Final Project

Introduction to Imaging AI with Applications in Medical Imaging

Anton ZAITSEV
Emmanouil F. PAPADIMITRIOU
Marie LONTSIE

UNIVERSITÉ DU LUXEMBOURG

Task I

Sampling and Fourier Transform of Grayscale Images

Error-free Image Reconstruction From a Digital Image

- The theoretical maximum spatial frequencies $f_{u_{max}}$ and $f_{v_{max}}$ that can be represented without aliasing, based on the sampling rates provided.
- This can be solved using the **Nyquist-Shannon sampling theorem**, which states that the maximum representable frequency is half the sampling frequency.
- So, the sampling rate in the x-direction and y-direction respectively:

$$\triangle x = \frac{1}{20} \frac{samples}{mm} \qquad \triangle y = \frac{1}{10} \frac{samples}{mm}$$

Error-free Image Reconstruction From a Digital Image

- Based on the Nyquist-Shannon theorem states the maximum frequency that can be represented without aliasing is:

$$f_{max} = 2\triangle$$

- Thus we get the theoretical maximum values:

$$f_{u_{max}} = 2\triangle x = \frac{1}{10}\,\frac{cycles}{mm} \quad \text{and} \quad f_{v_{max}} = 2\triangle y = \frac{1}{5}\,\frac{cycles}{mm}$$

Memory Requirement for a Color Image

- **Sampling Rate**: 20 samples/mm in the x-direction & 10 samples/mm in the y-direction.
- **Image Dimensions**: assume the image dimensions are W×H mm.
- **Color Channels**: a color image typically has three channels: Red, Green, and Blue (RGB).
- **Bit Depth**: each channel can distinguish 1024 values. Bit depth per channel=log_2(1024)=10 bits.

Memory Requirement for a Color Image

- **Number of Sampling Points** = (20 samples/mm) $\cdot$ (10 samples/mm) $\cdot$ W $\cdot$ H=200 $\cdot$ W $\cdot$ H samples
- **Memory Per Sample** = 3 $\cdot$ 10=30bits = 3.75 bytes
- **Minimum Memory Requirement** = (number of sampling points) $\cdot$ (memory per sample in bytes) = 750 $\cdot$ W $\cdot$ H bytes
- Since we are at a 10 bits RGB image case, the only known format that easily deals with this in a **lossless manner** is **TIFF** (can go to 3×16 bits for the 3-channel color image).

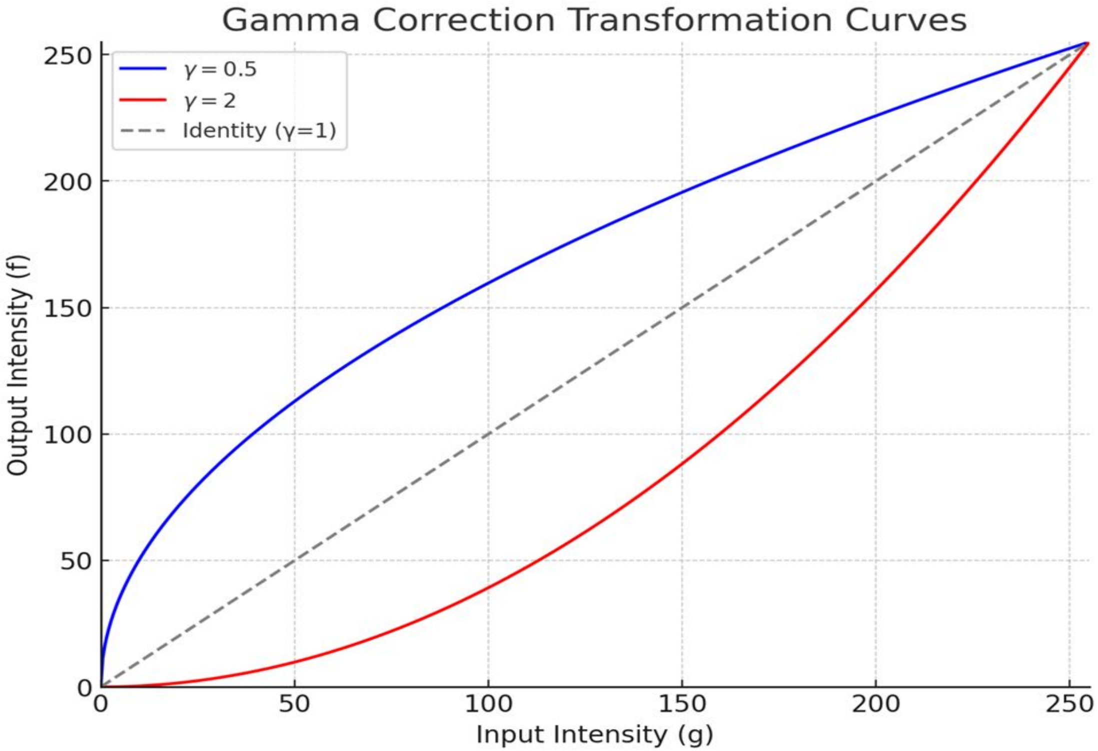
Number of Colors Based on Quantization

- Since there are 1024 potential values per color channel, there are as many color representations as the number of value combinations.
- So, **Number of Combinations** = 1024^3 = 1,073,741,824 colors represented with the quantization we chose.

Task II

Image Enhancement Using Gamma Correction

Gamma Correction Transformation Curves



How to Determine the Coefficient C

- The constant **c** is a scaling constant that ensures our outputs remain in the [0,255] range.
- Therefore, **c** determined as:

$$c = 255^{1-\gamma} \quad \text{if } g,f \in [0,255]$$

Gamma Values on Image Enhancement

- For γ_1, T_G increases the brightness of the image. Specifically, the transformation maps lower input values to higher output values, enhancing shadow details.
- For γ_2, T_G decreases the brightness of the image. For $\gamma > 1$, the transformation maps higher input values to lower output values, enhancing contrast in bright regions.

Minimum Slope of Transform Function

In general:

- For $\gamma>1$ (compression): The slope decreases as $g \to 0$. Minimum slope occurs near $g = 0$.
- For $\gamma<1$(spread): The slope increases as $g \to 0$. Minimum slope occurs near $g=255$.

