Deep Learning for Brain Tumor Detection Using Magnetic Resonance Imaging (MRI)

Leo Yao YY3959@NYU.EDU

Tianyu Zhao TZ2263@NYU.EDU

Yumo Li YL10192@NYU.EDU

Milestone 2

1. Methodology

When building the baseline ResNet model, we followed the pipeline: raw image files → data loaders → model definition → training with monitoring and early stopping → final evaluation. Specifically, we designed basic data transforms and augmentations, such as labeling and normalization. For training, we composed a pipeline of resizing, random rotations / affine shifts / horizontal flips, normalization, etc., to help the model generalize. For validation, we applied resizing and normalization. Next, we wrote a custom function `BrainTumorDataset`, which walks through each class subfolder, collects integer labels, loads images using OpenCV, converts them to RGB, wraps them in a PIL image, and applies the specified transforms. This design allowed easy integration with PyTorch's `DataLoader`. We then instantiated the full dataset and split it into 75% training and 25% validation subsets using a fixed random seed. We loaded a pre-trained ResNet-50 model (using ImageNet weights), replaced its final fully-connected layer with a dropout layer followed by a linear classification head for the four output classes, and moved the model to CUDA to utilize the GPU if available. For training, we selected `CrossEntropyLoss`, the Adam optimizer, and a `ReduceLROnPlateau` scheduler to lower the learning rate when the validation loss plateaued. We set up a TensorBoard `SummaryWriter` to track metrics including training and validation loss, accuracy, and confusion matrices over epochs. During each epoch, we trained on all batches (computing loss, backpropagating gradients, updating weights), evaluated on the validation set (computing loss, accuracy, and collecting predictions), logged scalar metrics and confusion matrix figures to TensorBoard, adjusted the learning rate via the scheduler, and saved the model if validation loss improved by more than 0.001. Otherwise, we incremented a no-improvement counter, and stopped early if validation loss did not improve for a number of consecutive epochs (defined by a `patience` parameter). We printed per-epoch training and validation loss and accuracy. After training, we plotted the loss and accuracy curves over epochs to visualize convergence and assess overfitting. Finally, we loaded the best-saved model, evaluated it on a held-out test set, computed the overall test loss and accuracy, and generated a final confusion matrix and classification report (including precision, recall, and F1-score per class) to assess real-world performance.

For data augmentation, we used TorchIO to simulate a more realistic setting. Using the TorchIO library, we applied two types of data augmentations to each image: `RandomBlur` and `RandomNoise`. These augmentations were chosen because real-world medical images

often contain imperfections, and applying such transformations can enhance dataset robustness and improve model generalization. For `RandomBlur`, we used `torchio.RandomBlur` to apply Gaussian blurring to each image. The standard deviation of the Gaussian kernel was randomly sampled from a uniform distribution between 0 and 2, ensuring that each image was blurred by a different amount. This mimics variations in focus or motion-induced softness commonly encountered in clinical imaging. The transformation was applied with a probability of 1.0, meaning every image underwent this augmentation. For `RandomNoise`, we used `torchio.RandomNoise` to inject Gaussian noise with a fixed mean of 0. The standard deviation was sampled from a uniform distribution between 0 and 25, allowing different levels of noise intensity. This simulates sensor noise or compression artifacts present in real-world image acquisition. Like the blur operation, noise was also applied with a probability of 1.0.

In addition to standard augmentations, we implemented a customized preprocessing pipeline based on edge detection to enhance structural information in the brain MRI dataset. Each image was first converted to grayscale and normalized to the $[0, 1]$ range. We applied a Gaussian filter (kernel size 5, $\sigma = 1.0$) to reduce noise while preserving anatomical structure. Sobel filters were used to compute gradient magnitudes and orientations, followed by non-maximum suppression to thin the edges. A double-thresholding step classified pixels into strong, weak, or non-edges, and hysteresis thresholding connected relevant edge segments. To enrich the input representation, we fused the grayscale image with the edge map using a weighted sum (emphasizing edges by a factor of 0.3), and further created pseudo-color RGB images by assigning varying weights to each color channel to highlight edge features. These enhanced RGB images were saved and used as the final dataset for training. This preprocessing strategy was designed to emphasize fine structural details while maintaining tissue contrast, potentially improving the model's ability to learn discriminative features for tumor classification.

2. Results

All four models achieved excellent performance on the brain MRI classification task. The accuracy is 0.9878 for the baseline, 0.9710 for the blurred, 0.9725 for the noised, and 0.9870 for the edged model. From the curve graphs (in the Appendix), we observe that during training, the training loss and the validation loss both rapidly decreased and stabilized at a very low value, whereas the training accuracy and the validation accuracy both rapidly increased and stabilized at a very high value. However, there are slight differences. The curves for the baseline ResNet are not as smooth as those of the other three models. In other words, the increasing or decreasing patterns for the baseline ResNet are not as stable as the others.

3. Analysis

The baseline model achieved the highest validation accuracy and weighted F1-score among the four, but its training curves exhibited more fluctuations, suggesting slightly less stability compared to the other models. Despite this, it converged quickly and maintained strong generalization performance.

	Baseline	Blurred	Noised	Edged
Validation Accuracy	0.9878	0.9710	0.9725	0.9870
Weighted Avg F1	0.99	0.97	0.97	0.99
Stops at Epoch	11	8	9	23

Table 1: Comparison of model performance across different data preprocessing strategies.

The blurred model converged the fastest, stopping at epoch 8, and showed smooth and stable training and validation curves. However, its validation accuracy and F1-score were slightly lower than the baseline, which may suggest that excessive blurring degraded critical features relevant for classification.

