

InC Presentation

# Deep Learning Approach for Detection of Alzheimer's Disease

- Aditi Bankar, Shreya Bansod and Aayush Mohod
- Guided by Dr. M. P. Turuk

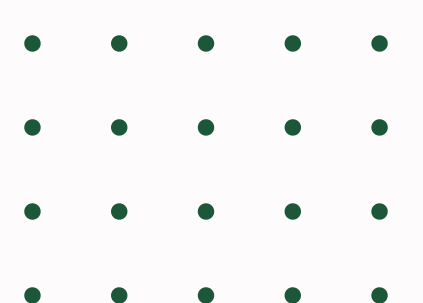




# Introduction

Alzheimer's Disease (AD) poses a significant global health challenge, with an increasing prevalence among ageing populations, affecting over 55 million people globally and an annual rise of 10 million new cases. Early detection and accurate diagnosis are crucial for effective intervention and treatment planning. In recent years, advanced imaging technologies, particularly Magnetic Resonance Imaging (MRI), have emerged as promising tools for assisting in the diagnosis of neurocognitive disorders such as AD.

This paper focuses on a comprehensive analysis of datasets, sophisticated Deep Learning and Transformer-based approaches, and evaluation metrics used to automate the detection and classification of AD based on MRI scans.



# Objectives

**Effective  
Processing of Raw  
MRI Images**

**1**

**Segmentation of  
the Hippocampus  
and other ROI**

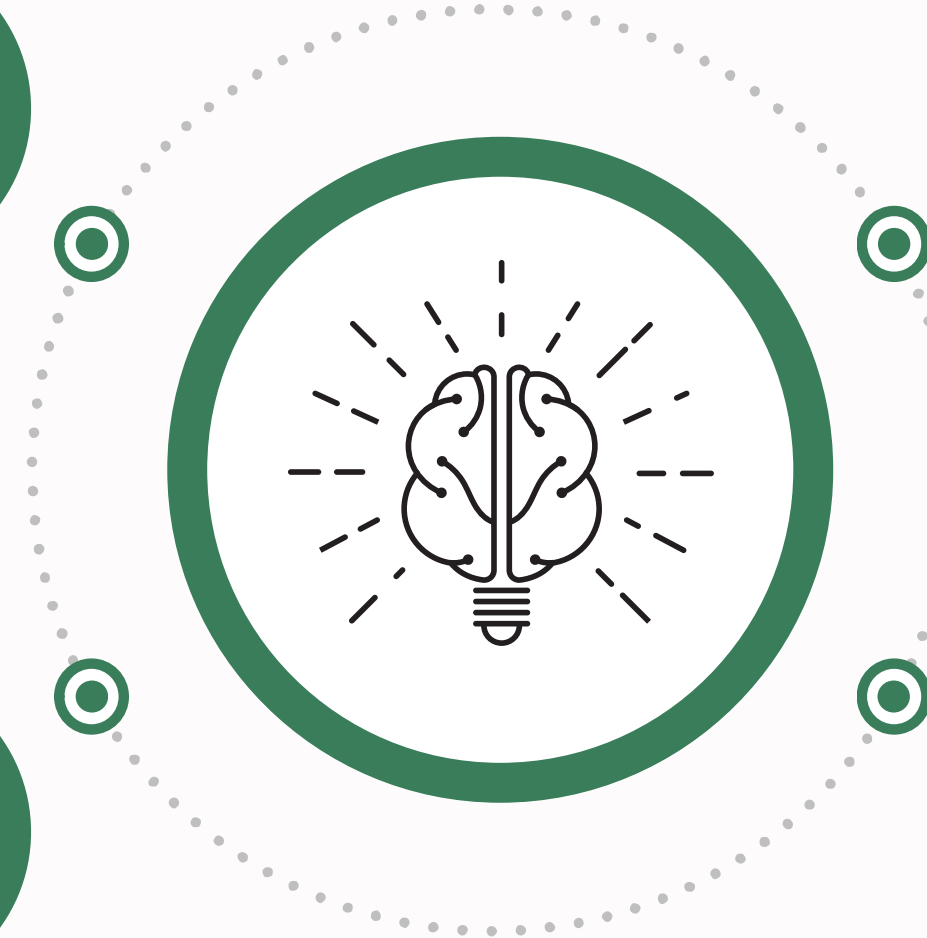
**2**

**Building a  
scalable pipeline  
for detection of  
Alzheimer's**

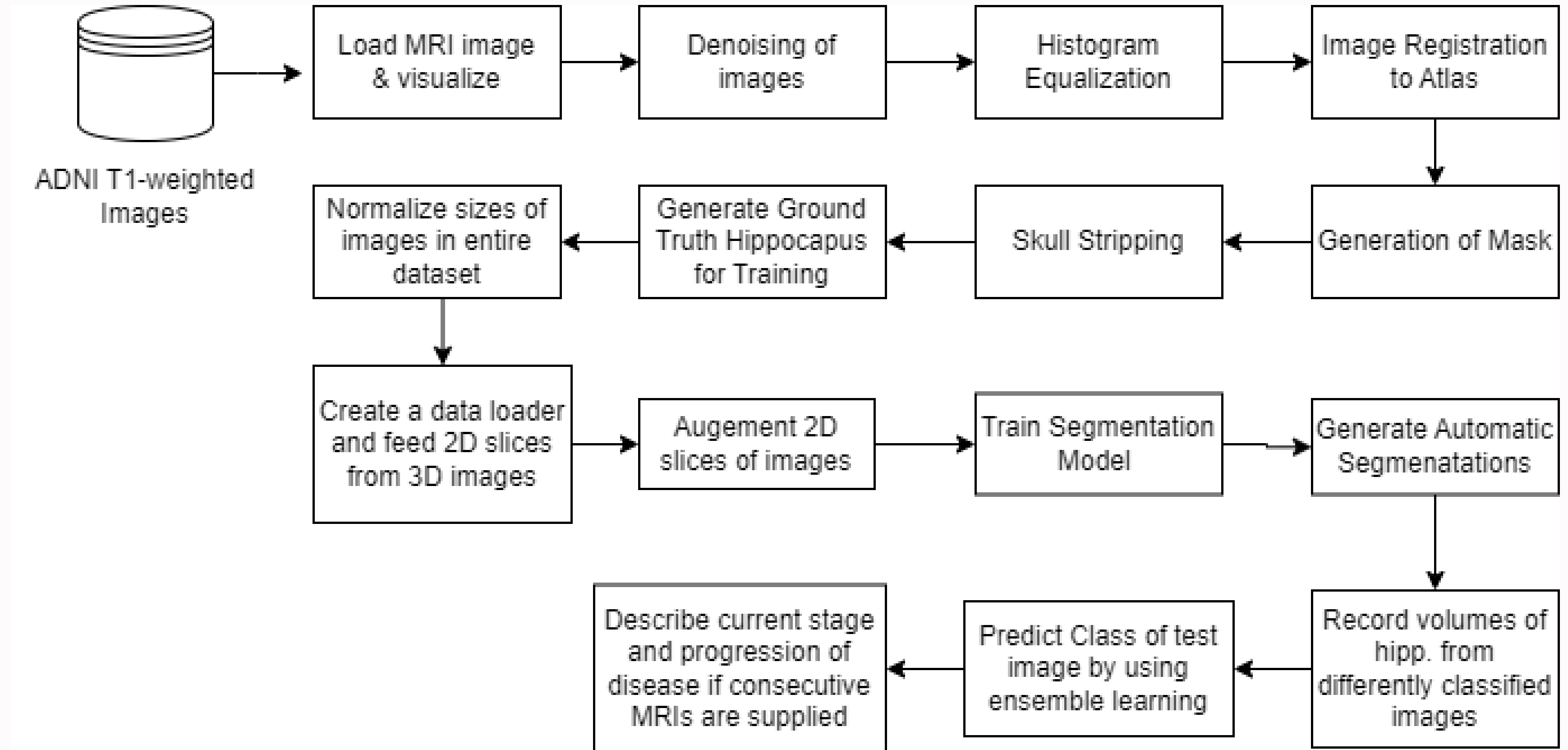
**4**

**Classification of  
MRIs on the basis  
of neurocognitive  
conditions**

**3**



# Pipeline Design



# Tech Stack

Since the project is based on using deep learning techniques for effective and fast detection of neurocognitive diseases, the tech stack consists primarily of niche Python libraries

