Advanced Brain Tumor Segmentation: Preprocessing, Augmentation, and Deep Learning Analysis Using BraTS 2024 Competition Data

DEEP LEARNING
Prof. Amir Jafari
DATS 6303_10

Report By:
Sai Avinash Nadakuditi
G26357147

INTRODUCTION

In this project, we're tackling a really important problem in brain cancer research. We're working with MRI scans of brains with a type of tumor called a "glioma." Gliomas are tricky because they spread out within the brain, making them hard to treat.

Our main task is to teach a neural network model to accurately identify different parts of the tumor from the MRI images. It's kind of like a really advanced coloring book where the computer has to color in the different areas of the tumor. This is called "segmentation." Here we are developing an automated segmentation algorithm for post-treatment glioma analysis using multi-parametric MRI (mpMRI) data. This is essential because post-treatment imaging plays a pivotal role in assessing treatment response, planning further interventions, and predicting patient outcomes. However, manual segmentation of tumor sub-regions is time-consuming and prone to variability in radiologist experience. An automated approach offers the potential for increased efficiency and objectivity.

We utilize the BraTS 2024 dataset, a comprehensive collection of mpMRI scans from multiple institutions, for this purpose. This dataset provides a diverse range of cases, including both high- and low-grade gliomas, and incorporates the complexities of post-treatment changes such as resection cavities. Each mpMRI scan includes four modalities: native T1 (t1), post-contrast T1-weighted (t1c), T2-weighted (t2w), and T2 Fluid Attenuated Inversion Recovery (FLAIR, t2f). These different modalities provide complementary information about tumor tissue characteristics.

Our algorithm aims to segment the following tumor sub-regions, each with distinct clinical significance:

1) **Enhancing Tumor (ET):** Indicates areas of active tumor growth and blood-brain barrier breakdown.
2) **Non-Enhancing Tumor Core (NETC):** Represents necrotic or cystic regions within the tumor.
3) **Surrounding Non-enhancing FLAIR Hyperintensity** (SNFH): Encompasses edema, infiltrating tumor cells, and post-treatment effects.
4) **Resection Cavity** (RC): Identifies areas where the tumor has been surgically removed.

By accurately segmenting these sub-regions, we can enable quantitative assessment of tumor volume, extent of infiltration, and treatment-related changes. This information can aid in personalized treatment planning, and monitoring.

Started by collecting the data, we performed necessary conversions and preprocessing steps. We then developed and trained two successful models: 3D U-Net and Residual 3D U-Net. Additionally, we explored the use of GANs for synthetic data generation and experimented with FCNNs, but these approaches did not yield satisfactory results. Further details about each stage will be elaborated in the subsequent sections of this report.

A crucial aspect of this project was the accurate measurement of tumor regions, both in terms of their volume and boundaries. Our primary goal was to develop algorithms that could reliably identify the full extent of the tumor, including its various sub-regions. To evaluate the performance of our models, we utilized lesion-wise Dice Similarity Coefficient (DSC) and 95% Hausdorff Distance (HD95). The DSC measures the overlap between the predicted segmentation and the ground truth, providing insight into the

volumetric accuracy of our models. The HD95, on the other hand, focuses on the boundary accuracy by assessing the distance between the predicted and actual tumor boundaries. By evaluating performance on a per-lesion basis, we can ensure that our models are effective at detecting and segmenting even small or complex tumor regions.

DATASET

The BraTS 2024 dataset, comprising multi-institutional, post-treatment mpMRI scans of gliomas, served as the foundation for our project. This dataset provided a rich source of information, with each of the 1350 subjects having five images: four MRI modalities (T1, T1Gd, T2, FLAIR) and a corresponding ground truth segmentation meticulously annotated by expert neuroradiologists. These annotations delineated crucial tumor sub-regions, including enhancing tumor (ET), non-enhancing tumor core (NETC), surrounding non-enhancing FLAIR hyperintensity (SNFH), and resection cavity (RC). To ensure robust model development and evaluation, we partitioned the dataset, allocating 80% (1080 subjects) for training and the remaining 20% (270 subjects) for validation. This division allowed us to train our models on a substantial portion of the data while retaining an independent set for unbiased assessment of their performance.

Through the literature review and research, we found the below relevant and significant information regarding the tumor regions and scans.

Types of scans:

1) **T1w (T1-weighted):** This is like a basic anatomical picture of the brain. It shows the different tissues with good detail. Watery areas look dark, and fatty tissues look brighter.
2) **T1Gd (T1-weighted with Gadolinium contrast):** This is like the T1w scan but with a special dye injected into the bloodstream. The dye helps highlight areas with increased blood flow or breakdown of the blood-brain barrier, which often happens in tumors.
3) **T2w (T2-weighted):** This scan is good at showing water. Areas with a lot of water, like fluid-filled spaces or swelling, appear bright.
4) **FLAIR (Fluid-Attenuated Inversion Recovery):** This is a special type of T2w scan that suppresses the signal from the fluid surrounding the brain. This makes it easier to see subtle abnormalities within the brain tissue itself, such as those caused by tumors or inflammation.

Scan significance

1) **T1c is crucial for identifying the enhancing tumor (ET - Label 3):** The contrast agent helps highlight the active, growing parts of the tumor.
2) **FLAIR is best for visualizing the surrounding non-enhancing FLAIR hyperintensity (SNFH - Label 2):** It helps distinguish the edema and infiltrating tumor from the surrounding cerebrospinal fluid (CSF).
3) **T2w provides valuable information about both the non-enhancing tumor core (NETC - Label 1) and the SNFH (Label 2):** It helps identify areas of necrosis and swelling.
4) **All sequences contribute to identifying the resection cavity (RC - Label 4):** It typically appears dark on T1w and bright on the other sequences.

Each MRI scan within the BraTS 2024 dataset is stored in the NIfTI (.nii) file format and possesses a three-dimensional structure of 182 x 218 x 182 voxels. These dimensions correspond to 182 slices in the sagittal plane (side view), 218 slices in the coronal plane (front view), and 182 slices in the axial plane (top-down view). Each voxel represents a 1 mm cube of brain tissue, providing a high-resolution representation of the brain's anatomy.

My individual contribution to this project was heavily focused on the research and conceptualization aspects of our brain tumor segmentation algorithms. To lay the groundwork for our model development, I embarked on an extensive literature review, immersing myself in the intricacies of tumor biology, growth patterns, and the underlying mechanisms driving their progression. This deep dive allowed me to understand how different tumor regions interact and influence each other, which proved invaluable in translating this biological problem into a framework suitable for neural networks.

