Tittle: Brain Tumor Detection using CNN

Group Members: Estefanos Kebebew, Eva Alvarez, and Sanket Shah

Introduction

Brain tumors are a leading cause of mortality and morbidity worldwide. Early and accurate diagnosis is crucial for effective treatment planning and improving patient prognosis. Traditional methods often rely on subjective interpretation of MRI scans, which can be time-consuming and prone to human error. Deep learning, particularly convolutional neural networks (CNNs), has emerged as a powerful tool for medical image analysis tasks, including brain tumor classification. This project aims to develop a CNN model using the PyTorch framework that can effectively classify MRI scans into normal and tumor-containing categories. This model has the potential to assist radiologists in diagnosis, expedite treatment planning, and ultimately contribute to improved patient outcomes.

Dataset

We will utilize the publicly available Brian Tumor Dataset from Kaggle https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset. This dataset comprises MRI scans in jpeg format, categorized as either normal or containing a tumor. We anticipate performing necessary preprocessing steps on the data, including normalization, and resizing. Depending on our initial analysis, techniques like data augmentation might also be explored.

Approach

Our approach to building a CNN model for brain tumor classification using MRI scans involves several key steps:

Preprocessing:

Data Loader: We will develop a custom data loader using PyTorch to load and manage the MRI scan data.
Splitting the Dataset: The data will be split into training, validation, and testing sets using a common split ratio (80%/10%/10%).
Noise Identification and Removal: Techniques for noise identification will be employed to detect and remove corrupted or irrelevant images from the dataset. This might involve analyzing image statistics, applying filtering techniques, or leveraging visual inspection.
Dimensionality Standardization: All MRI scans will be preprocessed to ensure they have consistent dimensions. This may involve resizing or padding images to a standard size suitable for the CNN architecture.

Building and Training the Model:

Hyperparameter Exploration: We will experiment with different hyperparameters of the CNN architecture to optimize its performance. This includes:

1. **Training Process:** The model will be trained using the PyTorch framework. The training process will involve iterating through the training data batches, calculating the loss on each batch, and updating the model's weights using the chosen optimizer to minimize the overall loss.
2. **Number of Layers:** We will explore models with varying numbers of convolutional and fully connected layers to find a balance between model complexity and learning capacity.
3. **Epochs:** We will determine the optimal number of training epochs to ensure the model converges effectively without overfitting to the training data.
4. **Optimizer:** We will evaluate different optimizers like Adam, or SGD.
5. **Loss Function:** We will choose an appropriate loss function like binary cross-entropy to measure the discrepancy between the model's predictions and the ground truth labels.

Results

Evaluation:

Unseen Data: The model's performance will be evaluated on the held-out testing set, which consists of unseen MRI scans not used during training. This ensures we assess the model's ability to generalize to new data.
Evaluation Metrics: We will employ various evaluation metrics to assess the model's performance comprehensively. These metrics include: accuracy,precision, recall, F1-Score.

- **AUC (Area Under the ROC Curve):** We will calculate the AUC to assess the model's ability to discriminate between normal and tumor scans..

- **Learning Curve:** We will monitor the model's training and validation loss over training epochs by plotting a learning curve.

Acknowledgment

Dr. Robert Mateescu