01

Nibabel, SimpleITK

MONAI, Pytorch

02

03

Tensorflow

Numpy, Scikit

04



# Methodology

## MRI Image Processing

- Image Visualization
- Registration to Atlas
- Derivation of masks
- Brain Extraction
- Denoising
- Reorientation

## Segmentation using UNet

- Creating dataset
- Deriving ground truth
- Standardizing Images
- Training of UNet
- Testing of UNet
- Plotting Accuracy

## Classification of MRIs

- Creating Datasets
- Volumetric Analysis
- Binary Classification using 3D CNN
- Classification using Visual Transformers
- Comparison of accuracy

References: <https://docs.google.com/spreadsheets/d/1AaiXoQD2875jcKHiyxGEXg-NFvViLX3M1jOehnYwIhM/edit?gid=1112319157#gid=1112319157>

# MRI Image Processing

## Data Selection Process

**Dataset Source:** Utilized ADNI (Alzheimer's Disease Neuroimaging Initiative) dataset for image acquisition.

### Image Corrections Applied:-

- **Gradwarp Correction** - Improves geometric accuracy of MRI images, ensuring that structures are accurately represented in space.
- **B1 Correction** - Provides uniform signal intensity across the entire image, improving contrast and quantitative accuracy.
- **N3 Correction** - Normalizes image intensities to reduce bias and improve the accuracy of image interpretation, particularly useful in brain imaging and other anatomical studies.

