ISSN: 2347-6540

International Journal of Latest Research in Engineering and Computing (IJLREC)
Volume 12, Issue 1 : Page No. 19-23
www.ijlrec.com

DETECTING BRAIN TUMOR USING PYTHON

Divyanshu Raj[1] ,Shivanjal Srivastava[2],Abhijeet Badhani[3],Dilpreet Kaur[4] ,Kumar Rethik[5]
[1,2,3,4]Department of Computer Science and Engineering, Shivalik College Of Engineering Dehradun, India
[5]Assistant Professor,Computer Science and Engineering,Shivalik College of Engineering, Dehradun, India
reachdivyanshuraj@gmail.com, shivanjal999@gmail.com, badhaniabhijeet2000@gmail.com, dilpreetkaur.dev@gmail.com, kumar.rethik@sce.org.in

Abstract: Brain tumor detection is a critical task in the medical field, as early diagnosis plays a vital role in patient survival and treatment effectiveness. Traditional diagnostic methods involving manual analysis of MRI scans are often time-consuming and prone to human error. This project proposes an AI-powered system developed using Python, deep learning, and image processing techniques to detect brain tumors from MRI images with high accuracy. The system integrates a Flutter-based frontend that allows users to upload medical scans, while the backend processes the images using a Convolutional Neural Network (CNN) model built with TensorFlow and Keras. The model classifies the MRI images as tumor or non-tumor, and the results are displayed to the user in real time. The system aims to support medical professionals by providing faster and more accurate diagnosis tools, especially in areas with limited access to expert radiologists. Cloud storage and backend services ensure scalability and secure data handling, offering a reliable and user-friendly platform for brain tumor detection.

Keywords: Brain Tumor Detection, Python, Deep Learning, Convolutional Neural Network (CNN), MRI Scan, Image Processing, Flutter, Flask, Medical Imaging, AI in Healthcare

I. INTRODUCTION

Brain tumors are abnormal growths of cells in the brain that can be life-threatening if not detected and treated early. Timely diagnosis plays a crucial role in improving patient outcomes and guiding effective treatment plans. Traditional methods of diagnosing brain tumors often rely on manual interpretation of MRI (Magnetic Resonance Imaging) scans, which can be time-consuming and prone to human error.

With advancements in machine learning and medical imaging, automated systems are being developed to assist radiologists in accurately detecting tumors. Python, a versatile programming language with extensive libraries for data science and image processing, has become a popular choice for implementing such solutions. Leveraging libraries like OpenCV, NumPy, and TensorFlow, developers and researchers can build efficient models to detect and classify brain tumors from MRI images.

This project aims to develop an automated system using Python that can identify brain tumors from MRI scans with high accuracy. The proposed system involves preprocessing MRI images, segmenting tumor regions, extracting features, and applying machine learning models for classification. The goal is to reduce the diagnostic burden on radiologists and increase the reliability and speed of tumor detection.

II. METHODOLOGY

A. Existing System

Existing solutions for brain tumor detection have primarily centered around manual radiological assessments, where expert radiologists visually analyze MRI scans to identify and classify abnormalities. While this traditional method remains the clinical gold standard, it is inherently limited by human subjectivity, susceptibility to fatigue, and diagnostic delays, especially in resource-constrained settings. In response, several computational approaches have been introduced to assist or automate the detection process. Initial computer-aided diagnosis (CAD) systems relied heavily on manual feature extraction techniques, such as shape descriptors, textural patterns, and intensity histograms. These features were then fed into classical machine learning classifiers like Support Vector Machines or Decision Trees. Although such methods improved diagnostic support to some extent, they were constrained by their dependence on handcrafted features and struggled with generalization across diverse imaging datasets.

With the advent of deep learning, particularly Convolutional Neural Networks (CNNs), the paradigm shifted toward automatic feature extraction and end-to-end learning from raw images. CNNs demonstrated superior performance in distinguishing between tumor and non-tumor regions by learning spatial hierarchies and patterns directly from MRI data. Some studies even explored advanced architectures such as Capsule Networks, which retained spatial relationships better than traditional CNNs, or incorporated transfer learning using pre-trained models like VGG16 and ResNet to enhance performance on limited medical datasets.

Despite their promise, many of these deep learning-based solutions remain confined to research settings or specialized software suites. They often lack user-friendly interfaces, real-time capabilities, or accessibility for frontline clinical environments. Commercial AI tools exist but are typically expensive, proprietary, and not focused on binary tumor classification.