Minimum Slope for Gamma < 1

- Minimum slope for grey value **spread** (γ<1):

$$\frac{df}{dg} = 255 \cdot 0.5 \cdot \left(\frac{g}{255}\right)^{-0.5}$$

- The slope decreases as g→255 and the **minimum slope occurs at g=255**:

$$\left.\frac{df}{dg}\right|_{g=255} = 255 \cdot 0.5 \cdot (1)^{-0.5} = 127.5$$

Minimum Slope for Gamma > 1

- Minimum slope for grey value **compression** (γ>1):

$$\frac{df}{dg} = 255 \cdot 2 \cdot \left(\frac{g}{255}\right)^{1}$$

- The slope decreases as g→0, and the **minimum slope occurs at g=0**. However, near g=0, the slope asymptotically approaches zero.
- To avoid extreme compression in practical applications, a small positive threshold g (e.g., g=1) is typically used. At g=1:

$$\frac{df}{dg}_{g=1} = 255 \cdot 2 \cdot \left(\frac{1}{255}\right)^{1} \sim 2$$

Task III

Threshold-based Image Analysis

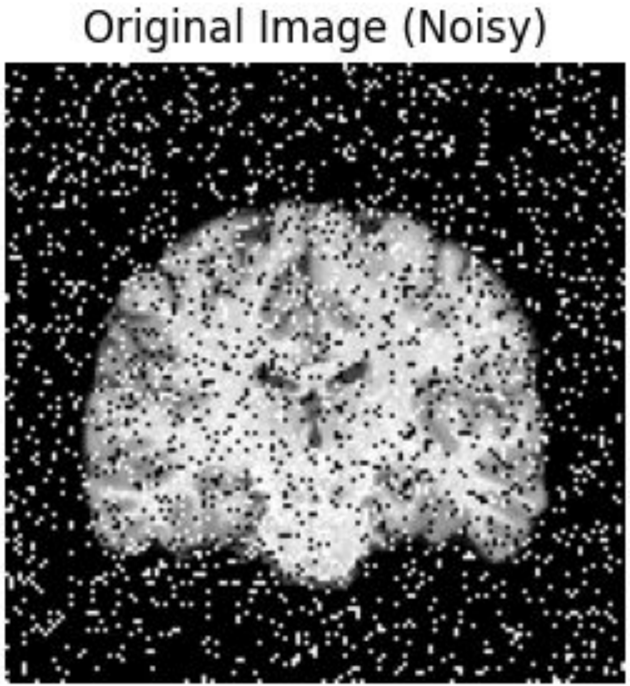
Task Overview

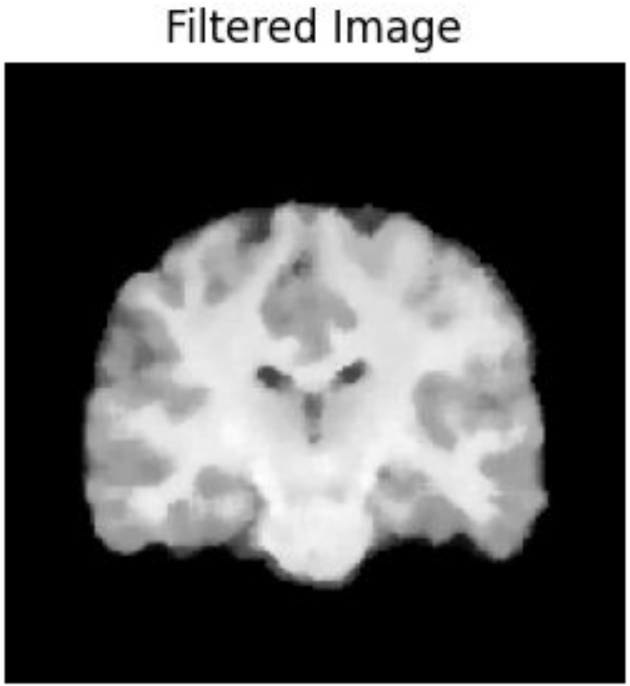
Objective: perform threshold-based segmentation on a brain image to distinguish between background, grey matter, and white matter.

Methods:

- Use of a median filter to remove noise.
- Otsu thresholding for segmentation.
- Bilinear and nearest neighbor interpolation for up-sampling

