C.V. Raman Global University

Bhubaneswar, Odisha

Brain Tumor Detection

|  |  |  |
| --- | --- | --- |
| Name | Cranes Reg. No | Reg. No |
| Aniket Kit | CL202501060195575 | 2201020013 |
| Anmol Chourasia | CL2025010601915894 | 2201020184 |
| Mohit Kumar Ghosh | CL2025010601935963 | 2201020095 |
| Shivam Singh | CL20250106019168104 | 2201020226 |

Guided By:

Mr. Sandeep Padesur

Mr. Arib N

TABLE OF CONTENTS

|  |  |
| --- | --- |
| SL.NO | CONTENTS |
| 1. | Introduction |
| 2. | Objective |
| 3. | Dataset Description |
| 4. | Methodology |
| 5. | Model Architecture |
| 6. | Training and Evaluation |
| 7. | Web Application using Streamlit |
| 8. | Results and Discussion |
| 9. | Future Scope |
| 10. | Conclusion |
| 11. | References |

**Introduction**

* Brain tumors represent a critical health issue, often leading to severe neurological complications and, in many cases, fatal outcomes if not diagnosed and treated promptly. Traditional diagnostic methods, primarily based on manual analysis of Magnetic Resonance Imaging (MRI) scans, are not only time-consuming but also highly dependent on the expertise and judgment of radiologists. The complexity of brain structures and subtlety of early tumor indicators further compound the diagnostic challenges.
* Recent advancements in artificial intelligence (AI) and deep learning have provided promising tools to aid in medical diagnostics. Specifically, Convolutional Neural Networks (CNNs), a class of deep neural networks, have shown exceptional capabilities in image classification tasks. This project aims to harness the power of CNNs to build an automated brain tumor detection system that can analyze MRI images and classify them as either 'Healthy' or 'Tumor'.
* Moreover, to make this tool accessible to medical practitioners and non-technical users alike, the project integrates the model into a web application using Streamlit. This user-friendly interface allows users to upload MRI scans and receive real-time predictions, enhancing the practical utility of the model.

This report details each stage of the project, from dataset preparation and model training to application deployment, and highlights the challenges encountered and potential future enhancements.

**Objective**

The primary goal of this project is to design and implement an AI-driven system capable of detecting brain tumors from MRI scans with high accuracy and reliability. The specific objectives include:

* **Model Development**: Construct a Convolutional Neural Network (CNN) that can accurately differentiate between healthy brain scans and those containing tumors.
* **Data Preprocessing**: Prepare and normalize MRI images to ensure compatibility and consistency for training the neural network.
* **Web Deployment**: Develop an interactive web application using Streamlit, enabling users to upload images and obtain instant predictions.
* **Performance Evaluation**: Assess the model using various metrics such as accuracy, loss, and confusion matrix to ensure robustness.

The overarching aim is to reduce the time required for preliminary diagnosis, thereby assisting medical professionals in making timely decisions. While this tool is not intended to replace expert diagnosis, it serves as an effective assistive technology, potentially useful in clinical settings, especially where radiological resources are limited.

Through this project, we also aim to gain practical experience in the integration of machine learning models into deployable applications, bridging the gap between theoretical learning and real-world implementation.

**Dataset Description**

The dataset used for this project consists of categorized MRI images of the human brain, divided into two distinct classes: 'Healthy' and 'Tumor'. This binary classification framework simplifies the diagnostic challenge to a two-option decision system, which is ideal for initial implementation and performance testing.

The dataset is structured as follows:

* Two main folders: **Training** and **Validation**.
* Each folder contains two subdirectories: **Healthy** and **Tumor**.
* The images are in common formats such as .jpg and .png.

**Preprocessing Steps:**

* **Resizing**: All images are resized to 64x64 pixels to standardize input dimensions for the neural network.
* **Normalization**: Pixel values are scaled to a [0, 1] range by dividing by 255, improving model convergence during training.
* **Label Encoding**: Classes are converted to one-hot encoded vectors, e.g., Healthy → [1, 0], Tumor → [0, 1].

The dataset is relatively balanced in terms of class distribution, which aids in fair model training. However, real-world datasets often exhibit imbalance, which is a consideration for future work. No significant data augmentation was performed in this version, though it remains a potential avenue for enhancement.

**Methodology**

The methodology followed in this brain tumor detection project combines traditional machine learning pipeline structure with modern practices in deep learning and deployment. The complete process is structured into several stages:

**1. Data Acquisition and Preprocessing**

Images are collected from the designated dataset directory, segregated into 'Training' and 'Validation' sets. Each image undergoes resizing and normalization to maintain consistency and computational efficiency. Labels are one-hot encoded for use in categorical classification.

**2. Model Design**

A Convolutional Neural Network (CNN) is chosen for its proven effectiveness in image processing tasks. Layers are added progressively to extract spatial features and classify the input images. The architecture consists of multiple convolutional and max pooling layers followed by fully connected dense layers.

**3. Model Compilation**

The model is compiled using the Adam optimizer with a learning rate of 0.001 and the categorical cross-entropy loss function. These choices ensure efficient learning and are appropriate for a two-class classification problem.

**4. Model Training and Validation**

The network is trained for a defined number of epochs (10 in our case), and its performance is evaluated using the validation dataset. Accuracy and loss metrics are tracked to observe learning progress and detect overfitting.

**5. Model Saving and Integration**

After training, the model is saved in HDF5 format using Keras. This saved model is then loaded into a Streamlit web application, which serves as the front-end interface.

This methodology ensures a structured and repeatable approach to solving the classification problem and sets the stage for future improvements.

**Model Architecture**

The Convolutional Neural Network (CNN) used in this project is a sequential model consisting of the following layers:

* **Conv2D Layer (32 filters, 3x3 kernel)**: Extracts low-level features from the input images.
* **MaxPooling2D (2x2 pool size)**: Reduces spatial dimensions to prevent overfitting.
* **Conv2D Layer (64 filters, 3x3 kernel)**: Captures more complex features at a deeper level.
* **MaxPooling2D (2x2 pool size)**: Further spatial reduction.
* **Flatten Layer**: Converts the 2D feature maps into a 1D vector.
* **Dense Layer (64 units, ReLU activation)**: Fully connected layer for classification.
* **Dropout Layer (0.5 rate)**: Randomly drops 50% of the neurons to prevent overfitting.
* **Output Layer (2 units, Softmax activation)**: Final classification into either 'Healthy' or 'Tumor'.

**Model Compilation Details:**

* **Optimizer**: Adam
* **Loss Function**: Categorical Crossentropy
* **Metrics**: Accuracy

This architecture strikes a balance between model complexity and computational efficiency. It is capable of learning both simple and complex features necessary for effective image classification. The inclusion of a dropout layer helps to mitigate overfitting, which is a common issue in small datasets.

The choice of softmax activation in the output layer enables the model to output a probability distribution across the two classes, aiding in interpretability of the results.

**Implementation (Code Walkthrough)**

The project is implemented using Python, with several key libraries:

* **TensorFlow/Keras**: For building and training the CNN model.
* **OpenCV**: For image reading and processing.
* **NumPy**: For array operations.
* **Streamlit**: For creating the web-based application interface.

**Key Implementation Steps:**

1. **Data Loading**: A custom function loads images from the specified directories, resizes them, normalizes them, and assigns labels.
2. **Model Building**: The CNN model is built using the Sequential API in Keras, layer by layer.
3. **Training**: The model is trained using model.fit(), specifying training and validation datasets.
4. **Prediction and Deployment**: After saving the model, it is loaded in a separate Streamlit app, where uploaded images are preprocessed and passed to the model for prediction.

**Example Code Snippet:**

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)),

MaxPooling2D(2, 2),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Flatten(),

Dense(64, activation='relu'),

Dropout(0.5),

Dense(2, activation='softmax')

])

This modular approach allows easy debugging, updating, and deployment.

**Web Application Deployment (Streamlit Interface)**

To make the brain tumor detection model accessible to users, a front-end web application was developed using **Streamlit**, an open-source Python framework ideal for creating data-driven applications.