Furthermore, I delved into the existing body of work on brain tumor segmentation, exploring various approaches and identifying the strengths and weaknesses of different models. This exploration led me to focus on three promising architectures: Fully Convolutional Neural Networks (FCNNs), 3D U-Net, and Residual 3D U-Net, all of which have demonstrated considerable success in medical image segmentation tasks. Recognizing the limitations posed by the relatively small size of our dataset, I investigated techniques for data augmentation. Specifically, I explored the potential of Generative Adversarial Networks (GANs) to create synthetic brain tumor images. GANs, with their ability to learn and mimic the underlying data distribution, offered a promising avenue for expanding our training set and improving the generalization capabilities of our models.

FCNN Architecture:

```
### FCNN Architecture ###
class FCNN3D(nn.Module):  2 usages
    def __init__(self, in_channels, out_channels):
        super(FCNN3D, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv3d(in_channels, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(kernel_size=2, stride=2)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose3d(64, out_channels, kernel_size=2, stride=2),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
```

Unet3d Architecture:

```
class UNet3D(nn.Module):  2 usages
    def __init__(self, in_channels, out_channels):
        super(UNet3D, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv3d(in_channels, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv3d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(2)
        )
        self.middle = nn.Sequential(
            nn.Conv3d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv3d(128, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool3d(2)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose3d(128, 64, kernel_size=2, stride=2),
            nn.ReLU(inplace=True),
            nn.Conv3d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.ConvTranspose3d(64, out_channels, kernel_size=2, stride=2),
            nn.Softmax(dim=1)
        )

    def forward(self, x):
        x1 = self.encoder(x)
        x2 = self.middle(x1)
        x3 = self.decoder(x2)
        return x3

# Instantiate the model
model = UNet3D(in_channels=4, out_channels=5)
```

GAN Architecture:

```
# Generator and Discriminator
class Generator(nn.Module):  3 usages  ± SaiAvinash-23
    def __init__(self, input_channels=4, output_channels=1):  ± SaiAvinash-23
        super(Generator, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv3d(input_channels, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv3d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.ReLU(inplace=True)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose3d(128, 64, kernel_size=3, stride=2, padding=1, output_padding=1),
            nn.ReLU(inplace=True),
            nn.Conv3d(64, output_channels, kernel_size=3, padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):  ± SaiAvinash-23
        return self.decoder(self.encoder(x))

class Discriminator(nn.Module):  4 usages  ± SaiAvinash-23
    def __init__(self, input_channels=5):  ± SaiAvinash-23
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Conv3d(input_channels, 64, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv3d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv3d(128, 1, kernel_size=3, padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):  ± SaiAvinash-23
        return self.model(x)
```

To ensure rigorous evaluation of our segmentation results, I also delved into the theoretical understanding of various performance metrics. I focused on understanding the Dice coefficient and the 95% Hausdorff distance, two widely used metrics in medical image segmentation.

Dice Coefficient: This metric measures the overlap between the predicted segmentation (S) and the ground truth (G). It's calculated as:

*Dice Coefficient (DSC) = 2 * |S ∩ G| / (|S| + |G|)*

where |S ∩ G| represents the number of voxels common to both the segmentation and the ground truth, and |S| and |G| represent the total number of voxels in each. A higher Dice coefficient indicates better agreement between the prediction and the ground truth.

Dice coefficient:

```python
def dice_coefficient(preds, targets, smooth=1e-6):  3 usages
    preds = preds.argmax(dim=1)
    intersection = (preds * targets).sum()
    union = preds.sum() + targets.sum()
    dice = (2.0 * intersection + smooth) / (union + smooth)
    return dice.item()
```

95% Hausdorff Distance (HD95): This metric focuses on the distance between the boundaries of the predicted segmentation and the ground truth. It's calculated as the 95th percentile of the distances between all pairs of boundary points. A lower HD95 value indicates better boundary agreement.

```python
def compute_hausdorff(pred, target):  1 usage  SaiAvinash-23
    """Compute Hausdorff Distance."""
    pred = (pred > 0.5).cpu().numpy()
    target = target.cpu().numpy()
    pred_coords = np.argwhere(pred)
    target_coords = np.argwhere(target)

    if len(pred_coords) == 0 or len(target_coords) == 0:
        return float("inf")  # Handle empty predictions or targets
    h1 = directed_hausdorff(pred_coords, target_coords)[0]
    h2 = directed_hausdorff(target_coords, pred_coords)[0]
    return max(h1, h2)
```

In addition to these metrics, I also explored techniques for visualizing and analyzing segmentation errors. This included the use of heatmaps to highlight misclassified regions on a per-patient basis, as well as boundary distance analysis and segmentation boundary analysis to pinpoint areas where the model struggled to accurately delineate tumor boundaries.

This comprehensive research and analysis phase provided a solid foundation for our team to develop and implement effective brain tumor segmentation algorithms. The insights gained from my individual work informed our model selection, training strategies, and evaluation procedures.

I worked on reprocessing the dataset and experimenting with various data augmentation techniques to enhance the performance of our segmentation models.

1) Resizing and normalizing: To ensure consistency and improve computational efficiency, I resized the images to a standardized resolution. This involved careful consideration of the original image dimensions and the desired output size. I then applied intensity normalization to scale the pixel values within a specific range. This step helps mitigate the effects of variations in image acquisition protocols and scanner settings across different institutions.

2) Recognizing the limited size of the dataset, I implemented a variety of data augmentation techniques to artificially increase the diversity of our training data. The augmentations I experimented with included:
   a) **Brain mask generation**: I generated brain masks to isolate the brain tissue from the surrounding skull and background.
   b) **Intensity Clipping:** I clipped the intensity values to a specific range, removing extreme outliers that could potentially skew the model's learning process.
   c) **Bounding Box Cropping:** I employed bounding box cropping to extract regions of interest containing the tumor and its surrounding tissues. This reduces the computational burden and allows the model to focus on the most relevant image areas.
   d) **Rotation:** I applied random rotations to the images, simulating variations in patient positioning during MRI acquisition.
   e) **Gaussian Filtering:** I utilized Gaussian filters to smooth the images and reduce noise, potentially improving the model's ability to discern subtle tumor boundaries.

```python
def create_brain_mask(image_data):  1 usage  SaiAvinash-23 *
    return (image_data > 0).astype(np.uint8)

def clip_intensity(image_data, lower_percentile=2.5, upper_percentile=97.5):  1 usage  SaiAvinash-23 *
    lower = np.percentile(image_data, lower_percentile)
    upper = np.percentile(image_data, upper_percentile)
    return np.clip(image_data, lower, upper)

def normalize_intensity(image_data):  1 usage  SaiAvinash-23 *
    mean = np.mean(image_data)
    std = np.std(image_data)
    return (image_data - mean) / (std + 1e-8)

def crop_to_bounding_box(image_data, mask_data, padding=20):  1 usage  SaiAvinash-23 *
    coords = np.array(np.nonzero(mask_data))
    min_coords = np.maximum(np.min(coords, axis=1) - padding, 0)
    max_coords = np.minimum(np.max(coords, axis=1) + padding, image_data.shape)
    slices = tuple(slice(min_coords[i], max_coords[i]) for i in range(3))
    return image_data[slices], mask_data[slices]
```

```python
def elastic_transform(image, alpha, sigma):  2 usages  SaiAvinash-23
    print("Applying elastic transformation...")
    shape = image.shape
    dx = gaussian_filter((np.random.rand(*shape) * 2 - 1), sigma) * alpha
    dy = gaussian_filter((np.random.rand(*shape) * 2 - 1), sigma) * alpha
    dz = gaussian_filter((np.random.rand(*shape) * 2 - 1), sigma) * alpha
    x, y, z = np.meshgrid( *xi: np.arange(shape[0]), np.arange(shape[1]), np.arange(shape[2]), indexing='ij')
    indices = [np.clip(x + dx,  a_min: 0, shape[0] - 1), np.clip(y + dy,  a_min: 0, shape[1] - 1), np.clip(z + dz,  a_min: 0, shape[2] - 1)]
    return map_coordinates(image, indices, order=1, mode='reflect')
```