Median Filter to Denoise the Image



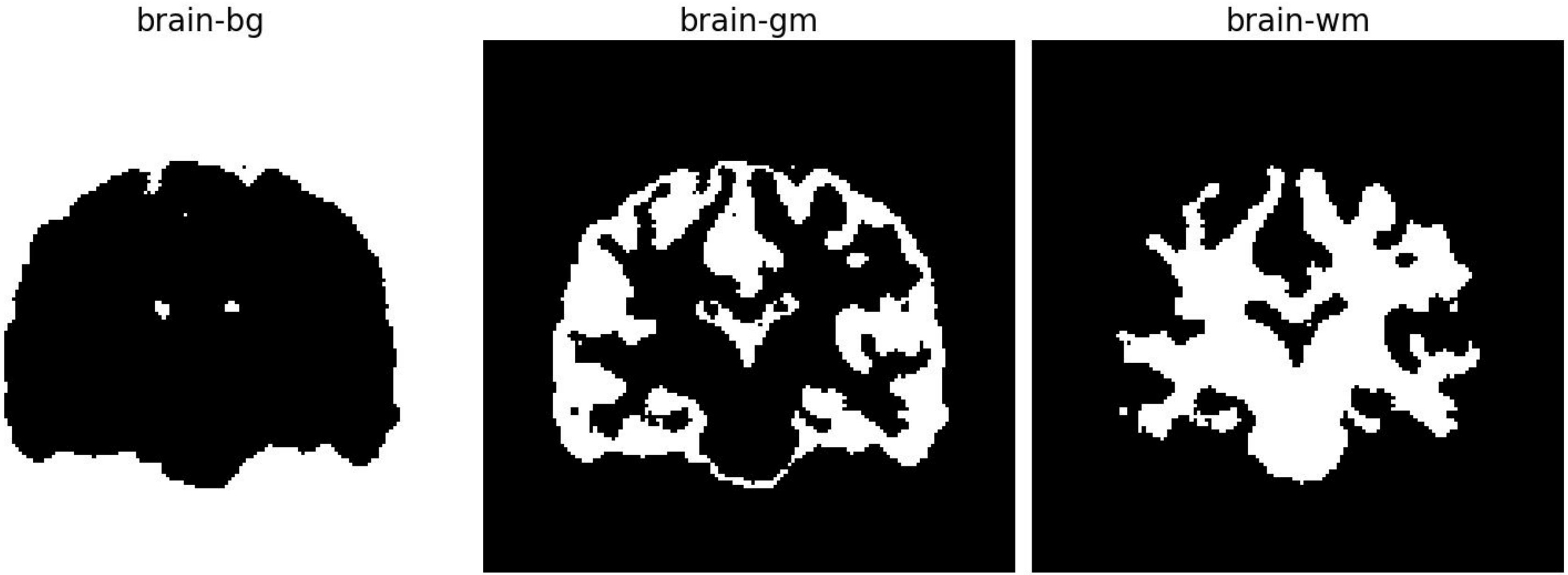


Otsu Thresholding for Segmentation

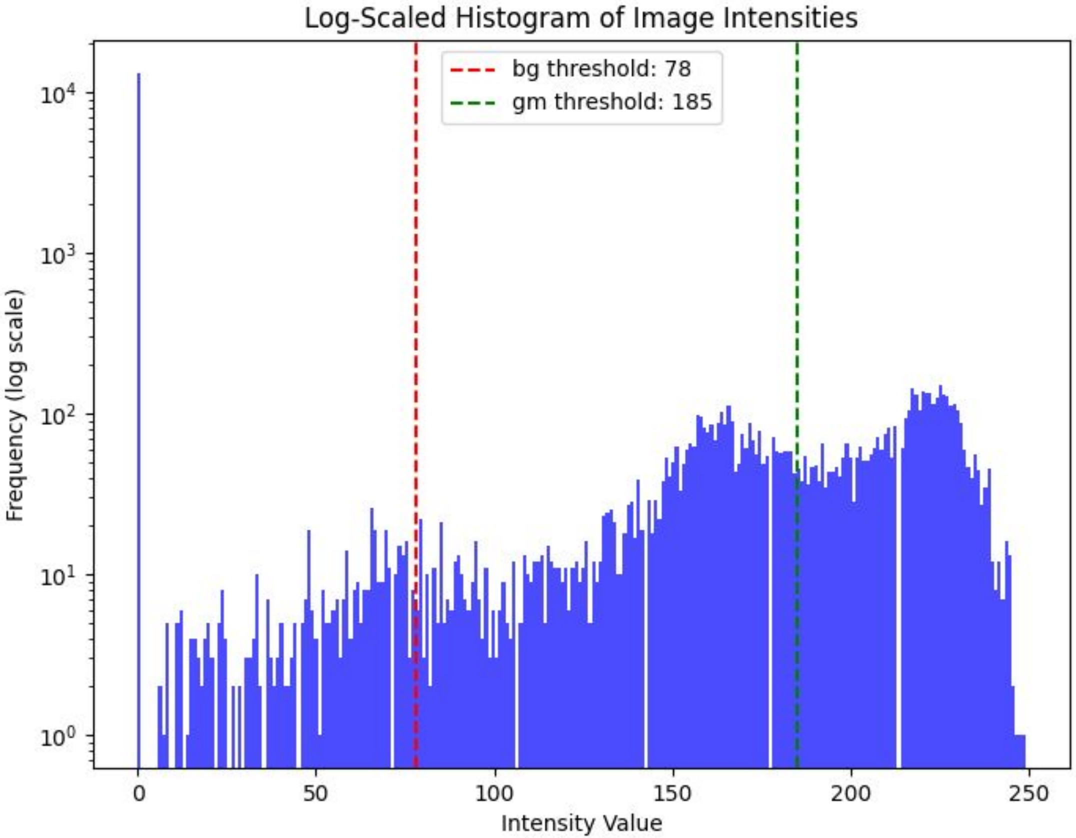
```
# perform multi-Otsu thresholding
thresholds = threshold_multiotsu(filtered_image, classes=3)
background_threshold, gray_matter_threshold = thresholds

# create binary masks
background_mask = filtered_image <= background_threshold
gray_matter_mask = (filtered_image > background_threshold) & (
    filtered_image <= gray_matter_threshold
)
white_matter_mask = filtered_image > gray_matter_threshold
```

Result (Images) - Otsu Thresholding

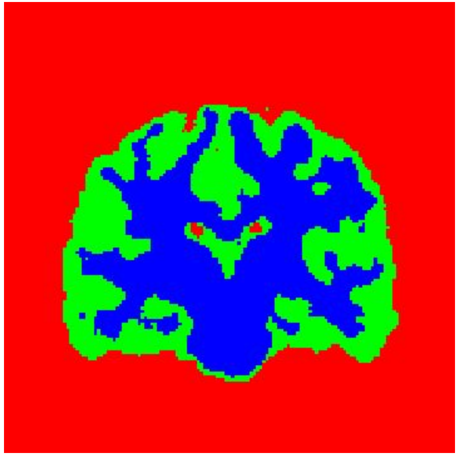


Result (Histogram + Intensities) - Otsu Thresholding



Three Masks Into a Single Image - Otsu Thresholding

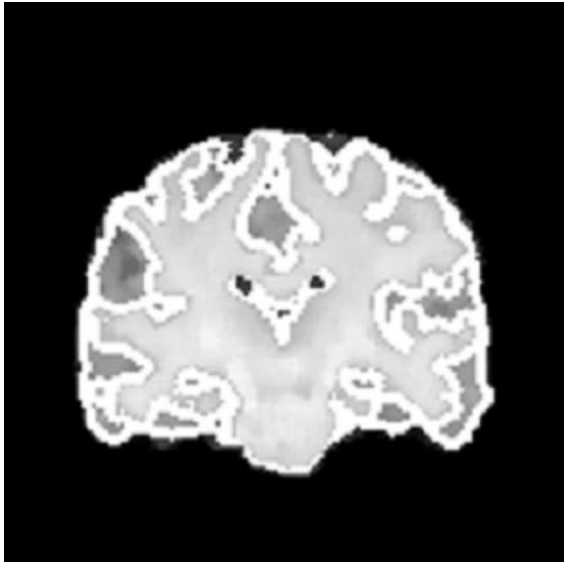
```
# combine masks into a single color image
# background: red, grey matter: green, white matter: blue
colored_image = np.zeros((*filtered_image.shape, 3), dtype=np.uint8)
colored_image[background_mask] = [0, 0, 255]  # red for background
colored_image[gray_matter_mask] = [0, 255, 0]  # green for gray matter
colored_image[white_matter_mask] = [255, 0, 0]  # blue for white matter
```



- Background in red, gray matter in green, and white matter in blue.

Erosion Filter

```
# create a boundary between grey and white matter
# using erosion and overlay it on the denoised image
boundary = erosion(
    gray_matter_mask.astype(np.uint8), disk(3)
) ^ gray_matter_mask.astype(np.uint8)
overlay_image = filtered_image.copy()
overlay_image[boundary.astype(bool)] = 255 # red boundary
```



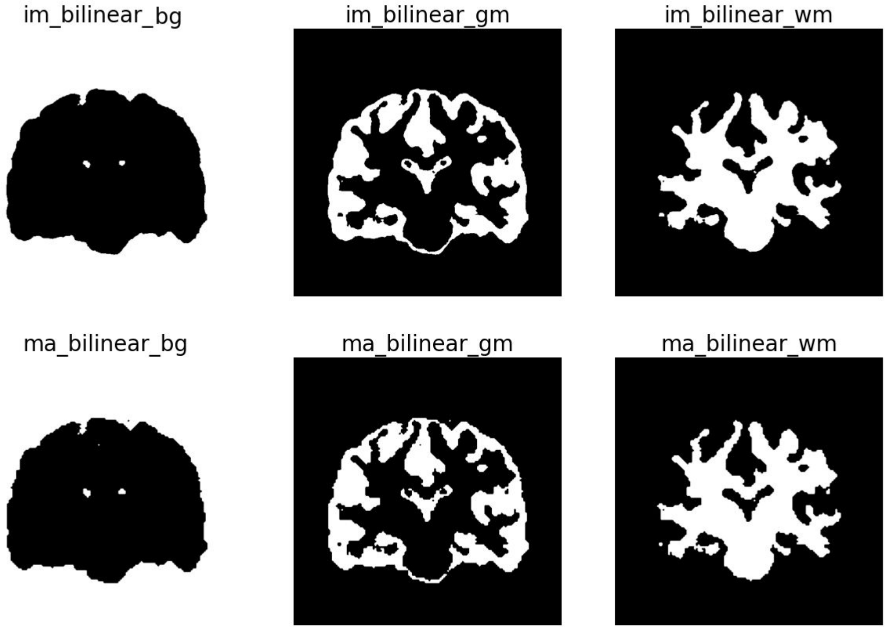
Bilinear Interpolation for Upsampling

- **Bilinear Interpolation:** which calculates pixel values by linearly interpolating between neighboring pixels, producing smoother transition

```
# bilinear interpolation
upsampled_bilinear = zoom(filtered_image, zoom=4, order=1)

bilinear_bg_mask = upsampled_bilinear < background_threshold
bilinear_gm_mask = (upsampled_bilinear >= background_threshold) & (
    upsampled_bilinear < gray_matter_threshold
)
bilinear_wm_mask = upsampled_bilinear >= gray_matter_threshold
```