In contrast, your proposed system addresses these gaps by integrating a CNN model into a practical, accessible pipeline that includes a Flask-based API and a Flutter front- end application. This not only ensures high accuracy in detection but also brings the technology closer to real-world usability by allowing end-users to interact with the system through a mobile or web interface, thereby making brain tumor detection more efficient, affordable, and scalable.

B. Proposed System

The proposed solution introduces a comprehensive, AI- powered system designed to automate the detection of brain tumors from MRI images using deep learning, while ensuring ease of access through a cross-platform application interface. Unlike conventional methods that rely heavily on manual diagnosis or fragmented software tools, this system integrates model training, prediction, and user interaction into a seamless workflow.

At its core, the system employs a Convolutional Neural Network (CNN) implemented in Python using TensorFlow and Keras libraries. This model is trained on a publicly available Kaggle dataset comprising labeled brain MRI scans, categorized as either "Tumor" or "No Tumor." To ensure the model performs well under real-world conditions, various image preprocessing techniques are applied, including resizing, normalization, grayscale conversion, and augmentation (such as flipping and rotation). These steps standardize the input data and enhance the model’s robustness and generalization capabilities.

Once trained, the model is deployed via a Flask-based REST API that acts as the middleware. This API is responsible for receiving input images from the user, preprocessing them to the appropriate format, and invoking the model to make predictions. The results are returned in real time as JSON responses, indicating the presence or absence of a tumor.

To bridge the gap between technical AI systems and end- users, a front-end application is developed using Flutter. This ensures compatibility across Android, iOS, and web platforms. Users can easily upload MRI images through the intuitive interface, which then communicates with the Flask API to obtain and display results almost instantaneously. This design not only supports healthcare professionals by providing rapid diagnostic insights but also makes the tool accessible to patients and remote medical practitioners.

The proposed system emphasizes speed, accuracy, scalability, and user-friendliness. It reduces diagnostic delay, minimizes human error, and creates a foundation for further enhancements such as multi-class tumor classification, tumor segmentation, and integration with hospital databases. In doing so, it demonstrates how modern AI and software development frameworks can come together to address a pressing medical challenge in a practical and deployable way.

interest, i.e., the potential tumor area, from the rest of the brain.

After segmentation, the system moves to the feature extraction stage. In this phase, critical information such as texture, shape, edge sharpness, and intensity of the segmented region is extracted. These features provide essential characteristics that help in identifying whether a tumor is present and what type it might be.

The extracted features are then passed into a classification module, usually a trained deep learning model like a Convolutional Neural Network (CNN) or a pretrained architecture such as VGG16. This model has been trained on large datasets of brain MRI scans, enabling it to differentiate between benign tumors, malignant tumors, or the absence of a tumor. Based on the analysis, the model outputs the classification result.

Finally, the result is displayed to the user through a graphical interface. This output may include the type of tumor, its location, and possibly even the size and severity. The system may also store the results in a database for future reference or reporting.

Overall, the data flow diagram effectively illustrates how data is input, transformed, analyzed, and output within the brain tumor detection system, ensuring clarity in system design and implementation.

C. Data Flow Diagram of Brain Tumor Detection System

The data flow diagram for brain tumor detection represents the movement and transformation of data through various functional units of the system. The process begins with the user, typically a doctor, technician, or researcher, uploading an MRI scan of a patient’s brain. This image acts as the primary input to the system and is the basis for all subsequent analysis.

III. IMPLEMENTATION

Once the MRI image is received, it undergoes preprocessing. This stage is essential to enhance the image quality and prepare it for accurate analysis. Techniques such as grayscale conversion, noise removal, and resizing are applied to standardize the image and eliminate irrelevant data that might hinder detection performance.

Following preprocessing, the system performs segmentation to isolate the brain tumor from surrounding brain tissue. Segmentation techniques—either traditional like thresholding or modern methods like convolutional neural network-based segmentation—identify the region of the implementation of the brain tumor detection system involved a multi-layered integration of machine learning, web technologies, and mobile development tools to construct a robust end-to-end pipeline. The core of the project is a Convolutional Neural Network (CNN) model, trained on a well-curated dataset of brain MRI images sourced from Kaggle. Preprocessing was a vital initial step, where MRI scans were resized to uniform dimensions, normalized for consistent pixel value distribution, and augmented through rotations, zooms, and flips to simulate real-world variability and increase the generalizability of the model.