Following the preprocessing and augmentation steps, I actively participated in designing the architectures for our 3D U-Net and Residual 3D U-Net models. This involved a collaborative effort with my teammates, where we experimented with various configurations to optimize performance. We started with a standard 3D U-Net architecture as our baseline and then explored modifications to tailor it to our case.

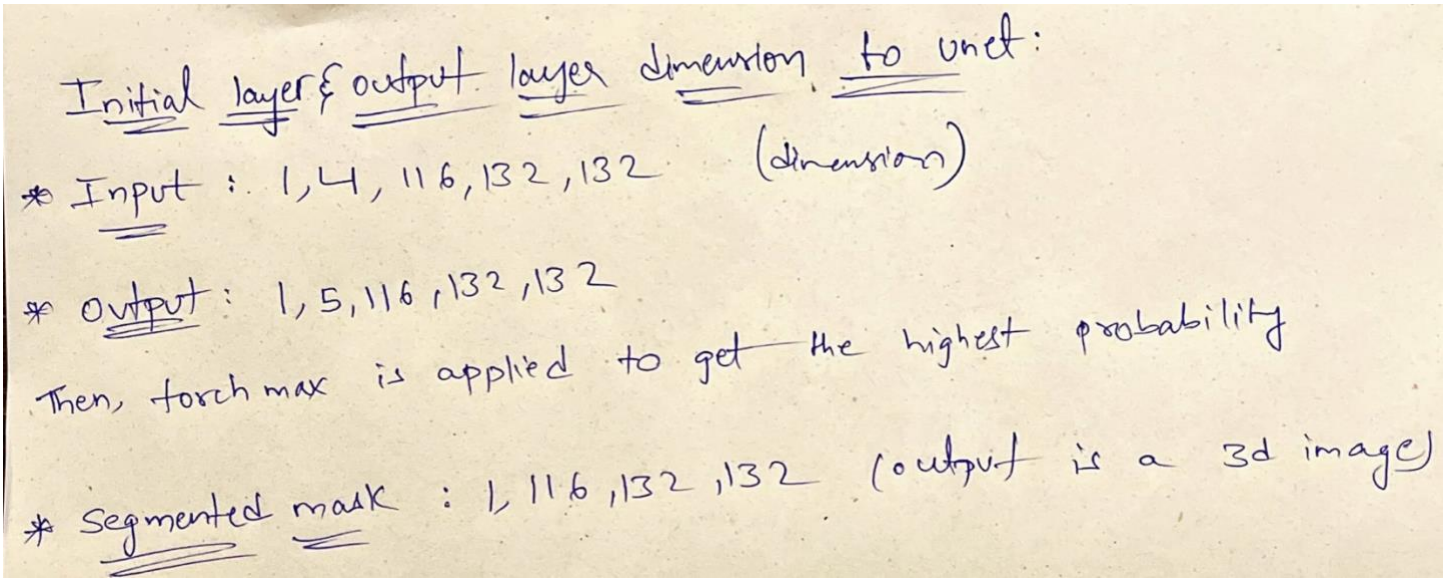
We experimented with different numbers of encoder and decoder stages to find a balance between capturing intricate details and maintaining computational efficiency. We considered architectures with 4 to 6 encoder/decoder stages, analyzing the impact on feature representation and segmentation accuracy.

We adjusted the number of convolutional layers within each encoder and decoder block. This involved considering factors such as receptive field size and feature map dimensionality. We explored variations with 2 to 4 convolutional layers per block.

We experimented with different filter sizes (e.g., 3x3x3, 5x5x5) in the convolutional layers to assess their impact on capturing spatial information and feature extraction. We carefully considered the placement and implementation of skip connections between encoder and decoder stages. These connections help preserve fine-grained details and mitigate the vanishing gradient problem.

During our initial experiments with the 3D U-Net, we observed signs of the vanishing gradient problem. This manifested as slow convergence during training, with the model struggling to learn effectively. Specifically, we noticed that the training loss plateaued early, and the model's performance on the validation set was suboptimal. This led us to explore the Residual 3D U-Net architecture.

Dimension Details:



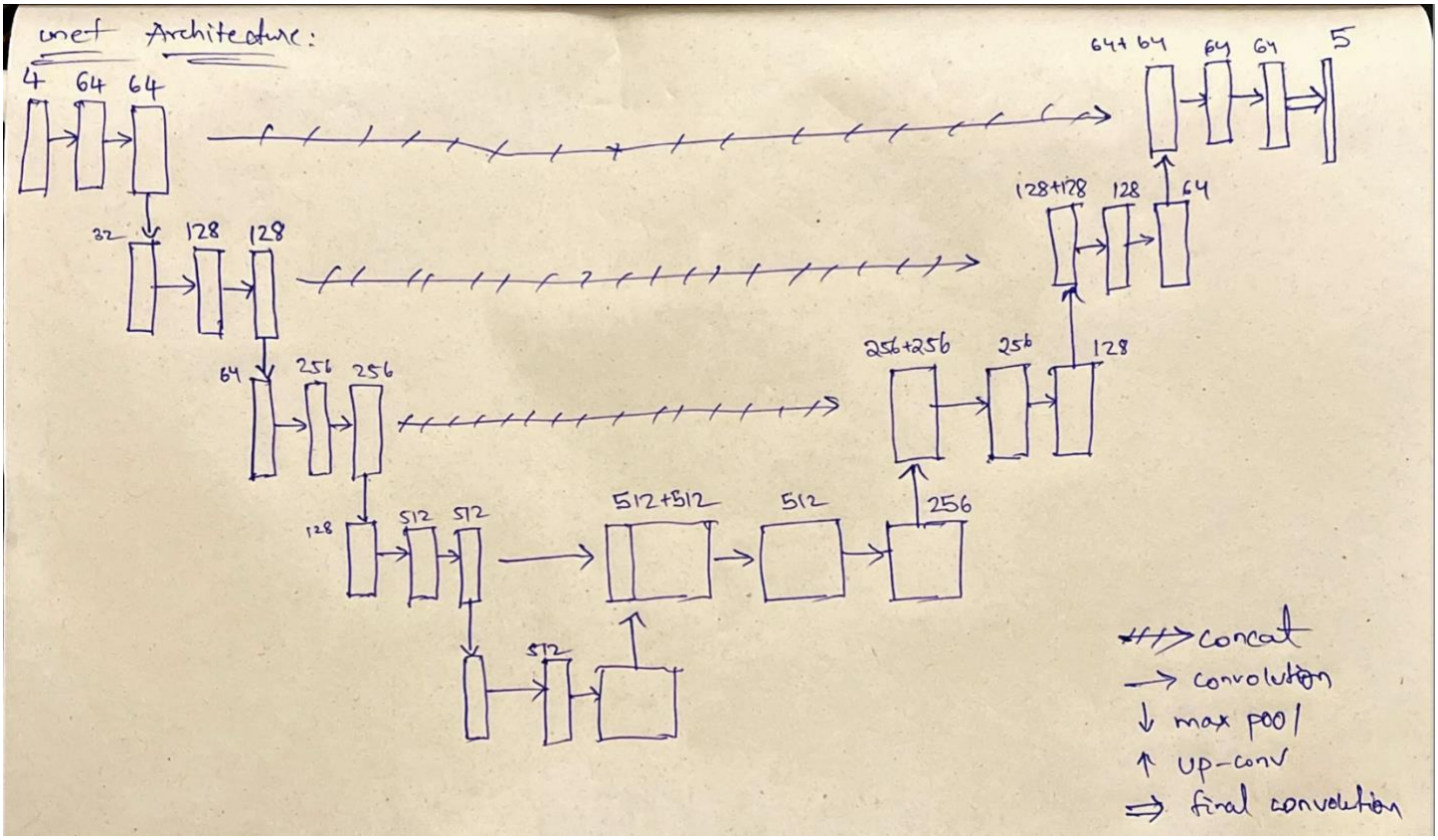
To combat the vanishing gradient problem and potentially improve performance, we incorporated residual connections into our U-Net architecture. These connections allow gradients to flow more easily through the network, facilitating training and enabling the model to learn more complex features. In designing the Residual 3D U-Net, we focused on:

1) We integrated residual blocks within the encoder and decoder pathways. Each residual block consisted of convolutional layers, batch normalization, and ReLU activation, with a skip connection that bypassed these operations.
2) We experimented with different strategies for placing residual connections, considering their impact on gradient flow and feature learning.

While the 3D U-Net and Residual 3D U-Net showed promising results, we also explored alternative approaches, namely GANs and FCNNs. However, we encountered significant challenges with these models:

1) Training GANs, especially for 3D medical image generation, proved to be extremely memory-intensive. Despite our efforts to optimize the architecture and training process, we faced persistent memory issues. The GAN models frequently ran out of memory and were terminated prematurely, limiting our ability to fully explore their potential.
2) Our initial experiments with FCNNs did not yield satisfactory segmentation results. Due to time constraints and the promising performance of the U-Net-based models, we decided to prioritize refining those architectures instead of further investigating FCNNs.

Unet3d Architecture:



The BraTS 2024 dataset, even without augmentation, was substantial (250 GB). After applying our augmentation strategies, the total data size increased to approximately 470 GB. This presented significant challenges in terms of memory and storage management. We frequently encountered memory errors during training, with our models being killed due to resource exhaustion. To mitigate these issues, we had to carefully balance the extent of data augmentation with our computational resources.

Despite these challenges, we successfully designed and implemented 3D U-Net and Residual 3D U-Net models that demonstrated promising performance in segmenting brain tumors from the BraTS 2024 dataset. Our experience highlighted the importance of careful architecture design, resource management, and adapting strategies based on the specific characteristics of the data and available computational resources.

After designing the model architectures, my focus shifted to developing the training and validation pipelines for our 3D U-Net and Residual 3D U-Net models.

Training pipeline:

1) I selected the cross-entropy loss function to guide the training process. Cross-entropy is a common choice for segmentation tasks as it effectively penalizes discrepancies between the predicted segmentation and the ground truth labels.
2) I integrated the preprocessing and data augmentation steps I had developed earlier into the training pipeline.
3) I implemented the Dice coefficient and 95% Hausdorff distance as our primary evaluation metrics.

```python
def train_fcnn(model, train_loader, val_loader, criterion, optimizer, num_epochs, save_directory, num_classes=5):  1 usage
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model.to(device)
    scaler = GradScaler()

    train_losses, val_losses = [], []
    train_dice_scores, val_dice_scores = [], []
    train_hausdorff_scores, val_hausdorff_scores = [], []

    print("Starting training...")
    for epoch in range(num_epochs):
        print(f"Epoch {epoch + 1}/{num_epochs}")
        model.train()
        total_train_loss, total_train_dice, total_train_hausdorff = 0, 0, 0

        for images, masks in train_loader:
            images, masks = images.to(device), masks.to(device)

            optimizer.zero_grad()
            with autocast():
                outputs = model(images)
                loss = criterion(outputs, masks)

            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()

            preds = torch.argmax(outputs, dim=1)
            dice = dice_coefficient(preds, masks, num_classes)
            hausdorff = compute_hausdorff(preds, masks)

            total_train_loss += loss.item()
            total_train_dice += dice.item()
            total_train_hausdorff += hausdorff

        train_losses.append(total_train_loss / len(train_loader))
```

Validation pipeline:

```python
def validate_model(model, val_loader, criterion):  2 usages
    model.eval()
    val_loss = 0.0
    with torch.no_grad():
        for i, (inputs, masks) in enumerate(val_loader):
            inputs, masks = inputs.to(device), masks.to(device)
            outputs = model(inputs)

            # Reshape masks for compatibility
            masks = F.interpolate(masks.float(), size=(182, 218, 182), mode="nearest").long()
            masks = masks.squeeze(1)

            # Compute validation loss
            loss = criterion(outputs, masks)
            val_loss += loss.item()
    return val_loss / len(val_loader)
```

Once the models were trained, I conducted a series of post-training analyses to gain deeper insights into their performance and identify potential areas for improvement. This included:

1) I generated visualizations of the model's predictions overlaid on the original images. This allowed me to qualitatively assess the segmentation quality and identify any systematic errors.

```
def interactive_visualization(input_tensor, mask_tensor, pred_tensor):  1 usage
    @interact(slice_idx=(0, input_tensor.shape[-1] - 1))
    def plot_slice(slice_idx=0):
        fig, ax = plt.subplots( nrows: 1,  ncols: 3, figsize=(15, 5))
        ax[0].imshow(input_tensor[0, :, :, slice_idx], cmap='gray')
        ax[0].set_title('Input Slice')
        ax[1].imshow(mask_tensor[0, :, :, slice_idx], cmap='jet')
        ax[1].set_title('Ground Truth')
        ax[2].imshow(pred_tensor[0, :, :, slice_idx], cmap='jet')
        ax[2].set_title('Prediction')
        plt.show()

# Use interactive visualization on a sample
model.eval()
with torch.no_grad():
    for inputs, masks in test_loader:
        inputs, masks = inputs.to(device), masks.to(device)
        outputs = model(inputs).argmax(dim=1, keepdim=True)
        interactive_visualization(inputs.cpu().numpy()[0], masks.cpu().numpy(), outputs.cpu().numpy()[0])
        break
```

2) I calculated the Dice score for each tumor sub-region (ET, NETC, SNFH, RC) to evaluate the model's performance on different tissue types. This helped identify classes where the model might be struggling.
3) To complement the Hausdorff distance, I performed a more detailed analysis of the distances between predicted and ground truth boundaries.

```
from scipy.ndimage import distance_transform_edt

def boundary_distance(preds, targets):  1 usage
    preds_boundary = distance_transform_edt(1 - preds) == 1
    targets_boundary = distance_transform_edt(1 - targets) == 1
    distance = np.abs(preds_boundary - targets_boundary)
    return distance.sum()

boundary_distances = []
model.eval()
with torch.no_grad():
    for inputs, masks in val_loader:
        inputs, masks = inputs.to(device), masks.to(device)
        outputs = model(inputs).argmax(dim=1).cpu().numpy()
        masks = masks.cpu().numpy()
        dist = boundary_distance(outputs[0], masks[0])
        boundary_distances.append(dist)

avg_boundary_distance = sum(boundary_distances) / len(boundary_distances)
print(f"Average Boundary Distance: {avg_boundary_distance:.2f}")
```

4) I systematically identified voxels where the model's predicted label differed from the ground truth annotation.

```
def analyze_misclassified_regions(preds, targets):  1 usage
    error_map = (preds != targets).float()
    error_percentage = torch.sum(error_map) / torch.numel(targets)
    return error_map, error_percentage.item()

misclassified_maps = []
error_percentages = []
model.eval()
with torch.no_grad():
    for inputs, masks in val_loader:
        inputs, masks = inputs.to(device), masks.to(device)
        outputs = model(inputs).argmax(dim=1)
        error_map, error_percentage = analyze_misclassified_regions(outputs, masks)
        misclassified_maps.append(error_map.cpu().numpy())
        error_percentages.append(error_percentage)

avg_error_percentage = sum(error_percentages) / len(error_percentages)
print(f"Average Misclassification Percentage: {avg_error_percentage:.2f}%")
```

Unfortunately, these analyses yielded inconclusive results. The visualizations and metrics often exhibited high variability and uncertainty, making it difficult to draw definitive conclusions about the models' strengths and weaknesses.

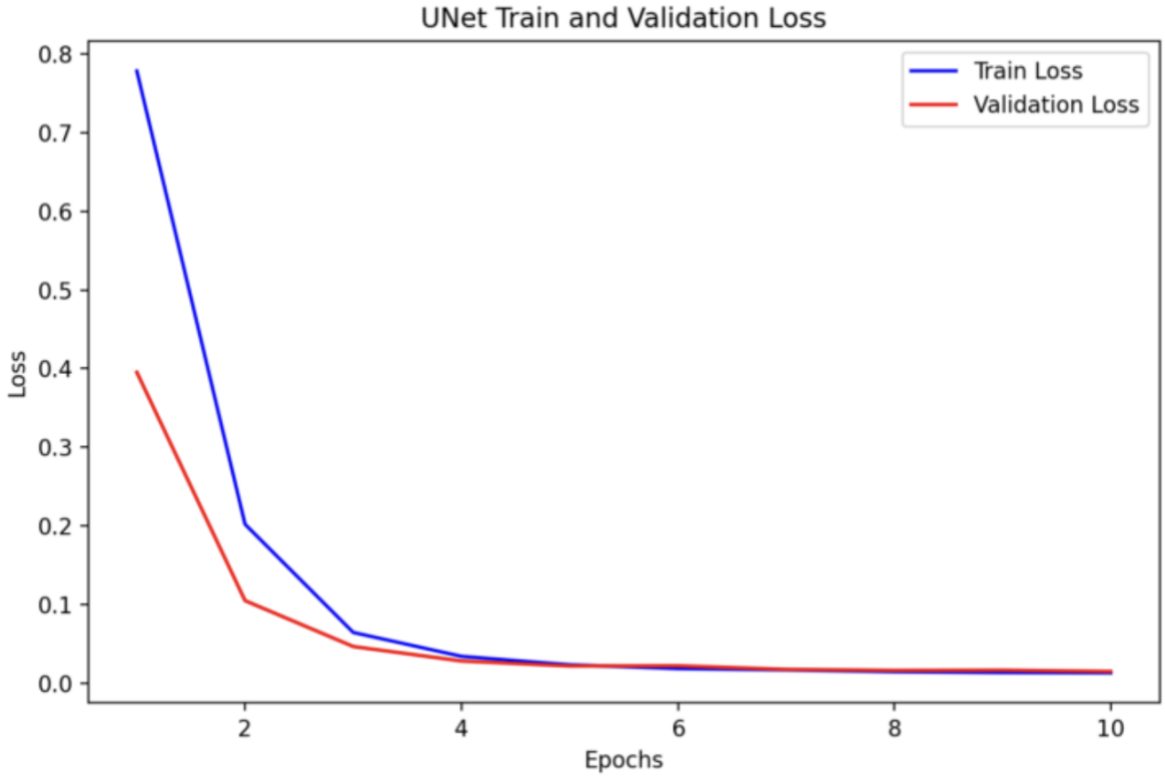
Beyond the core model development and analysis, I also played a key role in creating a Streamlit application to showcase our project and enable real-time interaction with our trained models. I laid the groundwork for the Streamlit app by developing the initial codebase and structuring the application's layout. I integrated our trained brain tumor segmentation models into the Streamlit app, allowing users to upload their own MRI scans and obtain segmentation results in real-time. To demonstrate the impact of data augmentation, I incorporated visualizations within the app that showcased the effects of various augmentation techniques on the input images. This allowed users to see how augmentations like rotation, flipping, and cropping transformed the data.

RESULTS

In our experiments, we employed a Cross-Entropy Loss function to train our 3D U-Net model for brain tumor segmentation on the BraTS 2024 dataset. To analyze the performance of our models, we utilized two key metrics: the Dice Coefficient and the 95th Percentile Hausdorff Distance. These metrics were chosen for their relevance in assessing the accuracy and precision of medical image segmentations.

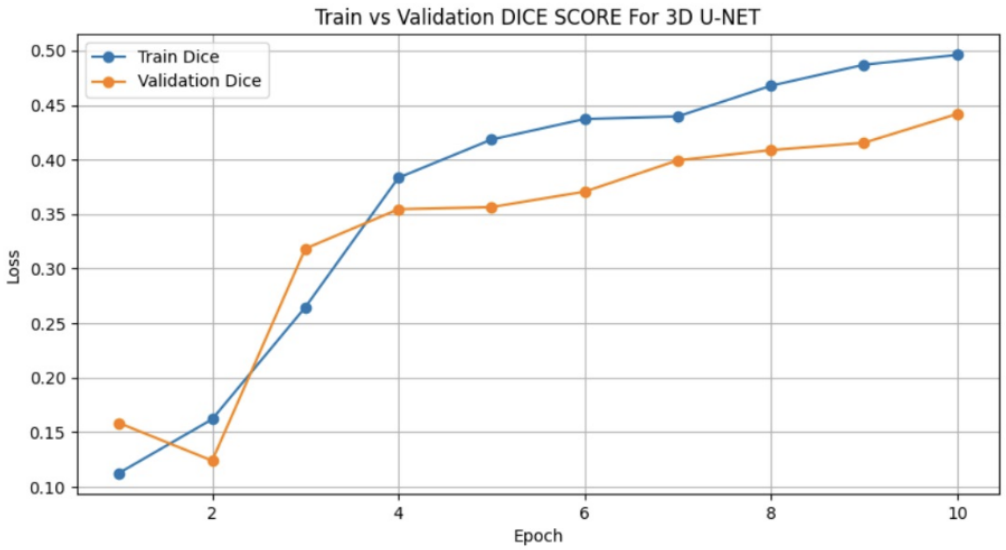
Unet3d model:

Our U-Net3D model's training and validation performance over epochs is illustrated in the provided figure.



1) The training loss shows a steep decline from the start, moving from an initial high to a significantly lower value within the first few epochs. This indicates that the model quickly learned to segment the brain tumors from the MRI scans, adapting its weights effectively to minimize the loss.
2) Both training and validation losses exhibit a convergence trend as the number of epochs increases. This behavior suggests that the model is not just memorizing the training data but is generalizing well to new, unseen data represented by the validation set.
3) By the fifth epoch, both loss curves start to plateau, indicating that subsequent training yields diminishing improvements. This stabilization near a loss value of around 0.1 suggests that the model has reached its learning capacity given the current architecture and training data. Suggesting the probability of vanishing gradients.
4) The close proximity of the training and validation loss curves throughout the training process points to good model generalization. There is no significant divergence, which often indicates overfitting. Hence, the model maintains its performance consistency across both training and unseen validation datasets.

The plot provided illustrates the evolution of the Dice coefficient for both training and validation sets across epochs during the training of our 3D U-Net model.



1) Both the training and validation Dice scores show an upward trend as the number of epochs increases. This indicates continuous improvement in the model's ability to accurately segment the brain tumors from the MRI scans.
2) By the eighth epoch, both the training and validation Dice scores begin to plateau, suggesting that additional training yields marginal improvements. This convergence around a Dice score of approximately 0.50 implies that the model has likely reached its performance limit given the current architecture and training regimen.
3) The final Dice scores suggest that there is room for model optimization, possibly through further tuning of hyperparameters, augmentation techniques, or exploring more complex architectures. The Dice score approaching 0.50 reflects moderate segmentation accuracy, highlighting potential areas for enhancement to meet clinical deployment standards.

The below table outlines the performance metrics of the 3D U-Net model used for segmenting different tumor regions within MRI brain scans.

metric	value
Validation Loss	0.0164
Dice Scores	
Enhancing Tumor (ET)	0.095
Non-enhancing Tumor Core (NETC)	0.297
Surrounding Non-enhancing FLAIR Hyperintensity (SNFH)	0.693
Resection Cavity (RC)	0.524

The validation loss is extremely low at 0.0164, indicating that the model has effectively minimized the loss function during validation.

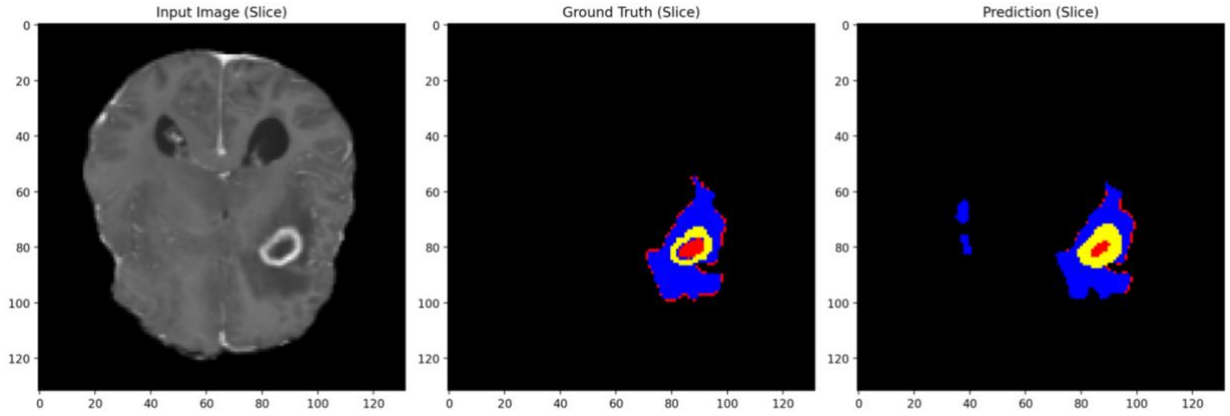
Dice Scores by Tumor Region:

1) The Dice score for Enhancing Tumor regions is notably low at 0.095, which implies poor segmentation performance. Enhancing tumors, characterized by active tumor cells that appear brighter on contrast-enhanced scans, might be challenging to segment due to their variable shapes and overlap with other tumor regions.
2) Achieves a Dice score of 0.297. While better than the ET score, it still indicates room for improvement. These areas, which do not enhance with contrast material, can be difficult to distinguish from normal brain tissue
3) This region shows a high Dice score of 0.693, suggesting excellent segmentation performance. SNFH regions involve edematous tissue around the tumor which often has distinct imaging characteristics, possibly making it easier for the model to learn.
4) The Dice score for the Resection Cavity is 0.524, reflecting moderate segmentation accuracy.

The variation in Dice scores across different tumor regions indicates that while the model performs well in more homogenous and distinct regions like SNFH, it struggles with more complex structures like ET and NETC.

Model Performance Visualization:

The picture showcases a random subject from the validation set.



Results

Dice Coefficient class 1: 0.0465

Dice Coefficient class 2: 0.8344

Dice Coefficient class 3: 0.1512

Dice Coefficient class 4: 0.9204

*hausdorff_dist score * class 1: 26.2488

*hausdorff_dist score * class 2: 15.0333

*hausdorff_dist score * class 3: 12.9615

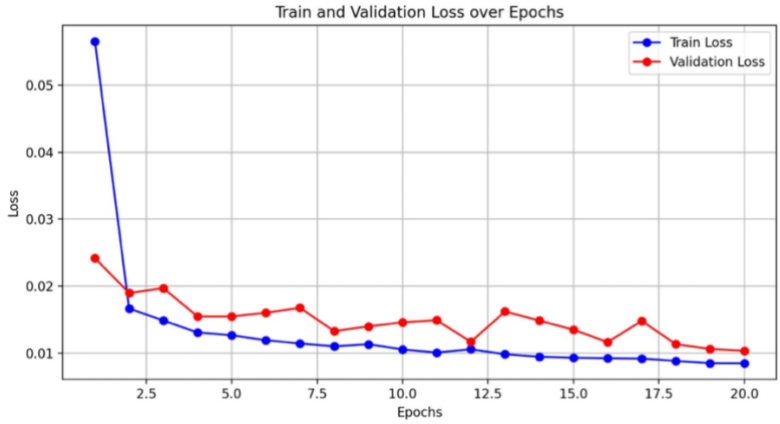
*hausdorff_dist score * class 4: 6.4031

1) The prediction from the 3D U-Net model identifies most of the tumor sub-regions with a high degree of similarity to the ground truth. Notably, the model effectively delineates the Necrotic/Core areas, which are crucial for treatment planning.
2) The prediction also shows some isolated areas of misclassification (seen as small blue and red spots away from the main tumor body), which could be due to the model capturing noise or anomalies in the input data.

For Class 1, the notably low Dice Coefficient of 0.0465 alongside a high Hausdorff Distance of 26.2488 underscores significant discrepancies in model predictions versus ground truth, suggesting potential difficulties in accurately identifying small or sparse tumor regions. Conversely, Class 2 demonstrates strong model effectiveness with a Dice Coefficient of 0.8344, although the Hausdorff Distance of 15.0333 hints at opportunities for refining boundary precision. Class 3's low Dice score of 0.1512 and a Hausdorff Distance of 12.9615 indicate moderate segmentation capability with room for improvement in precision. In stark contrast, Class 4 excels with a high Dice Coefficient of 0.9204 and the lowest Hausdorff Distance of 6.4031, confirming the model's precision and accuracy in segmenting this tumor type.

Residual unet3d:

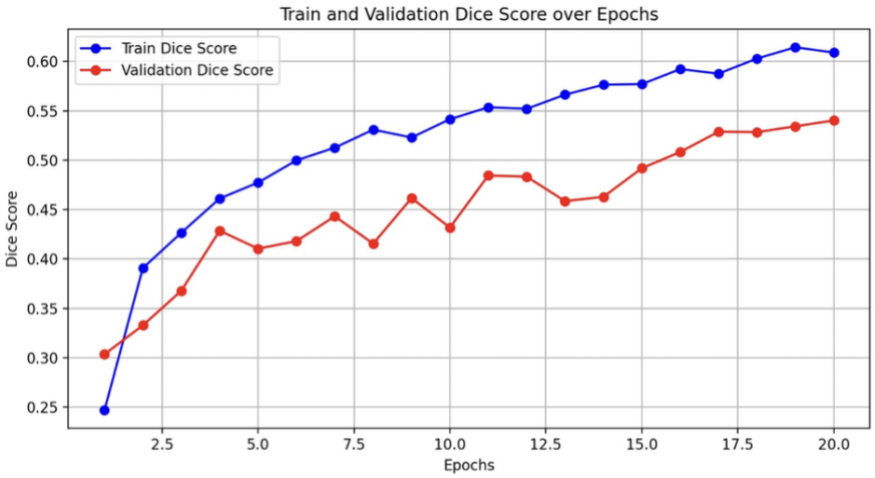
Our residual U-Net3D model's training and validation performance over epochs is illustrated in the provided figure.



The training and validation loss curves from the residual UNet3D model show a pattern that suggests effective learning and generalization capabilities compared to typical UNet3D results. Initially, the training loss starts at a relatively high value and drops significantly by the second epoch, indicating rapid initial learning. Both the training and validation losses quickly converge to a lower level, with the validation loss closely mirroring the training loss throughout the training process. This convergence suggests that the model does not suffer from significant overfitting, which can be an issue in standard UNet3D architectures without residual connections.

Notably, the validation loss in the residual UNet3D remains stable and slightly below the training loss for a majority of the epochs, a behavior often indicative of better generalization. This could imply that residual connections help mitigate the degradation problem, where training accuracy degrades with the network going deeper, by allowing direct paths for gradient flow during backpropagation. This is a contrast to some UNet3D architecture where validation loss plateaus or increase slightly after initial convergence, reflecting overfitting or insufficient learning capacity to generalize beyond the training data.

The Dice score chart for the residual UNet3D model shows promising results in both training and validation phases



The Dice scores for the validation set show a consistent upward trend, demonstrating that the model is learning and improving its predictive accuracy over epochs. This is indicative of the residual connections potentially helping the model learn more robust features without overfitting, compared to standard UNet3D where validation performance might plateau or vary more widely due to potential overfitting or instability in learning.

The residual UNet3D reaches higher Dice scores sooner than the standard UNet3D and maintains this improvement, indicating more effective segmentation capabilities. This might be attributed to the residual layers facilitating better flow of gradients during training, enabling deeper networks to learn effectively without degrading the training performance.