Bilinear Interpolation



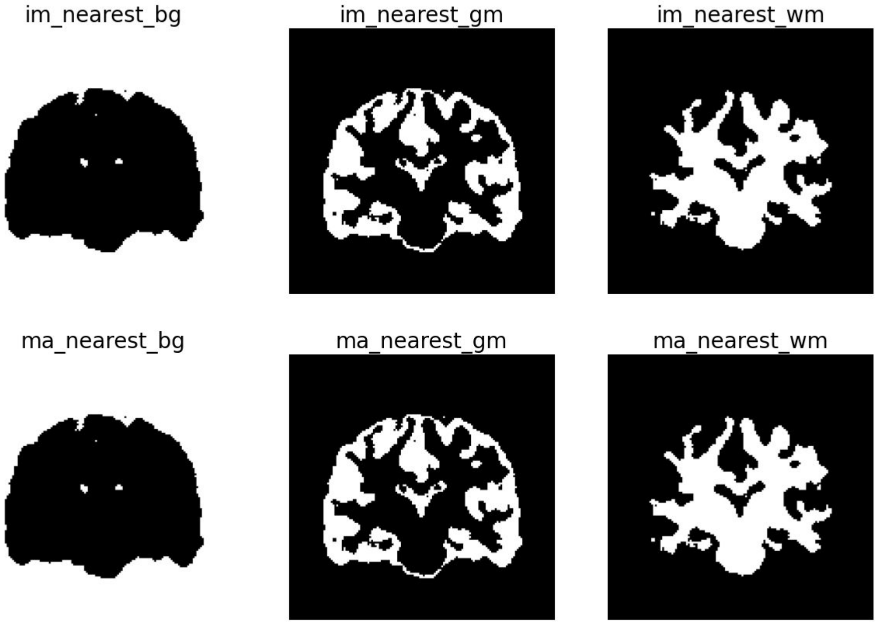
Nearest Neighbor Interpolation for Upsampling

- **Nearest neighbor interpolation:** When resizes an image, it assigns the values of the nearest pixel in the input image to each pixel in the output image, preserving edges but creating a more "staircase" effect.

```
# nearest neighbor interpolation
upsampled_nearest = zoom(filtered_image, zoom=4, order=0)

nearest_bg_mask = upsampled_nearest <= background_threshold
nearest_gm_mask = (
    upsampled_nearest > background_threshold
    ) & (
    upsampled_nearest <= gray_matter_threshold
)
nearest_wm_mask = upsampled_nearest > gray_matter_threshold
```

