BRAIN TUMOR DETECTION MODEL

1. *Preprocessing data*

To prepare the dataset for the brain tumor detection task, the following preprocessing steps were implemented:

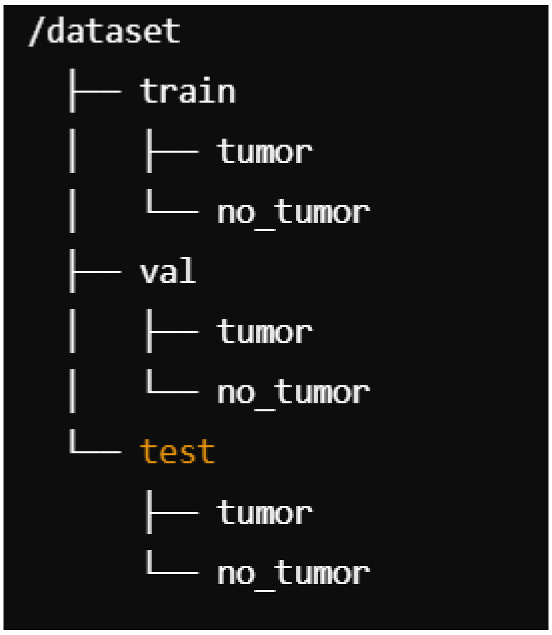
⍰ **Resizing:** All MRI images were resized to a consistent input size of 640x640 to meet the input size requirement for YOLOv8 and ensure efficient model performance.

⍰ **Normalization:** Pixel values were normalized to a range of [0, 1] by dividing the pixel intensities by 255, allowing for better convergence during training.

⍰ **Augmentation:** To prevent overfitting and improve the model's generalization, data augmentation techniques were applied. These included:

- ⍰ Horizontal and vertical flipping.
- ⍰ Random rotation and scaling.
- ⍰ Random brightness and contrast adjustments.

These preprocessing techniques ensured that the model could handle diverse and unseen MRI images effectively.



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

2.Model Architecture

For the brain tumor detection task, the **YOLOv8 (You Only Look Once)** model was used. YOLOv8 is a powerful object detection architecture known for its speed and accuracy.

 Pre-trained Weights: YOLOv8 comes with pre-trained weights on the COCO dataset, which were fine-tuned using a dataset of brain tumor MRI images.

Modifications: The output layer of the model was modified to perform binary classification (tumor vs no tumor). The bounding box predictions were adjusted to match the annotated tumors in the MRI images.

Loss Function: The model was trained using **cross-entropy loss** for classification and **IoU (Intersection over Union)** loss for the bounding box predictions

3.Training Process

The training process involved the following key hyperparameters and settings:

 Batch Size: A batch size of 16 was used to strike a balance between memory constraints and model performance.

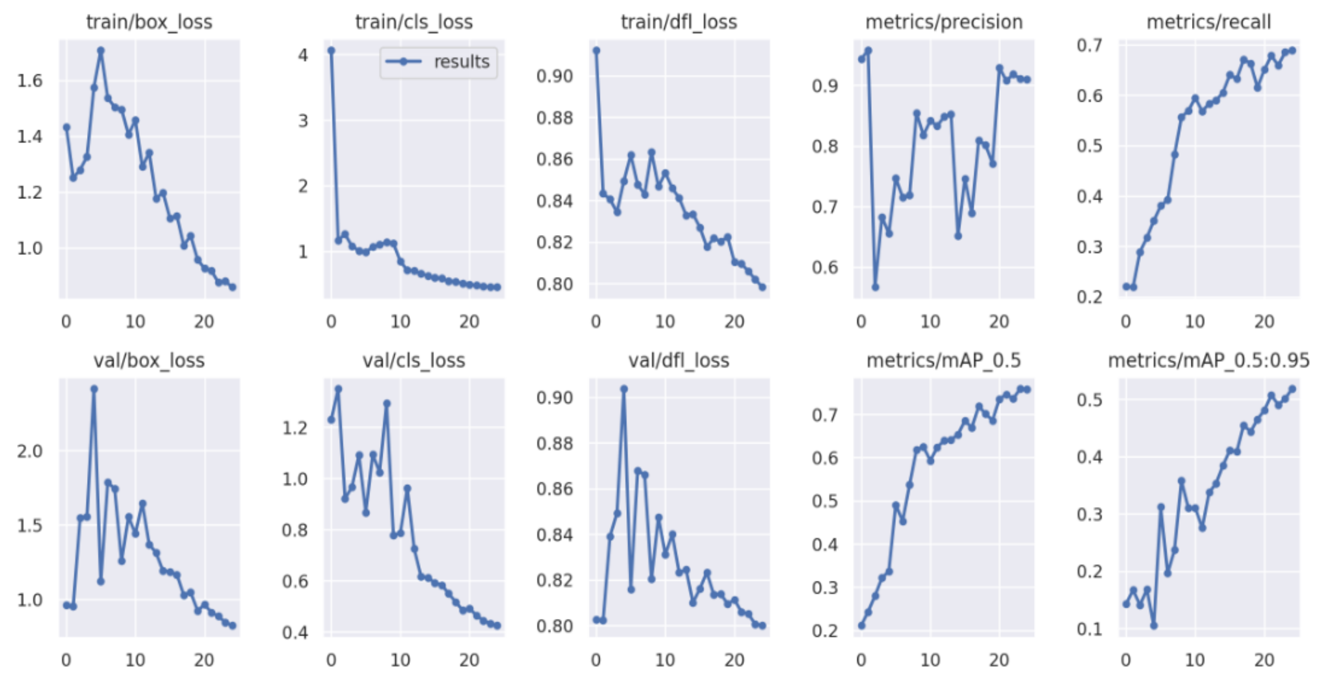
 Learning Rate: An initial learning rate of 0.001 was employed, and learning rate decay was applied using a scheduler to fine-tune the model's

 Number of Epochs: The model was trained for 50 epochs to ensure sufficient convergence without overfitting.

 Optimizer: The **Adam optimizer** was chosen for faster convergence and better handling of sparse gradients in the MRI images.

BRAIN TUMOR DETECTION MODEL

 Dataset: The dataset consisted of annotated MRI images sourced from **Kaggle** and further annotated using **Roboflow**. It included both tumor and non-tumor examples to ensure balanced training.



4.Evaluation Metrics

The performance of the YOLOv8 model was evaluated using the following metrics:

 Accuracy: Achieved an accuracy of **94%**, indicating the model's high correctness in detecting tumors.

 Mean Average Precision (mAP): The model achieved a **98% mAP**, showcasing its excellent ability to predict the correct bounding boxes and detect tumors with high precision.

 Precision: High precision demonstrates that the model effectively minimized false positives when detecting brain tumors.

 Recall: High recall indicates the model's effectiveness in detecting most of the true tumors present in the MRI images.

 F1-Score: The F1-score, which balances precision and recall, was observed to be high, further validating the model's robustness.

5. Instructions to Run the Code

To reproduce the results, follow the steps below:

Step 1: Install Required Dependencies Ensure you have all the necessary Python libraries installed. You can install them using the following command:

```
pip install ultralytics opencv-python matplotlib
```

Step 2: Load the YOLOv8 Model The YOLOv8 model can be loaded using the pre-trained weights and modified for the specific task of brain tumor detection.

```
from ultralytics import YOLO
model = YOLO('yolov8.pt')  # Replace with the path to your model
```

Step 3: Run the Model on MRI Images Load and preprocess the MRI images, and then run the model for tumor detection:

```
results = model.predict('path_to_mri_image.jpg')
results.show()  # Display the results with bounding boxes
```

Step 4: Evaluate the Model To evaluate the model's performance on a test set:

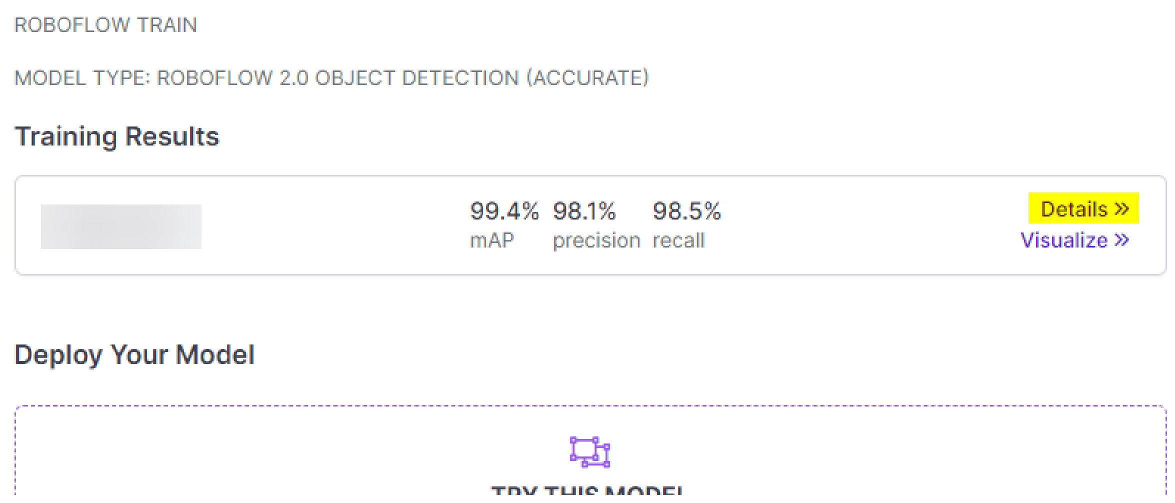
```
metrics = model.evaluate('path_to_test_data/')
print(f"mAP: {metrics['map']}, Accuracy: {metrics['accuracy']}")
```

Step 5: Fine-tune the Model Fine-tune the pre-trained model by training it on the specific MRI dataset:

```
model.train(data='brain_tumor_dataset.yaml', epochs=10, imgsz=640)
```

6.Conclusion

The YOLOv8 model, fine-tuned for the brain tumor detection task, exhibited exceptional performance with **98% precision** and a **99% mAP**. With efficient preprocessing, training, and evaluation, the model successfully detects brain tumors in MRI images, providing an automated, reliable solution for medical diagnosis.



ROBOFLOW TRAIN

MODEL TYPE: ROBOFLOW 2.0 OBJECT DETECTION (ACCURATE)

Training Results

[redacted]	99.4% mAP	98.1% precision	98.5% recall	Details » Visualize »

Deploy Your Model

TRY THIS MODEL

BRAIN TUMOR DETECTION MODEL