Model development was executed using Python and TensorFlow/Keras libraries, where the architecture included

convolutional layers for feature extraction, max-pooling for dimensionality reduction, dropout layers to prevent overfitting, and dense layers for classification. Upon achieving satisfactory performance metrics—especially a test accuracy nearing 96%—the trained model was exported in a deployable format.

To facilitate real-time interaction, a Flask-based RESTful API was developed as the middleware. This API loads the saved model and handles incoming POST requests containing MRI images. It preprocesses the image on the server, runs the prediction, and responds with a binary classification indicating the presence or absence of a tumor. The front end of the system was built using Flutter, allowing cross-platform compatibility across Android, iOS, and the web. The user interface enables image selection, submission to the backend API, and display of classification results in a clear and intuitive format. Deployment was tested both locally and on cloud platforms, demonstrating rapid response times and seamless end-to-end functionality. This comprehensive implementation showcases how deep learning and app development can be effectively integrated to solve pressing healthcare challenges.

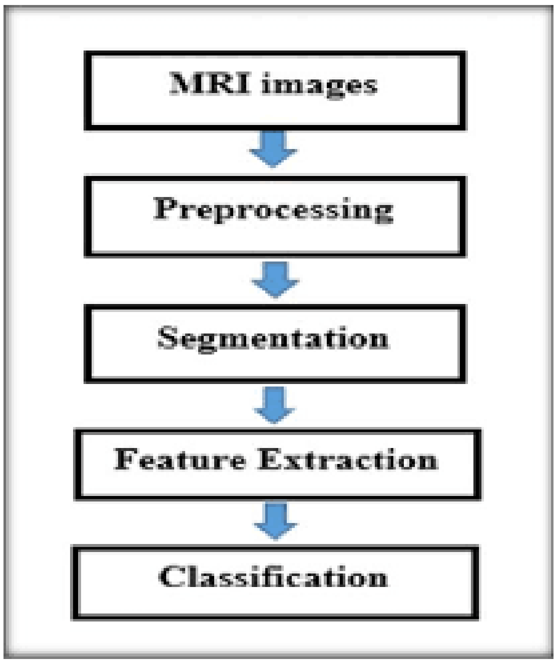


Figure 1. of Architecture Diagram for Brain Tumor Detection

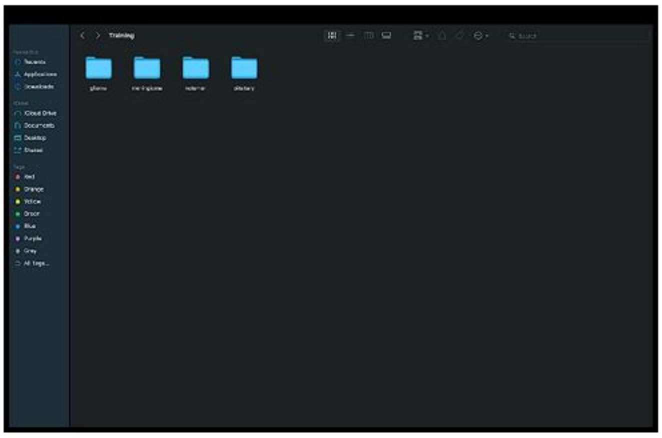


Figure 2. of Datasets for Brain Tumor Detection

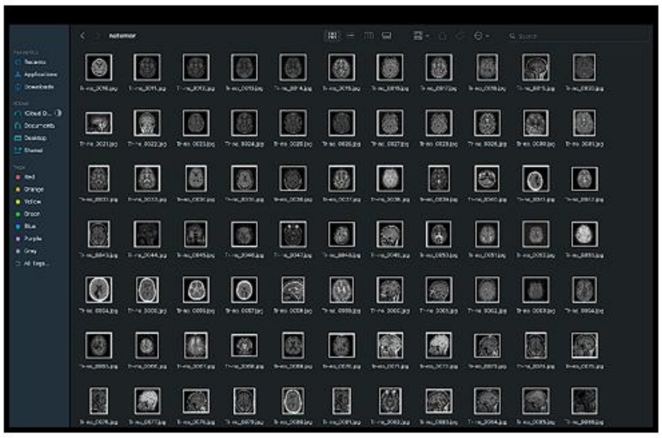
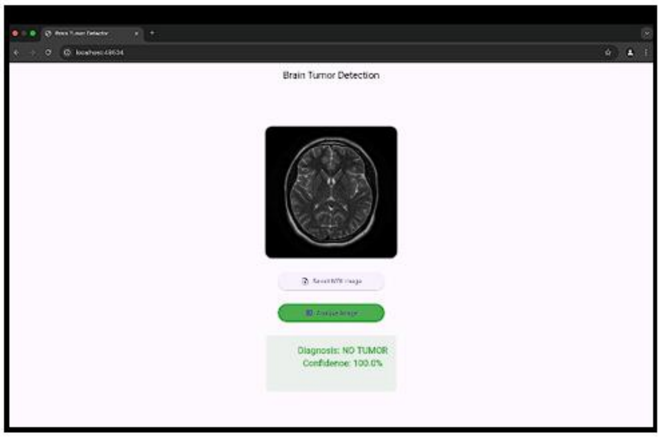