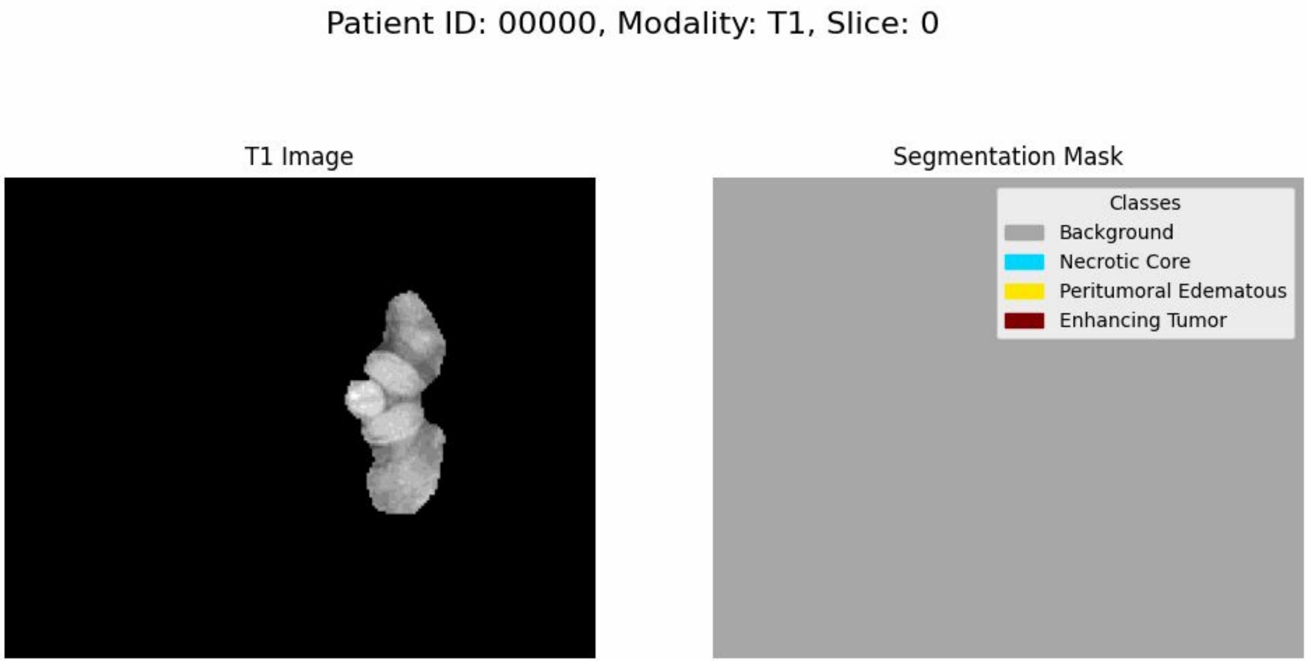
Nearest Neighbor Interpolation



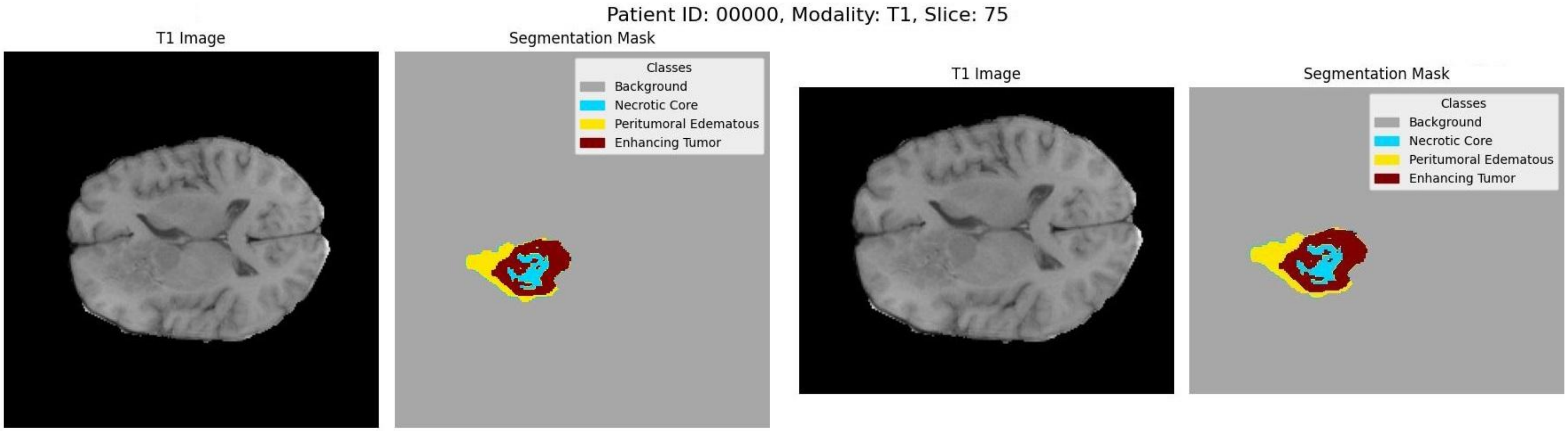
Task IV

Automatic Glioma Segmentation Pipeline with U-Net Model

Data Exploration



Data Cropping



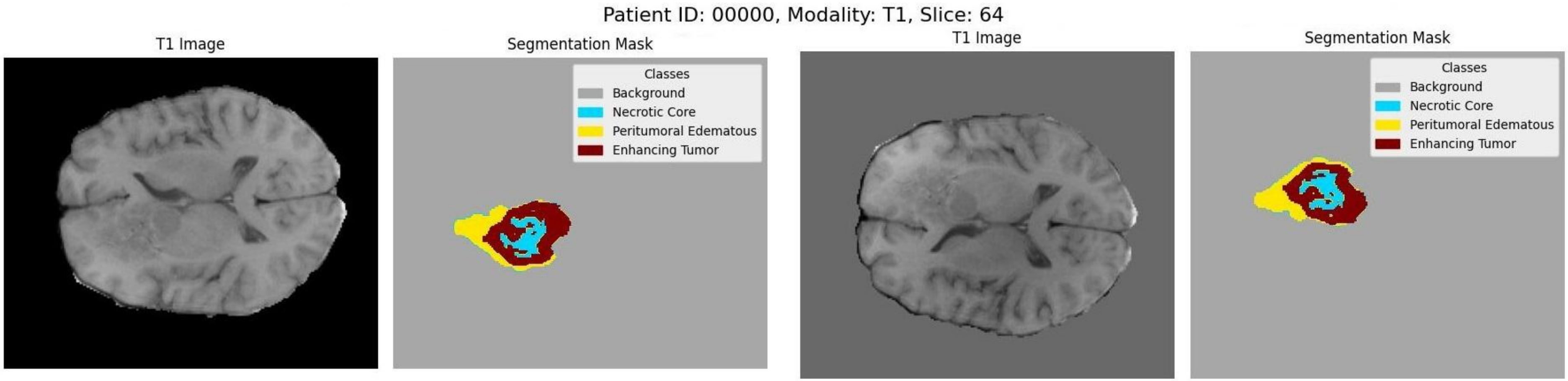
Original

Cropped

Data Augmentation

Original

Augmented

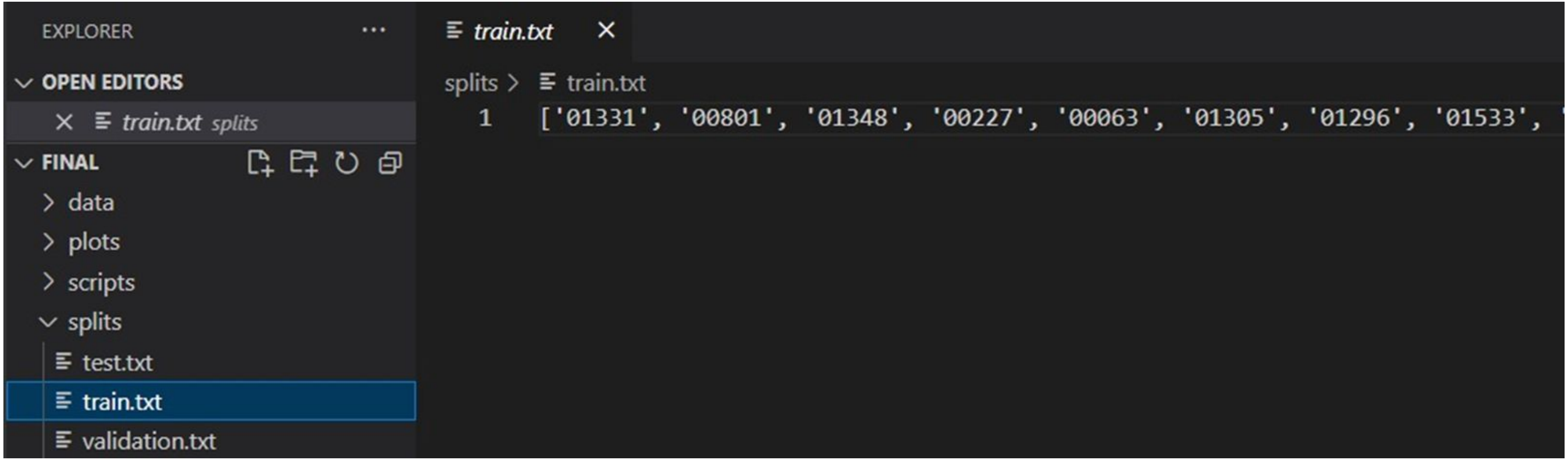


- Random flip
- Random bias field

- Random Gaussian noise
- Normalization to [0,1] scale

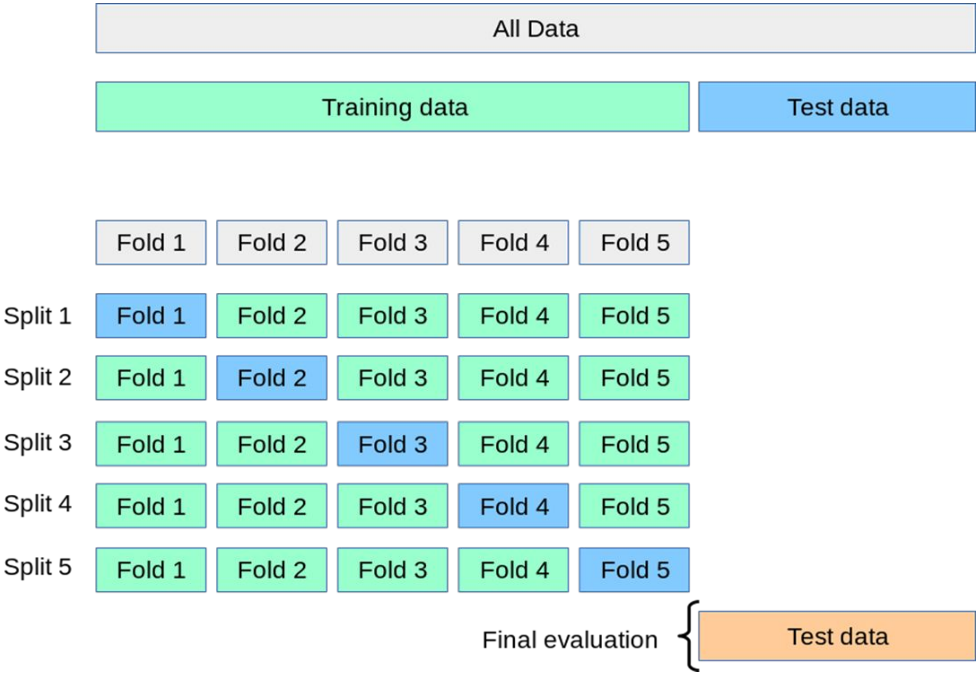
Data Splitting

- 75% of data reserved for training, 15% for validation, and 10% for testing.

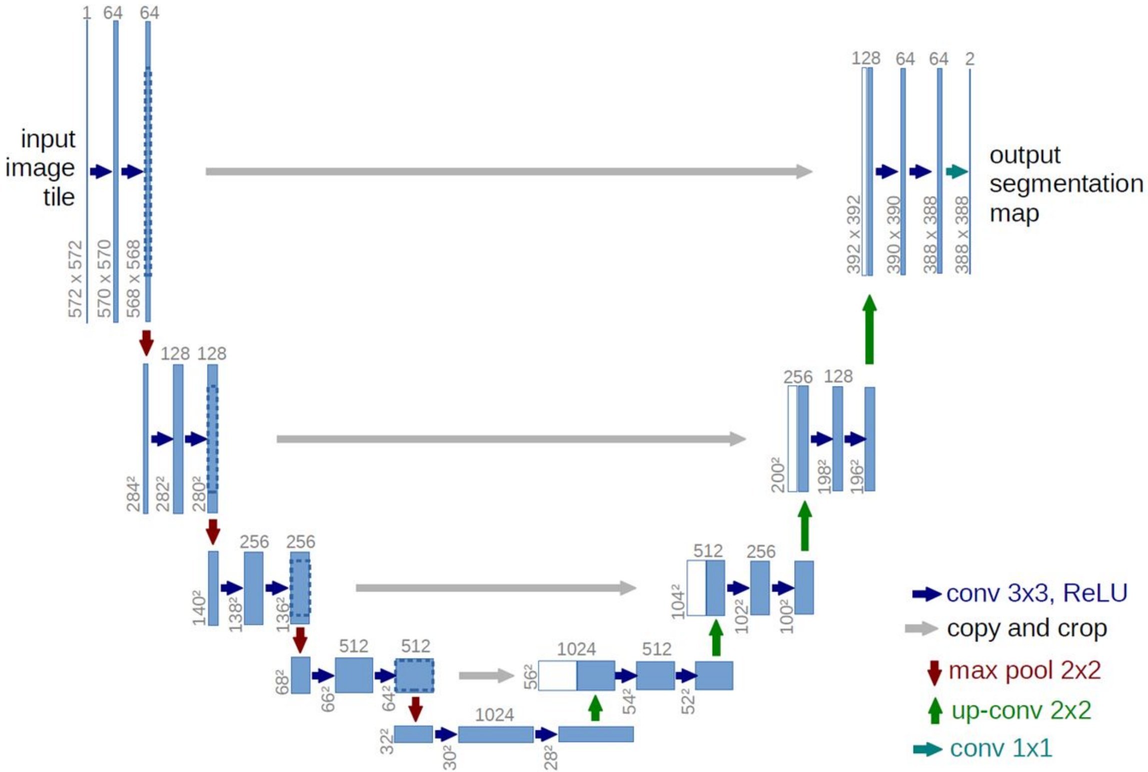


Data Splitting: KFold

- 10% of data reserved for testing (same data as without cross-validation).
- 3 folds.



Neural Network: UNet

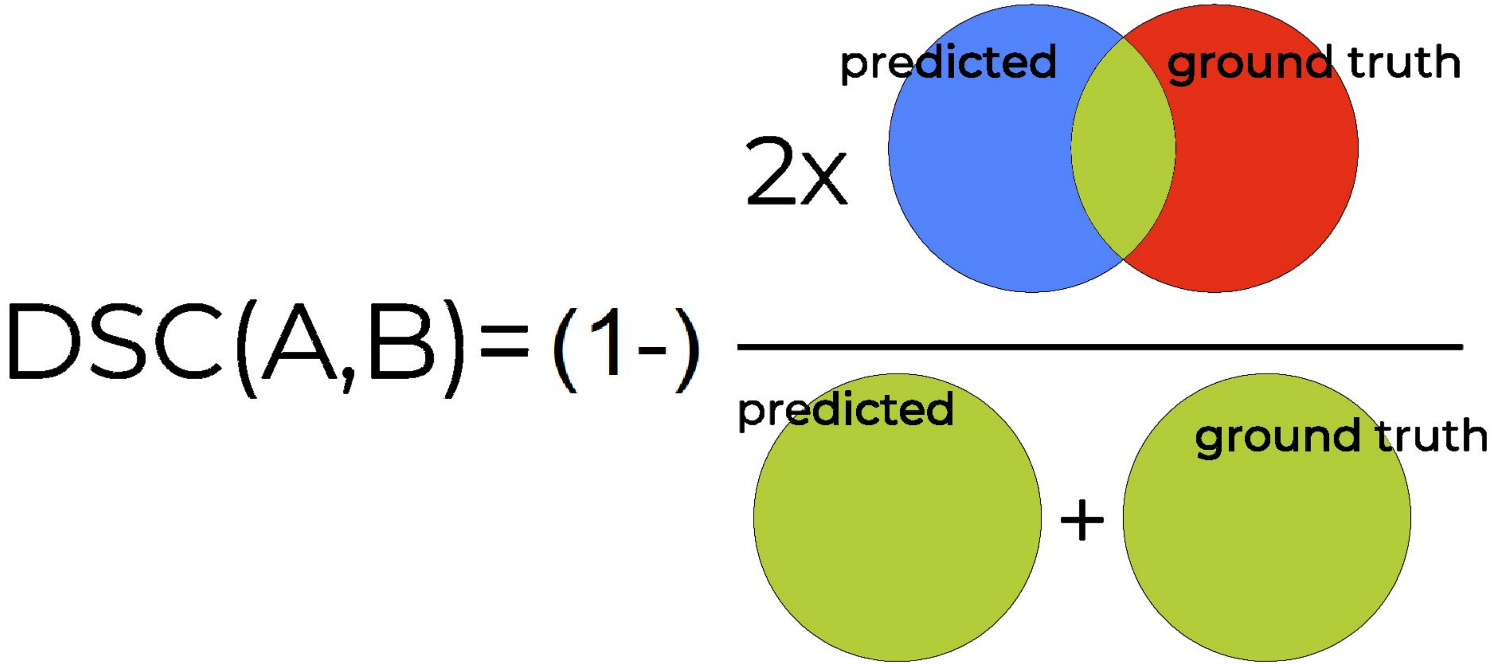


Models we tried:

- **UNet**
- UNETR
- SwinUNETR
- HighResNet

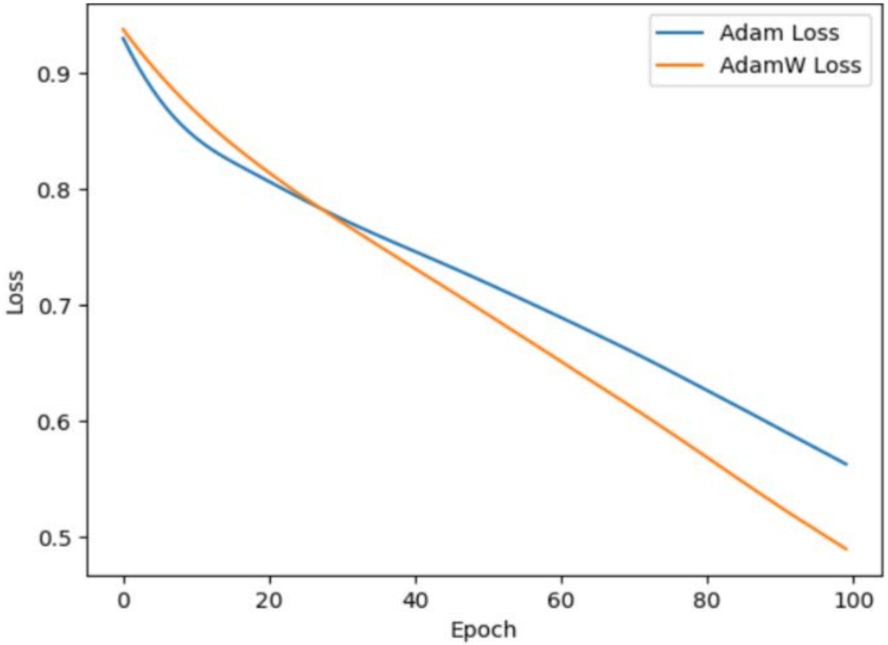
Only UNet worked: other architectures required too much memory.

