Project Report: Brain Tumor Segmentation using UNet++

- By Sreekar Balagoni

Introduction

This project focuses on identifying and segmenting brain tumors from MRI scans using a deep learning model. Medical image segmentation helps doctors analyze scans more easily by marking the regions where tumors are present. In this project, I used a model called UNet++ along with data from the BraTS 2021 dataset.

Dataset

The dataset consists of 3D MRI scans of brains and corresponding segmentation masks. These files are stored in `.nii.gz` format, which is commonly used in medical imaging. Each scan has different types of images (modalities), but for simplicity, I used the FLAIR modality and its corresponding tumor segmentation mask. The dataset was downloaded using kagglehub, and I selected 200 patients to work with.

Tools and Libraries Used

- Python — Programming language used
- PyTorch — For building and training the deep learning model
- Albumentations — For data augmentation like flipping, rotating, and resizing
- segmentation_models_pytorch — Provided the UNet++ model with a ResNet34 backbone
- nibabel — For reading `.nii.gz` medical image files
- matplotlib — For visualizing results
- scikit-image — For resizing the images
- CUDA — Used GPU for faster training

Model Used: UNet++

UNet++ is an improved version of the popular UNet model. It has extra skip connections that help it better understand image features. I used a version of UNet++ with a ResNet34 encoder, which helps the model learn better from image patterns and gives the output precisely with exact boundaries, which UNet could not do.

Data Preprocessing

- Selected the middle slice from each 3D brain scan (to simplify the data)
- Resized all images and masks to 128x128 pixels
- Normalized the image intensity values between 0 and 1
- Converted tumor masks into binary format: 1 for tumor, 0 for background

Data Augmentation

To help the model learn better, I applied random augmentations using Albumentations:

- Horizontal and vertical flips
- Random rotations
- Brightness and contrast changes
- Shifts and scale variations

These help simulate real-world variations in the scans and reduce overfitting.

Training the Model

- Loss Function: Binary Cross Entropy with Logits (BCEWithLogitsLoss)
- Optimizer: Adam
- Learning Rate: 0.0001
- Epochs: Trained for 100 epochs
- Mixed Precision Training: Used torch.amp for faster training with less memory
- Used 180 patients for training and 20 for testing

```
Saved checkpoint: unetpp_epoch80.pth
Epoch 81/100 | Loss: 0.3802
Epoch 82/100 | Loss: 0.3765
Epoch 83/100 | Loss: 0.3798
Epoch 84/100 | Loss: 0.3878
Epoch 85/100 | Loss: 0.3775
Epoch 86/100 | Loss: 0.3684
Epoch 87/100 | Loss: 0.3617
Epoch 88/100 | Loss: 0.3570
Epoch 89/100 | Loss: 0.3616
Epoch 90/100 | Loss: 0.3631
Saved checkpoint: unetpp_epoch90.pth
Epoch 91/100 | Loss: 0.3519
Epoch 92/100 | Loss: 0.3465
Epoch 93/100 | Loss: 0.3575
Epoch 94/100 | Loss: 0.3823
Epoch 95/100 | Loss: 0.3406
Epoch 96/100 | Loss: 0.3460
Epoch 97/100 | Loss: 0.3340
Epoch 98/100 | Loss: 0.3362
Epoch 99/100 | Loss: 0.3278
Epoch 100/100 | Loss: 0.3147
Saved checkpoint: unetpp_epoch100.pth
```

Evaluation Metrics

To check how well the model works, I used:

- Dice Coefficient: Measures how much the predicted tumor area overlaps with the actual tumor
- IoU (Intersection over Union): Measures the common area between prediction and ground truth

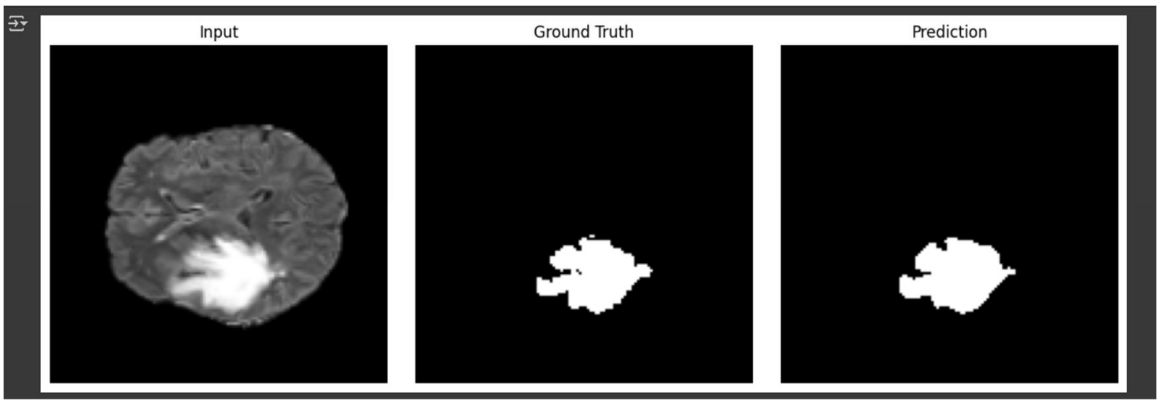
These metrics were calculated on test images after thresholding predictions.

```
Average Dice Coefficient: 0.8230
Average IoU Score: 0.7491
```

Results and Visualizations

After training:
- The model was able to identify tumor regions on new test images
- I visualized results by showing:
  1. Input MRI slice
  2. Ground truth tumor mask
  3. Predicted mask from the model

The predicted masks showed a good match with the real masks in most cases.



Output Files

- unetpp_epochXX.pth — Saved model weights after training 10 epochs.
- Google Colab Notebook — Contains the complete code for downloading data, preprocessing, training, and evaluating the model

Conclusion

This project shows how deep learning can be used for medical image segmentation. Using UNet++ and Albumentations helped improve the performance of the model. Even though I started with limited knowledge, the project was successful in training a working brain tumor segmentation model.