PET scans, or fMRI**, to improve classification accuracy and provide a more comprehensive diagnostic tool.

6. **Web or Cloud-Based Deployment:**
   Developing a **cloud-based or web-accessible version** of NeuroVision would make it easier for hospitals and research centers to use the system without requiring high-end local hardware. It would also enable remote access for clinicians in resource-limited areas.

7. **Enhanced User Interface:**
   Improving the interface to include features like **interactive image viewing, annotation**

tools, and confidence score visualization can make the system more user-friendly and informative for medical professionals.

8. **Integration with Clinical Decision Support Systems:**
   Linking the system with **hospital information systems (HIS) or electronic health records (EHR)** can streamline workflows and allow clinicians to access classification results alongside other patient data for more informed decision-making.

9. **Continuous Model Improvement:**
   Implementing **incremental learning or model retraining** with new patient data can improve accuracy over time and adapt the system to evolving clinical standards and imaging technologies.

10. **Mobile Application Development:**
    In the long term, a **mobile version** could allow clinicians to quickly upload images and view predictions on tablets or smartphones, providing greater flexibility and accessibility in clinical environments.

Summary:
Future enhancements aim to make NeuroVision not only more accurate and robust but also more clinically valuable and user-friendly. By adding tumor segmentation, malignancy prediction, database integration, and multi-platform accessibility, the system can evolve into a comprehensive diagnostic support tool, bridging the gap between artificial intelligence and real-world medical decision-making.

7. References and Bibliography

1. **Menze, B. H., Jakab, A., Bauer, S., et al. (2015).** *The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS).* IEEE Transactions on Medical Imaging, 34(10), 1993–2024. https://doi.org/10.1109/TMI.2014.2377694

2. **Chakrabarty, S., & Sarkar, R. (2020).** *Brain Tumor Detection and Classification from MRI Images using Deep Learning Techniques.* International Journal of Advanced Computer Science and Applications, 11(6), 230–237.

3. **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012).** *ImageNet Classification with Deep Convolutional Neural Networks.* Advances in Neural Information Processing Systems (NeurIPS), 25, 1097–1105.

4. **Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018).** *UNet++: A Nested U-Net Architecture for Medical Image Segmentation.* Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 3–11.

5. **Python Software Foundation. (2023).** *Python Language Reference, version 3.x.* https://www.python.org/

6. **TensorFlow. (2023).** *TensorFlow: An end-to-end open source machine learning platform.* https://www.tensorflow.org/

7. **Keras Documentation. (2023).** *Keras: The Python Deep Learning API.* https://keras.io/

8. **OpenCV Library. (2023).** *Open Source Computer Vision Library.* https://opencv.org/

9. **Pillow (PIL Fork). (2023).** *Python Imaging Library (Pillow).* https://python-pillow.org/

10. **Scikit-learn. (2023).** *Machine Learning in Python.* https://scikit-learn.org/

11. **NumPy Developers. (2023).** *NumPy: Array programming with Python.* https://numpy.org/

12. **Matplotlib Developers. (2023).** *Matplotlib: Visualization with Python.* https://matplotlib.org/

13. **Seaborn Developers. (2023).** *Seaborn: Statistical Data Visualization.* https://seaborn.pydata.org/
14. **Parvar, M. N. (2023).** *Brain Tumor MRI Dataset.* Kaggle. https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset