Brain Tumor Segmentation

Team Members:

- Ujjawal
- Bala Krishna
- Bhagawath Sai
- Sai Avinash

Project Relevance

Brain tumor segmentation is a crucial task in medical imaging that assists in the diagnosis and treatment of patients. Accurate segmentation helps radiologi
regions, identify malignancies, and monitor treatment progress. The project aims to implement deep learnoing models (3D FCNNs, Unet, Residual Unet) to s
from multi-modal MRI scans.

Dataset Description

This project utilizes the BraTS dataset, which contains pre-processed multi-modal MRI scans for brain tumor segmentation. The dataset includes the followi

- T1-weighted (t1n)
- T1-weighted post-contrast (t1c)
- T2-weighted (t2w)
- Fluid-attenuated inversion recovery (t2f)

Types of Scans and Tumor Regions

The dataset includes annotations for various tumor regions:

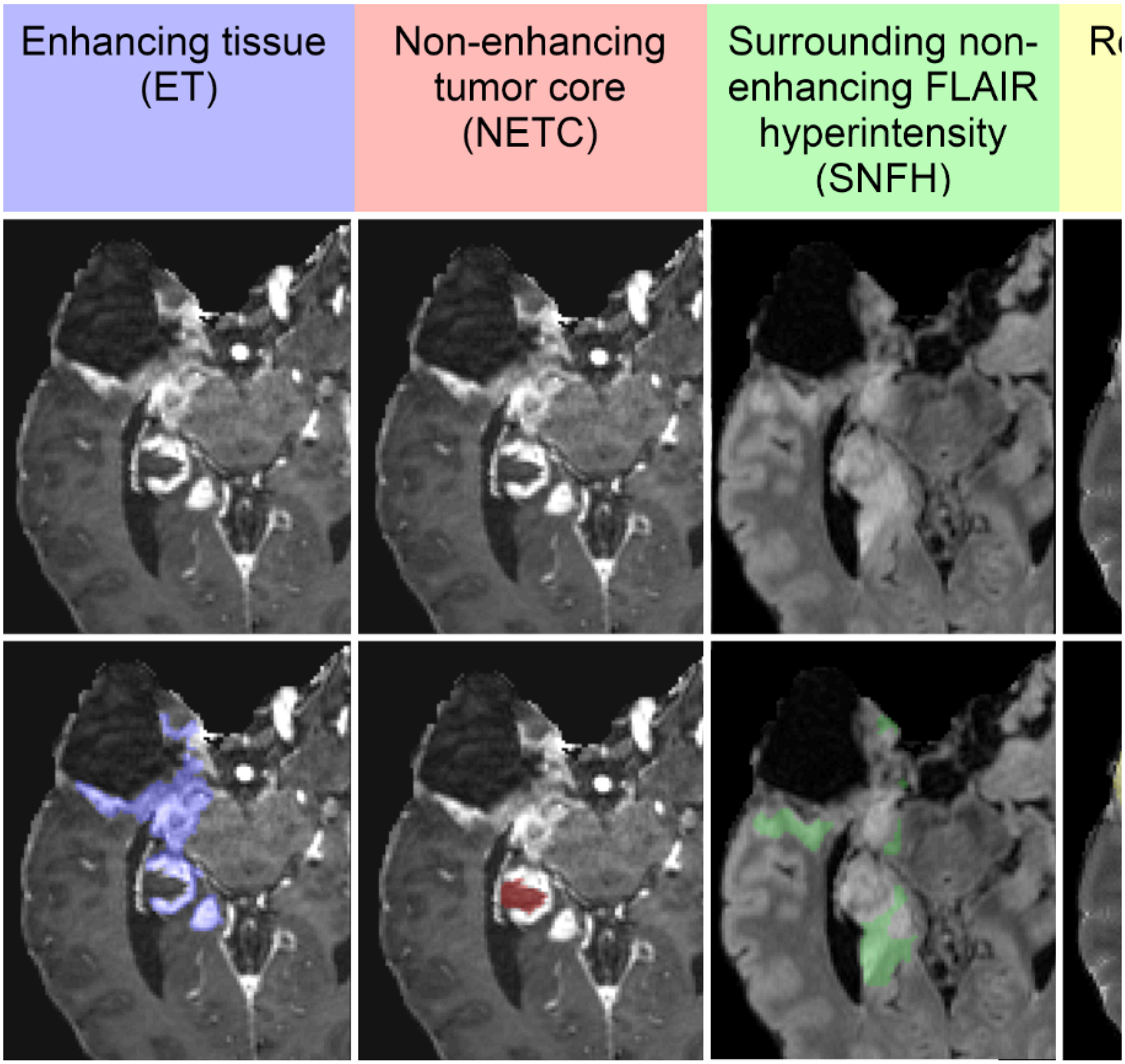
- *Whole Tumor (WT)*: All visible tumor regions.
- *Necrotic and Non-enhancing Tumor Core (NETC)*: Central dead tissue and non-enhancing regions.
- *Enhancing Tumor (ET)*: Actively growing tumor cells.
- *Surrounding Non-tumor Fluid Heterogeneity (SNFH)*: Swelling or fluid accumulation around the tumor.
- *Resection Cavity*: Post-surgical void in the brain.

Select Tumor Region:

[✓ Whole Tumor (WT)]
Necrotic and Non-enhancing Tumor Core (NETC)
Enhancing Tumor (ET)
Surrounding Non-tumor Fluid Heterogeneity (SNFH)
Resection Cavity

Description of Selected Region: All visible tumor regions.

Tumor Sub-region Image


Enhancing tissue (ET)
Non-enhancing tumor core (NETC)
Surrounding non-enhancing FLAIR hyperintensity (SNFH)
R

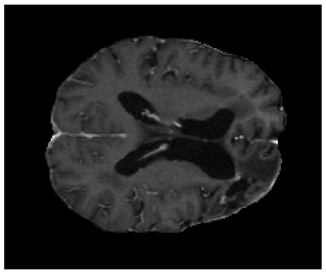
Tumor Sub-region Image

Image Viewer

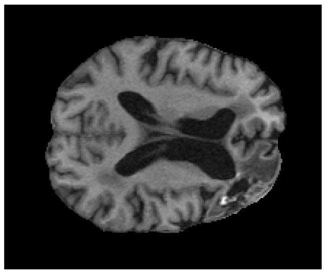
Select a view category

Axial

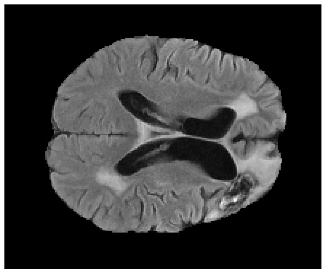
Category: t1c



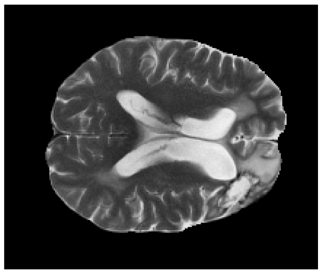
Category: t1n



Category: t2f



Category: t2w



Shape of the Raw image: `(182, 218, 182)`

Image Resgistration

```
for mod in self.modalities:
        mod_path = os.path.join(self.image_dir, f'{patient_id}_{mod}.nii')
        image = load_image(mod_path)

        # Keep the original image
        original_image = resize_image(image, self.target_size)
        original_image_np = sitk.GetArrayFromImage(original_image)
        normalized_original = normalize_image(original_image_np)
        images_original.append(normalized_original)

        # Apply transformations
        transformations = apply_transformations(image)
        transformed_image = random.choice(list(transformations.values()))
        transformed_image = resize_image(transformed_image, self.target_size)
        transformed_image_np = sitk.GetArrayFromImage(transformed_image)
        normalized_transformed = normalize_image(transformed_image_np)
        images_transformed.append(normalized_transformed)

    # Stack original and transformed images
    image_stack_original = np.stack(images_original, axis=0)
    image_stack_transformed = np.stack(images_transformed, axis=0)
```

Image Augmentation Viewer

Select a directory

BraTS-GLI-02973-100

Select a view category

Axial

Select the type of augmentation

None

Apply Transformation

Medical Image Segmentation for BRATS 2024

Modeling Brain Tumors with 3D U-Net and Residual 3D U-Net

Why 3D U-Net for the BRATS 2024 dataset?

- **Contextual Preservation:**
- **Precise localization:**

Understanding U-Net Architecture

U-Net architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. The key components

- **Contracting Path:** Captures the context in the image, which helps in understanding the global structure of the brain and the tumor.
- **Expanding Path:** Allows for precise localization using transposed convolutions to recover spatial resolution.

- **Skip Connections:** Provide essential high-resolution features to the expanding path, improving the accuracy of segmentation.

Implementing U-Net for BRATS 2024

```
class DoubleConv(nn.Module):
    """(convolution => [BN] => ReLU) * 2"""

    def __init__(self, in_channels, out_channels, mid_channels=None):
        super().__init__()
        if not mid_channels:
            mid_channels = out_channels
        self.double_conv = nn.Sequential(
            nn.Conv3d(in_channels, mid_channels, kernel_size=3, padding=1),
            nn.BatchNorm3d(mid_channels),
            nn.ReLU(inplace=True),
            nn.Conv3d(mid_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm3d(out_channels),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.double_conv(x)
        ..............

        class UNet3D(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=True):
        super(UNet3D, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = DoubleConv(n_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        self.down4 = Down(512, 512)
        self.up1 = Up(1024, 256, bilinear)
        self.up2 = Up(512, 128, bilinear)
        self.up3 = Up(256, 64, bilinear)
        self.up4 = Up(128, 64, bilinear)
        self.outc = OutConv(64, n_classes)

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits
```

Evaluating Model Performance

Model performance is assessed using the Dice coefficient, which measures the overlap between the predicted segmentation and the ground truth labels. Thi
effectiveness of the model in medical segmentation tasks.

```
def dice_coefficient(preds, targets, num_classes):
    # Compute Dice Coefficient
    ...
    return dice / (num_classes - 1)
```

Data Preprocessing and Augmentation

Effective data preprocessing and augmentation are crucial for training robust models. Here's how data is prepared:

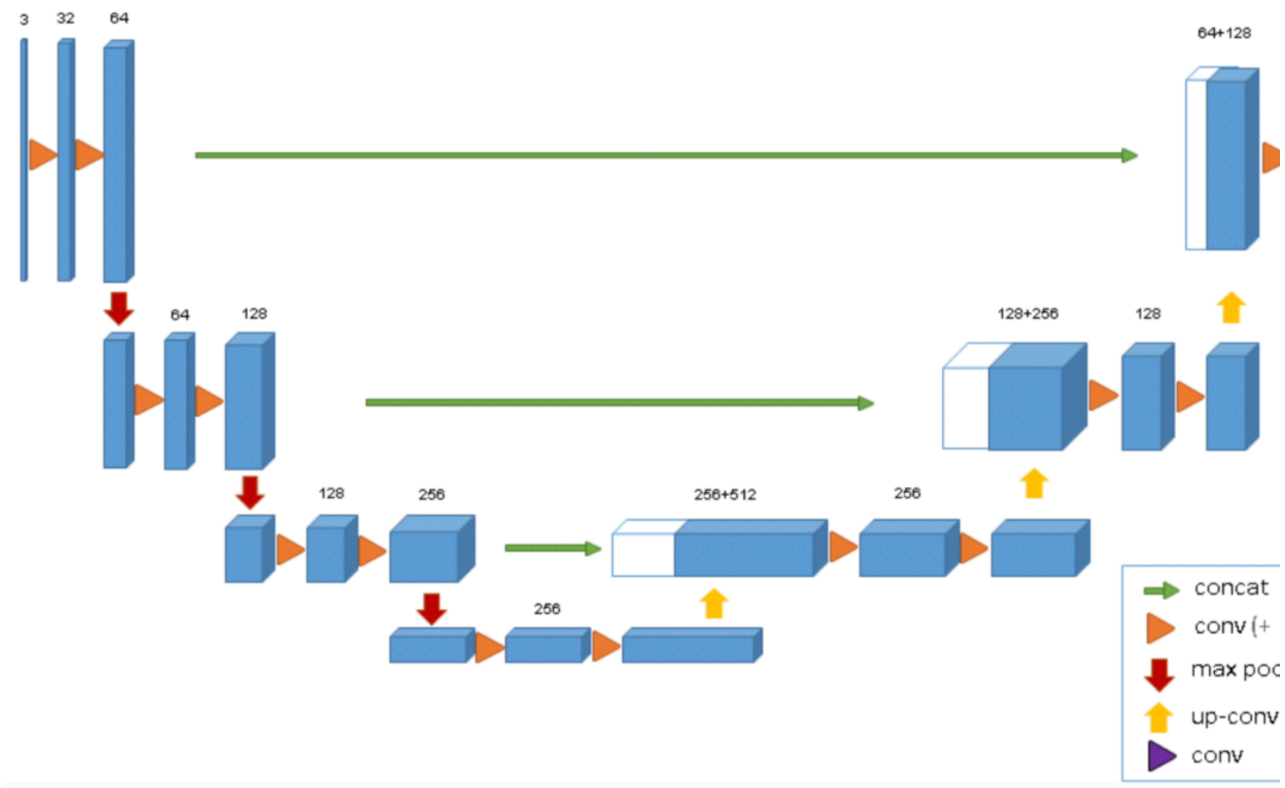
- **Normalization**: Ensures that MRI scans have similar intensity ranges, enhancing model training stability.
- **Augmentation**: Includes zooming, cropping, and rotation, which helps the model generalize better by presenting varied examples during training.

```
transformations = apply_transformations(image)
```

Additional Resources and References

- Original U-Net Paper
- 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation
- TensorFlow U-Net Tutorial

3D u net Architecture



3d-UNET Architecture

Select an NII file for prediction with 3D UNET

BraTS-GLI-02759-101_t2f.nii