**Features of the Application:**

* **Image Upload**: Users can upload an MRI scan in .jpg, .jpeg, or .png format.
* **Real-Time Prediction**: The app preprocesses the image and predicts whether the scan is 'Healthy' or shows signs of a 'Tumor'.
* **User Feedback**: The prediction result is displayed along with the confidence score.

**Application Workflow:**

1. **Upload MRI Image**
2. **Image Resizing and Normalization**
3. **Model Prediction**
4. **Result Display**

**Example UI Code:**

uploaded\_file = st.file\_uploader("📁 Upload MRI Image", type=["jpg", "jpeg", "png"])

if uploaded\_file:

image = Image.open(uploaded\_file)

prediction = model.predict(preprocess\_image(image))

st.success(f"Prediction: {labels[class\_index]} ({confidence:.2f}%)")

This interactive interface ensures the system is not only powerful but also user-friendly, making it suitable for both medical professionals and the general public.

**Results and Evaluation**

The effectiveness of the brain tumor detection model was assessed using standard performance metrics after training and testing. The dataset was split into training and validation sets using predefined folders. The images were preprocessed uniformly by resizing to 64x64 pixels and normalizing pixel values to fall between 0 and 1. The training was performed over 10 epochs with validation feedback at each step.

**Evaluation Metrics**

* **Accuracy**: This is the proportion of correctly predicted labels out of the total number of predictions. The model achieved over 95% accuracy on both the training and validation datasets, indicating high reliability.
* **Loss**: The model’s loss steadily decreased with each epoch, indicating that the learning algorithm was effectively minimizing errors.
* **Confusion Matrix**: This was used to observe the types of errors the model made. Most misclassifications occurred in borderline images where tumors were small or low contrast.
* **Precision and Recall**: Precision was high, which means that when the model predicted a tumor, it was usually correct. Recall was also strong, suggesting that most actual tumors were identified correctly.

**Sample Predictions**

Upon uploading MRI scans in the Streamlit web interface, the model confidently predicted whether the brain was healthy or showed signs of a tumor. The predictions were usually accurate, and confidence scores were typically above 90%.

**User Experience**

* Users could upload .jpg, .jpeg, or .png files.
* A prediction with confidence percentage was displayed clearly after clicking the "Predict" button.
* Visualization of uploaded images helped confirm the input.

Overall, the model demonstrated robust performance suitable for educational or prototype-level medical diagnostic applications.

**Conclusion**

This project successfully implemented a Convolutional Neural Network to classify MRI images into two categories: healthy or tumor. The CNN was trained on a labeled dataset and showed high accuracy and reliability. The integration of the model with a Streamlit-based web app enhanced the project’s accessibility and user-friendliness, allowing real-time predictions through a simple interface.

Key accomplishments include:

* Building and training a deep learning model with high classification accuracy.
* Preprocessing medical image data efficiently.
* Deploying the model using a lightweight, interactive web app interface.
* Applying concepts like dropout for regularization, categorical crossentropy loss, and softmax classification.

**Future Work**

While the current model performs well, there are several ways it could be improved:

* **Expand the Dataset**: A larger, more diverse dataset will improve generalizability.
* **Data Augmentation**: Introduce rotations, flips, and lighting variations to make the model more robust.
* **Transfer Learning**: Use pre-trained models like VGG16 or ResNet for even better accuracy.
* **Mobile or Cloud Deployment**: Convert the model to TensorFlow Lite or deploy on platforms like Heroku or AWS.
* **Multi-class Classification**: Extend the model to detect different types or grades of tumors.

This project demonstrates how AI can make an impact in the healthcare sector by supporting early and accurate disease detection.

**References**

1. **Dataset Source**: Brain Tumor Classification Dataset - Kaggle
2. **Keras Documentation**: https://keras.io/api/
3. **Streamlit Documentation**: https://docs.streamlit.io/
4. Brownlee, J. (2019). *Deep Learning for Computer Vision*. Machine Learning Mastery.
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.