The noised model performed similarly to the blurred model, achieving comparable accuracy and F1-score. Its training was stable and early-stopped at epoch 9, indicating efficient convergence. The added Gaussian noise did not significantly degrade performance, which suggests that the model remained robust under moderate perturbations.

The edged model showed consistent and stable training with the longest training duration (23 epochs), likely due to the added complexity from the edge enhancement preprocessing. Its performance was nearly identical to the baseline, with similarly high accuracy and F1-score, and smoother learning curves.

Comparing across all models, we conclude that while all variations generalize well and exhibit strong performance, the baseline and edged models perform best in terms of accuracy and F1, albeit with different convergence behaviors. The blurred and noised models are slightly less accurate, possibly due to over-regularization, but still demonstrate robustness and smooth training. These results suggest that while basic augmentations can enhance generalization, excessive perturbation may obscure important anatomical features. Overall, the edge-based preprocessing appears to be the most beneficial augmentation for this brain tumor classification task.

4. Plan for Additional Analysis

Based on the current results, we propose conducting additional analyses to gain deeper insight into the impact of preprocessing techniques and model architectures on brain tumor classification performance.

First, we plan to experiment with alternative preprocessing methods, such as skull stripping to remove irrelevant non-brain tissues, simulated random motion to mimic realistic patient movement artifacts, and the addition of random blur, motion and noise to introduce variability. These techniques are expected to challenge the model's robustness and generalization ability, revealing whether the current pipeline is overly dependent on clean, high-contrast inputs.

Second, we propose to evaluate different deep learning architectures, such as YOLOv7, Faster R-CNN, and U-Net variants. By adapting them to the brain MRI classification

problem, we can assess whether models that incorporate localization awareness or region-based attention improve predictive performance.

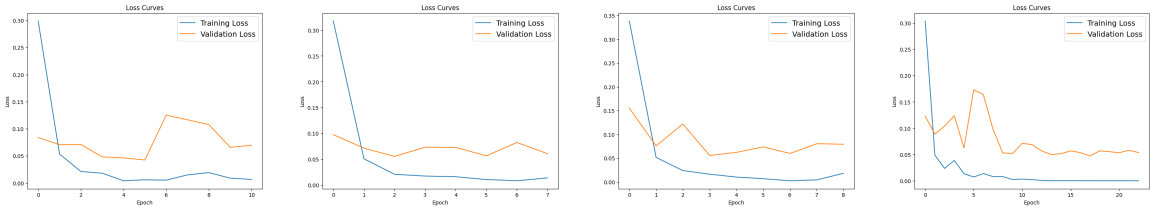
Through this extended analysis, we aim to understand how sensitive the model is to different types of image distortions and structural changes, and whether incorporating alternative architectures can further boost classification accuracy or robustness. This would provide a more comprehensive evaluation of the system's potential for real-world clinical deployment.

5. Work Plan

Milestone 2	**Leo Yao**	**Tianyu Zhao**	**Yumo Li**
3/10–3/23	Discussion with the TA about the possibilities of the methodology		
3/31–4/6	Did some research on using ResNet and YoloV7 on medical image classification	Did some research on medical image data augmentation using TorchIO	Did some research on edge detection using Canny and skull stripping using `brainles_preprocessing`
4/7–4/13	After further discussion with the TA to refine our methodology, we decided not to proceed with the YOLOv7 approach used in the original brain tumor paper. The method was overly complex, incorporating numerous mechanisms beyond this class. Additionally, the reported accuracy of 99.5% seemed unrealistically high, likely due to the use of fancy techniques unfamiliar to us. We also ruled out the possibility of training models on compressed data for comparison with the original dataset, as well as the option of performing feature extraction.		
4/14–4/20	Wrote the code for a baseline ResNet model trained on the brain tumor images for further comparison and trained it using original images	Wrote the code to add blur and noise to images using TorchIO; trained the ResNet using these transformed images as input	Wrote the code to detect the edges from the images; trained the ResNet using the images with emphasis on the edges as input
Future	**Leo Yao**	**Tianyu Zhao**	**Yumo Li**
4/21–4/27	Continue doing research on YoloV7	Do some research on RCNN	Write the code for skull stripping to remove irrelevant non-brain tissues
4/28–5/4	Try writing the code for YoloV7; if technically or timely difficult, switch to doing some research on U-Net	Write the code for motion transformation and mixing the three transformations as one single transformation	Write the code for adding the edge detection as a mask to the original image (optional, if technically and timely possible)
5/5–5/11	Write the code for YoloV7 or RCNN or U-Net (if technically and timely possible, we can do more than one)		

Appendix A.

This appendix includes plots generated for examining the training and validation loss and accuracy of the four models trained in Python.

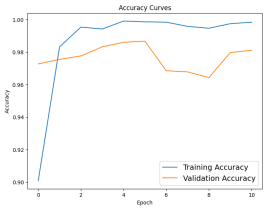


(a) Baseline - Loss

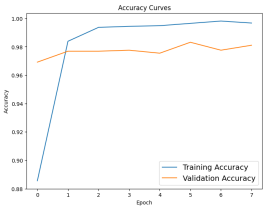
(b) Blurred - Loss

(c) Noised - Loss

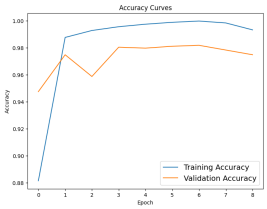
(d) Edged - Loss



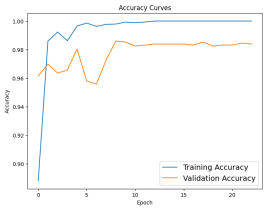
(e) Baseline - Accuracy



(f) Blurred - Accuracy



(g) Noised - Accuracy



(h) Edged - Accuracy

Figure 1: Training Loss and Accuracy Curves for All Models