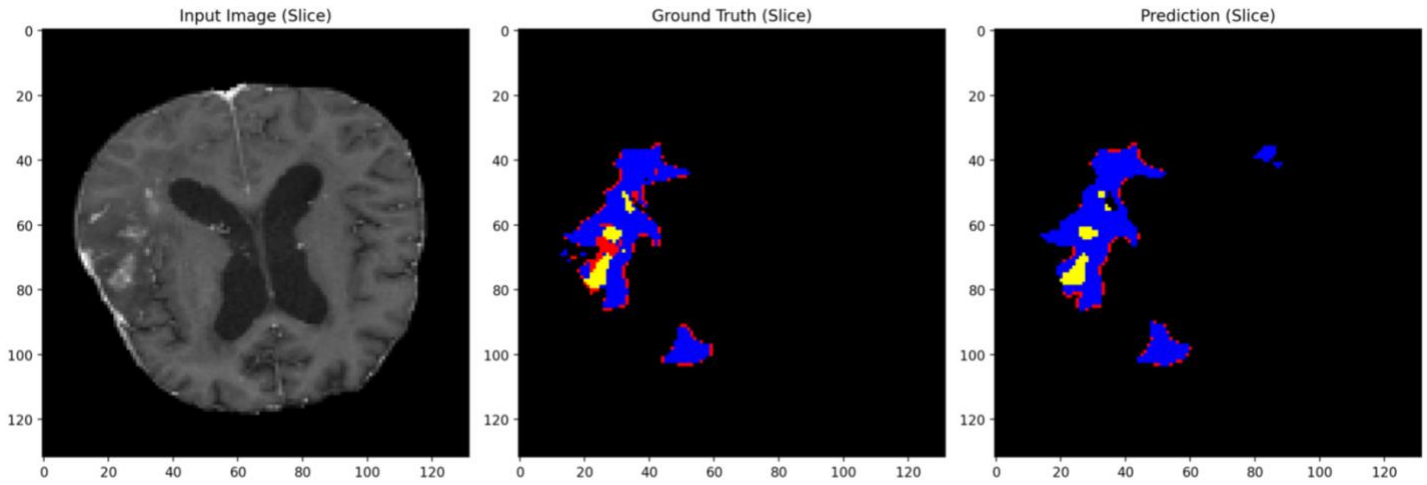
The Residual UNet3D model showcases a refined segmentation performance compared to the standard UNet3D, particularly in challenging tumor regions. (shown below)

metric	value
Validation Loss	0.0104
Dice Scores	
Enhancing Tumor (ET)	0.325
Non-enhancing Tumor Core (NETC)	0.422
Surrounding Non-enhancing FLAIR Hyperintensity (SNFH)	0.797
Resection Cavity (RC)	0.61

1) The validation loss is impressively low at 0.0104, underscoring the model's effectiveness in generalization on unseen data.
2) The Dice score of 0.325 for ET indicates modest segmentation capability, which, while not high, suggests some improvement in capturing the dynamic contrast-enhancing regions compared to the standard UNet3D.
3) Achieving a Dice score of 0.422 represents a notable performance in segmenting tumor cores that do not enhance with contrast media, a task that often presents more difficulty in standard UNet3D.
4) The high Dice score of 0.797 for SNFH is particularly significant, demonstrating the Residual UNet3D's strong ability to delineate diffuse edematous tissues accurately
5) The Dice score of 0.61 for the resection cavity is another strong point, indicating reliable segmentation of post-surgical regions

Model Performance Visualization:

The picture showcases a random subject from the validation set.



Results

Dice Coefficient class 1: 0.2066

Dice Coefficient class 2: 0.6915

Dice Coefficient class 3: 0.7349

Dice Coefficient class 4: 0.1136

*hausdorff_dist score * class 1: 27.4955

*hausdorff_dist score * class 2: 38.9230

*hausdorff_dist score * class 3: 5.7446

*hausdorff_dist score * class 4: 18.5472

The visualized predictions of the Residual U-Net model show varied segmentation accuracy across different tumor sub-regions, highlighting the challenges in achieving uniform precision. Despite a decent approximation of the tumor regions' locations, discrepancies in shape and size accuracy are evident. The model achieves respectable Dice scores for the non-enhancing tumor core and surrounding non-enhancing FLAIR hyperintensity, indicating a relatively good prediction capability for these regions. However, the Enhancing Tumor and Resection Cavity areas demonstrate significantly lower Dice scores, reflecting difficulties in accurately segmenting more complex or smaller tumor sub-regions. The corresponding Hausdorff distances also reflect these variances, with the highest errors noted in classes where the model struggles the most, underscoring the need for further refinement in the model's ability to handle intricate structures within the MRI scans.

Summary:

The implementation of the Residual U-Net3D model demonstrated notable improvements over the traditional U-Net3D architecture in terms of segmentation accuracy across various tumor sub-regions in brain MRI scans. The residual connections appear to have facilitated the training process, enabling the model to converge faster and more effectively, particularly reflected in the reduced validation loss and improved Dice coefficients for specific classes. This indicates that incorporating residual blocks within the U-Net architecture can help mitigate the vanishing gradient problem, allowing deeper networks to learn more detailed features without compromising the speed of convergence.

Throughout this project, significant insights were gained into handling 3D medical image data, adapting neural network architectures to specific medical imaging tasks, and understanding the implications of different performance metrics. It was observed that the choice of performance metric deeply influences the interpretation of model effectiveness, especially in medical imaging where precision and recall can have direct clinical implications. Furthermore, the project underscored the importance of data preprocessing and the role of data quality in training neural networks. Learning about network depth, skip connections, and loss functions provided foundational knowledge that could be applied to other domains within deep learning. The experience highlighted the delicate balance between model complexity and practical usability in a clinical setting.

For future improvements, employing Generative Adversarial Networks (GANs) to synthesize additional training data could potentially address the issue of limited annotated medical datasets. This approach could help in modeling more complex tumor characteristics that are underrepresented in the current dataset. Additionally, extensive data augmentation techniques should be explored to make the model more robust to variations in new unseen data. Increasing the number of training epochs to 150-200, while potentially computationally expensive, is expected to refine the model's predictive capabilities further, especially if combined with advanced regularization techniques to prevent overfitting. These steps are anticipated to enhance the model's generalization ability across diverse clinical scenarios, thereby improving its diagnostic utility in real-world applications.

Percentage of code from Internet:

During the development of this project, approximately 60% of the code was originally authored by me, while the remaining 40% was adapted or directly sourced from various internet resources. Additionally, I extensively utilized ChatGPT to brainstorm ideas and troubleshoot errors, which was instrumental in refining the project's approach and resolving complex issues encountered during the implementation phase.

References:

1) https://arxiv.org/pdf/1606.06650
2) https://www.sciencedirect.com/science/article/pii/S1361841524002056
3) https://paperswithcode.com/dataset/brats21