BrainTumorClassifier-CNN-Gradio

Project Documentation

Introduction:

(i) What is Brain Tumor Classification?
Brain tumor classification is a technique in medical imaging that identifies and categorizes brain tumors from MRI scans using machine learning or deep learning models. It helps doctors detect cancer early.

(ii) LLM (What is LLM/CNN?)
LLM in this project may refer to **Large Language Models** generally, but in image processing, **CNN (Convolutional Neural Network)** is the correct deep learning model used for image classification tasks like tumor detection.

(iii) What are models used in CNNs?
Models such as **VGG16, ResNet, and MobileNet** are commonly used CNN architectures. In this project, a custom CNN model is used for classification.

(iv) Features of Brain Tumor Classification using CNN:

- Automatic feature extraction
- High accuracy
- Handles large datasets
- Reduces manual efforts in diagnosis

(v) How CNN is used in Brain Tumor Classification?
The CNN model takes MRI images as input, extracts features, and classifies them into categories like **Glioma, Meningioma, Pituitary**, or **No Tumor**.

(vi) Evolving of Brain Tumor Classification:
From manual diagnosis to AI-based systems using CNNs and deep learning, the classification process is now faster, more accurate, and less dependent on manual interpretation.

Problem Statement:

→ About Project Statement

To build a deep learning model that can accurately classify different types of brain tumors from MRI images.

→ Limitation

- Requires high computation
- Dependent on quality of dataset
- May not perform well on unseen MRI types

Proposed System of Solution:

A CNN-based model that processes brain MRI images and classifies them into categories using image preprocessing, data augmentation, training, and evaluation steps.

Pipeline of Project:

→ **Explain Dataflow Chart:**

- **Data Collection:** Brain tumor MRI dataset (from Kaggle)
- **Evaluation:** Accuracy, Loss curves, and Classification report
- **Error Minimize:** Using image augmentation, dropout layers

→ **Coding:**

✅ **Key Code Snippets with One-Line Explanations**

import tensorflow as tf

🔹 Imports TensorFlow, the main deep learning library used for model building.

from tensorflow.keras.models import Sequential

🔹 Imports the Sequential model, which lets you build the model layer by layer.

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

🔹 Imports necessary layers to build a CNN architecture.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

- Loads a tool to automatically read and process images from folders.

---

train_datagen = ImageDataGenerator(rescale=1./255, ...)

- Scales image pixel values to the range [0, 1] for better model performance.

train_generator = train_datagen.flow_from_directory('Training', ...)

- Loads training images from the "Training" folder and assigns labels.

test_generator = test_datagen.flow_from_directory('Testing', ...)

- Loads test images from the "Testing" folder for evaluation.

---

model = Sequential()

- Starts a new sequential neural network model.

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)))

- Adds a convolutional layer with 32 filters for feature extraction from images.

model.add(MaxPooling2D(pool_size=(2, 2)))

- Reduces the spatial size (downsampling) to make computation easier.

model.add(Conv2D(64, (3, 3), activation='relu'))

- Adds another convolution layer to learn deeper image features.

model.add(MaxPooling2D(pool_size=(2, 2)))

- Again reduces the image size after the second convolution.

model.add(Flatten())

- Converts 2D feature maps into a 1D array to feed into dense layers.

model.add(Dense(128, activation='relu'))

- Adds a fully connected layer with 128 neurons and ReLU activation.

model.add(Dropout(0.5))

- Drops 50% of neurons randomly to prevent overfitting.

model.add(Dense(4, activation='softmax'))

- Output layer with 4 classes (types of tumors) using softmax for classification.

---

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

- Compiles the model using Adam optimizer and categorical cross-entropy loss.

model.fit(train_generator, epochs=10, validation_data=test_generator)

- Trains the CNN model using the training data and validates using test data.

---

model.save("BrainTumorCNNModel.h5")

- Saves the trained model in .h5 format for future use.

from keras.models import load_model

- Imports a function to load previously saved models.

model = load_model("BrainTumorCNNModel.h5")

- Loads the trained model from the saved file.

---

img = cv2.imread("image path")

- Reads an input MRI image using OpenCV.

img = cv2.resize(img, (64, 64))

- Resizes the image to 64x64 pixels as required by the model.

img = np.reshape(img, [1, 64, 64, 3])

- Reshapes the image to match the input shape expected by the model.

result = model.predict(img)

- Predicts the tumor class for the input MRI image.

print(np.argmax(result))

- Prints the predicted class index (e.g., 0 for Glioma, 1 for Meningioma, etc.).

→ **Architecture:**

- **Conv2D → MaxPooling2D → Dropout → Flatten → Dense → Softmax**
  This is the base structure of CNN in this project.

Feature Enhancement:

→ Overcome of a limitation of the project:

- **Data Augmentation** is used to increase the size and variety of training data
- Future scope: Use transfer learning or more advanced models like ResNet50

Conclusion:

The system successfully classifies brain tumors using CNN and MRI images. It supports doctors in faster diagnosis and can be improved further with advanced models and better datasets.