TRAINING A BRAIN TUMOR DETECTION MODEL

Task Objectives

1. Preprocessing the Dataset

Objective: Preparing dataset of MRI images for model.

- Selecting a dataset

  Using a pre-existing dataset Kaggle dataset ,which contains annotated images of brain tumor.

- Organizing the Data

  Creating a directory structure

```
/dataset
  ├── train
  │   ├── tumor
  │   └── no_tumor
  ├── val
  │   ├── tumor
  │   └── no_tumor
  └── test
      ├── tumor
      └── no_tumor
```

- Data Augmentation

  Apply transformations to increase the diversity of the dataset. This could include:

  a. Random rotations
  b. Flipping
  c. Resizing
  d. Normalization

- Normalize Images

  Normalize the pixel values for better training stability. Use values like mean=[0.5, 0.5, 0.5] and std=[0.5, 0.5, 0.5] for normalization.

2.Fine-tuning a Pre-trained Model

Objective: Using transfer learning to adapt a pre-trained model for binary classification.

Steps:

- Selecting a pre-trained model :
  I have selected ResNet50(a 50-layer deep residual network) pre-trained model because it is trained on large dataset(ImageNet). With the pre-trained weights, the model can converge faster during training, which is especially beneficial when fine-tuning on a new dataset.
- Modifying the final layer
  Replacing final fully connected layer of pre-trained model to match number of classes 2(tumor/no tumor)
- Loading Pre-trained Weights
  Load weights for the model to leverage previously learned features.

- Setting to Training Mode
  Ensuring that model is in training mode to update weights during training.

3.Model Training

Objective: Training model on dataset, adjusting hyperparameters for best performance.

Steps:

- Setting of Hyperparameters:
  Defining the batch size, learning rate, number of epochs,etc;

- Defining Function and Optimizer
  a)Used CrossEntropyLoss for binary classification because it is mostly used for classification and has gradient behavior.
  b) Used Adam as the optimizer because of it's faster convergence and robust to hyperparameters.

- Training the loop
  For each epoch:
  a) Iterating throughout the training data.
  b) Performing forward pass.
  c) Computing loss and gradients
  d) Updating model weights
  e) Printing loss for monitoring.

- Validation

After each epoch, evaluating the model on validation set to monitor performance and avoid overfitting.

4.Evaluating the Model

Objective: Measuring the performance of model using relevant metrics.

Steps:

- Testing:
  Using a separate test dataset for evaluating the model.

- Computing Matrics:
  Calculating of accuracy, precision, recall and F1-score:
  [TP: True positive ; TN: True Negative; FP: False Positive ; FN:False Negative]
  a) Accuracy: (TP+TN)/(TP + TN + FP + FN)
  b) Precision: TP/(TP+FP)
  c) Recall: TP/(TP+FN)
  d) F1- score: 2 *((precision*recall)/(precision + recall))

*****Starting with no prior experience in coding, I independently developed the MRI scanner project by utilizing various online resources and self-guided learning. This journey has been a testament to my dedication to learning and problem-solving.**

*****I found Yolov8 bit challenging so I completed my work using resnet**