Loss & Metric

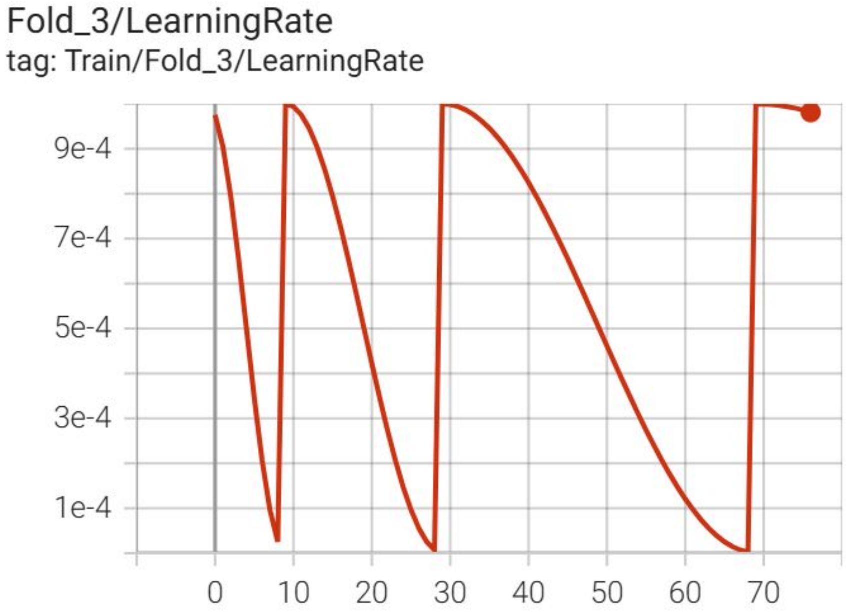


Optimizer & Learning Rate Scheduler

AdamW: Adam with Decoupled Weight Decay

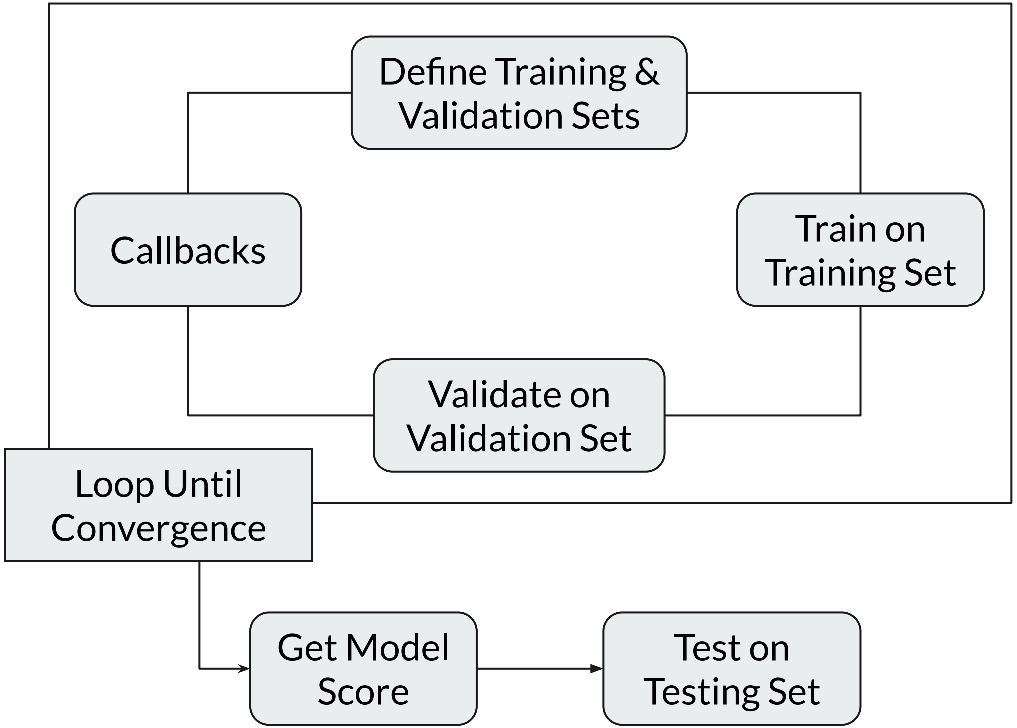


Cosine Annealing with Warm Restarts

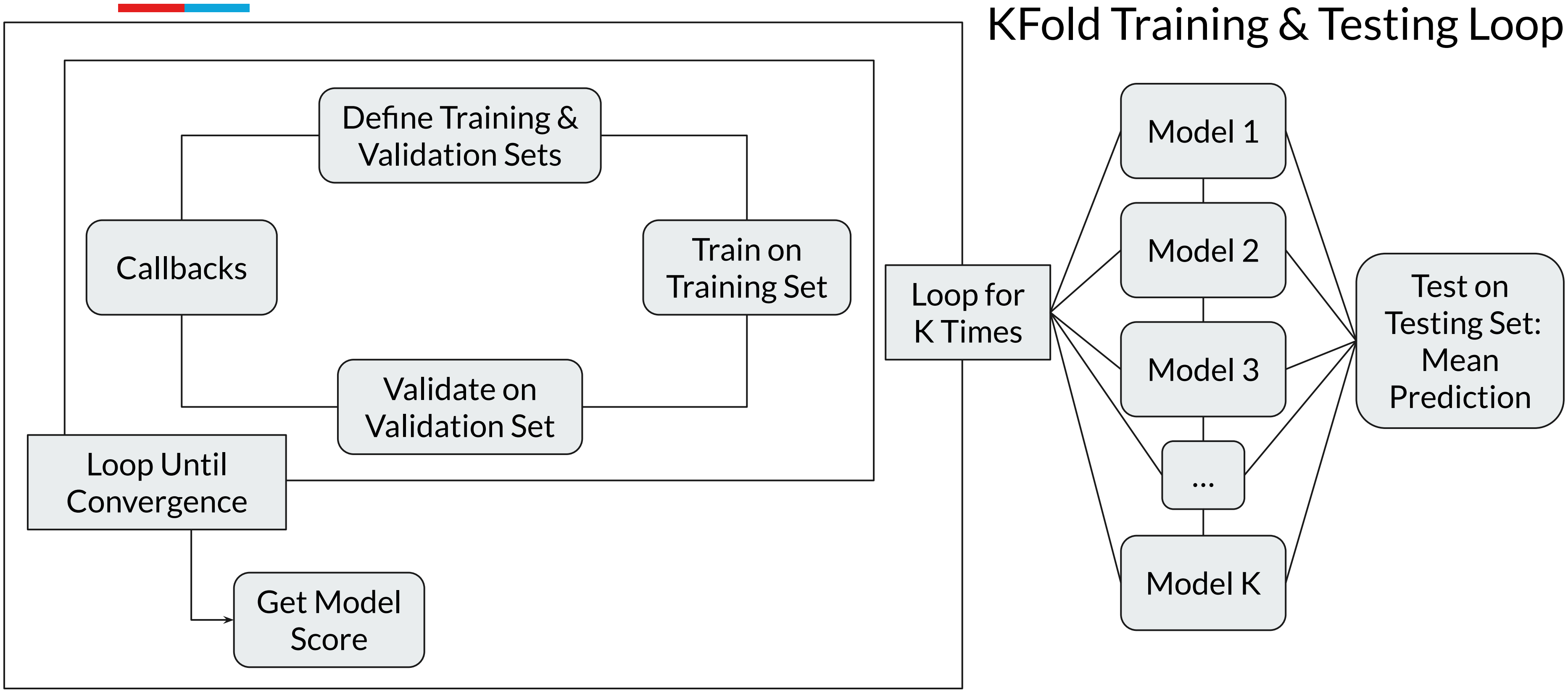


Training & Testing: Default Strategy

Default Training & Testing Loop



Training & Testing: KFold Cross-Validation

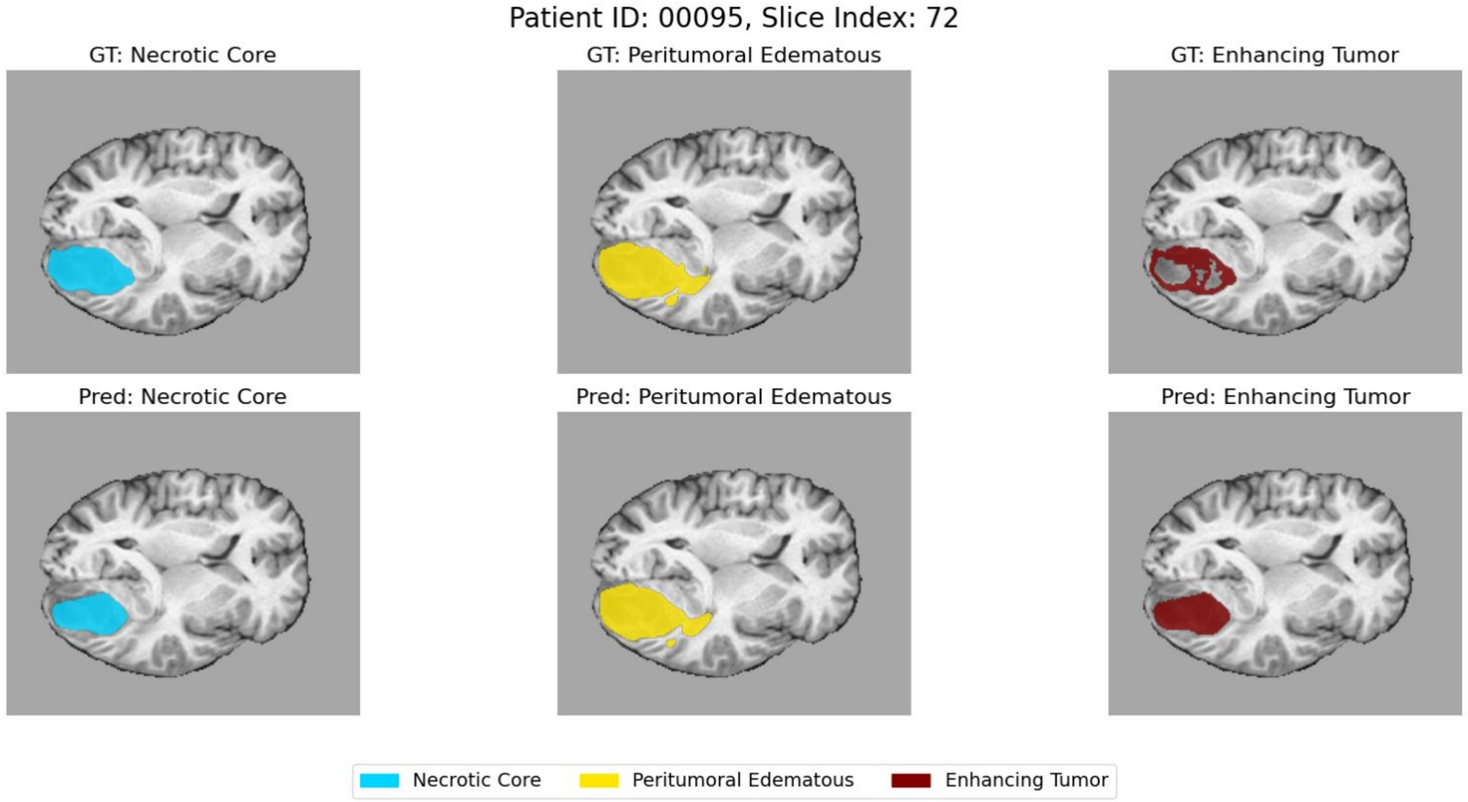


Results

Method	Default Training Procedure, T1 Modality	Default Training Procedure, Multiple Modalities	**KFold Training Procedure, Multiple Modalities**
Dice Score	0.6935	0.7576	**0.7616**

- Introducing more modalities improves model performance.
- Using KFold training and ensemble prediction technique increase prediction accuracy, although not substantially.

Results


Patient ID: 00095, Slice Index: 72
GT: Necrotic Core
GT: Peritumoral Edematous
GT: Enhancing Tumor
Pred: Necrotic Core
Pred: Peritumoral Edematous
Pred: Enhancing Tumor
Necrotic Core
Peritumoral Edematous
Enhancing Tumor