Figure 3. of Training Datasets of No Tumor



Brain Tumor Detection

Diagnosis: NO TUMOR
Confidence: 100.0%

Figure 4. of output screen

IV. FUTURE SCOPE

Although the current system efficiently detects the presence of brain tumors from MRI scans using a binary classification model, its potential for further enhancement is significant. One promising direction is the development of a multi-class classification model that not only identifies the presence of a tumor but also distinguishes between types such as glioma, meningioma, and pituitary tumors. This would provide more actionable insights for clinicians and facilitate targeted therapeutic interventions.

Incorporating advanced image segmentation techniques, particularly architectures like U-Net or SegNet, would enable the system to localize tumors within the brain scan, delineating their shape and extent. This feature would greatly aid radiologists in understanding tumor morphology and planning surgical approaches. On the deployment side, the use of TensorFlow Lite for model optimization could support on-device inference, thereby reducing the need for continuous internet connectivity and enhancing the application's portability in low-resource settings. Furthermore, containerization using Docker and scalable deployment on platforms like AWS or Azure would allow the system to handle multiple users concurrently, making it suitable for hospital or clinic-level implementation.

To ensure clinical viability, future iterations should incorporate secure integration with hospital databases and patient records, enabling continuous tracking of diagnostic history. Enhanced privacy measures, such as HIPAA- compliant encryption and user authentication, would be essential for safeguarding medical data. Moreover, collecting user feedback from healthcare professionals could support active learning frameworks that iteratively refine model performance based on real-world outcomes. Ultimately, collaboration with medical institutions for clinical validation and regulatory approval would be the next pivotal step in transitioning this system from a prototype to a deployable medical diagnostic tool.

.

V.CONCLUSION

This project successfully demonstrates the development of an intelligent and accessible brain tumor detection system by leveraging the capabilities of deep learning, RESTful APIs, and cross-platform mobile application frameworks. By integrating a Convolutional Neural Network trained on MRI images with a responsive Flutter front end and a Flask- based backend, the system provides a complete pipeline for real-time tumor detection. The high accuracy achieved during testing, along with low latency in image processing and prediction, confirms the technical soundness and clinical relevance of the solution.

The project not only addresses the limitations of manual diagnosis—such as time consumption and human error—but also offers a cost-effective, scalable alternative that can be deployed in diverse healthcare environments. Although the current system is limited to binary classification, its architecture allows for future enhancements including tumor segmentation, multi-class classification, and integration with hospital information systems.

In summary, this work exemplifies how modern AI tools can be effectively applied to solve critical problems in medical diagnostics. With further validation and development, the system holds strong potential to become a valuable decision-support tool for radiologists, enhancing the speed, accuracy, and reach of brain tumor diagnosis.

REFERENCES

1. Kaggle. (n.d.). Brain Tumor MRI Images Dataset. Retrieved from https://www.kaggle.com
2. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88.
3. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234–241). Springer, Cham.
4. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
5. Abadi, M., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467. Retrieved from https:// www.tensorflow.org
6. Flask Documentation. (n.d.). Flask: Python Web Framework. Retrieved from https://flask.palletsprojects.com
7. Flutter Documentation. (n.d.). Build Beautiful Apps. Retrieved from https://flutter.dev
8. Brownlee, J. (2018). Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python. Machine Learning Mastery.
9. Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481–2495.
10. Radiopaedia. (n.d.). MRI Brain Tumor Imaging Guide. Retrieved from https://radiopaedia.org