Brain Tumor Classification Using Deep Learning

Project Details

Project Title: Brain Tumor Classification Using Deep Learning

Developer: Aryan Sengar

Institution: The NorthCap University

Specialization: B.Tech CSE (AI & ML)

Project Type: Individual

Technologies Used: Python, TensorFlow, Keras, Streamlit, OpenCV, Matplotlib, FPDF

Repository: https://github.com/aryansengar007/brain-tumor-classifier

Introduction

Brain tumors are serious medical conditions that require accurate and early diagnosis. This project leverages Convolutional Neural Networks (CNNs) and a user-friendly Streamlit interface to classify MRI and CT brain scans as either "Healthy" or "Tumor". The goal is to provide a lightweight, interactive tool that supports real-time classification, visualization, and downloadable diagnostic reports.

Problem Statement

Radiologists face challenges in quickly and consistently analyzing brain scans. Manual inspection is prone to errors and delays. The project addresses this by:

- Automating classification using CNNs.

- Providing visual feedback and model confidence.

- Supporting batch predictions and detailed PDF report generation.

Key Observations

- Contrast-enhanced images yield better prediction accuracy.

- 128x128 grayscale images are sufficient for effective classification.

- A simple CNN architecture with dropout performs well for binary classification.

- Streamlit's ease of use allows rapid prototyping of diagnostic tools.

Challenges / Doubts

- Selecting the appropriate image preprocessing pipeline (normalization, enhancement).

- Balancing model complexity and training time.

- Ensuring clean separation between training and prediction pipelines.

- Managing file handling and temporary resources in Streamlit and PDF generation.

Libraries Used

- TensorFlow/Keras – Deep learning model training and prediction.

- Streamlit – Web app interface for interaction and visualization.

- OpenCV – Image loading and resizing from directories.

- Matplotlib/Seaborn – Graphing training metrics and confusion matrix.

- Pandas – Data logging and tabular export.

- FPDF – PDF report generation with images and text.

- NumPy – Efficient numerical computation.

- PIL – Image conversion and enhancement.

- Sklearn – Dataset splitting and performance reporting.

Functions Explained

- load_images(data_path): Loads and preprocesses images from Healthy/Tumor folders.

- generate_pdf_report(...): Produces diagnostic PDF with all relevant data and images.

- save_plot_to_buffer(...): Saves matplotlib figures as PNGs for PDF use.

- cleanup_temp_files(...): Deletes temporary files created during PDF generation.

- streamlit_app.py: GUI logic for uploading images, predictions, plots, and batch processing.

Main Components

1. Model Training (brain_ct_mri_classifier.py):

- Trains CNN using CT and MRI images.

- Produces .h5 file for deployment.

- Displays classification report, confusion matrix, and accuracy graph.

2. Streamlit Interface (streamlit_app.py):

- Uploads and processes single/multiple images.

- Performs real-time classification with image previews.

- Generates downloadable PDF reports with metadata, histograms, and prediction scores.

3. Batch Prediction Support:

- Accepts ZIP file of images.

- Classifies and outputs a CSV summary.

4. Interactive Dashboard:

- Shows prediction history and confidence distribution.

- Allows optional image contrast enhancement and flipping.

Future Improvements

- Extend model to detect tumor types (e.g., glioma, meningioma, pituitary).

- Integrate with cloud APIs for storage and model serving.

- Add DICOM support for hospital-grade MRI/CT images.

- Visualize Class Activation Maps (CAM) for explainability.

- Enhance report formatting with patient details.

Conclusion

This project demonstrates a complete pipeline from data preprocessing to model training and deployment in an interactive web application. It effectively bridges deep learning with real-world diagnostic use cases, allowing non-technical users (e.g., clinicians) to benefit from AI-powered tools. The codebase and documentation are published on GitHub for community collaboration and further development.