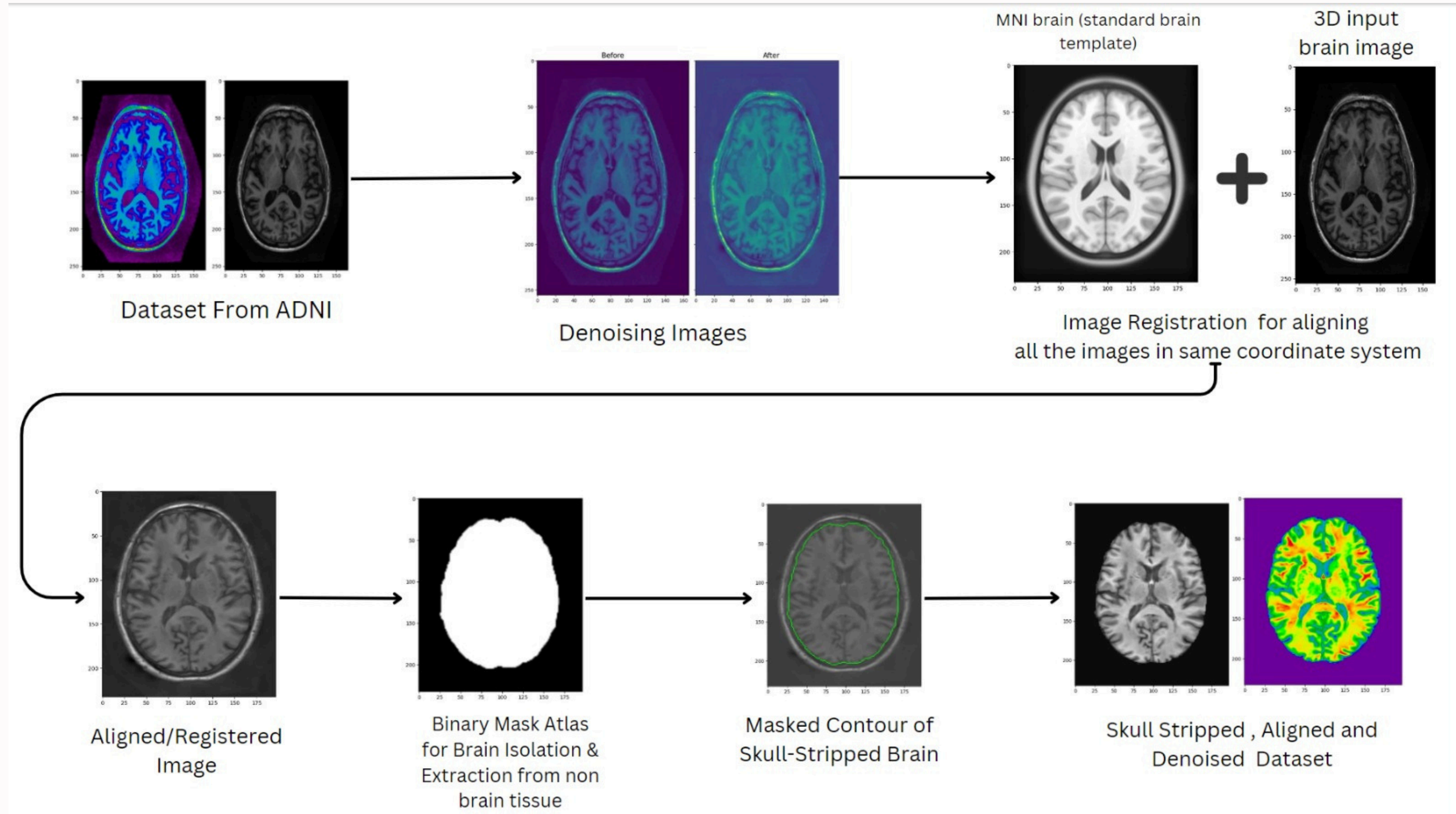
# MRI Image Processing

## Why we are using T1-Weighted Images?

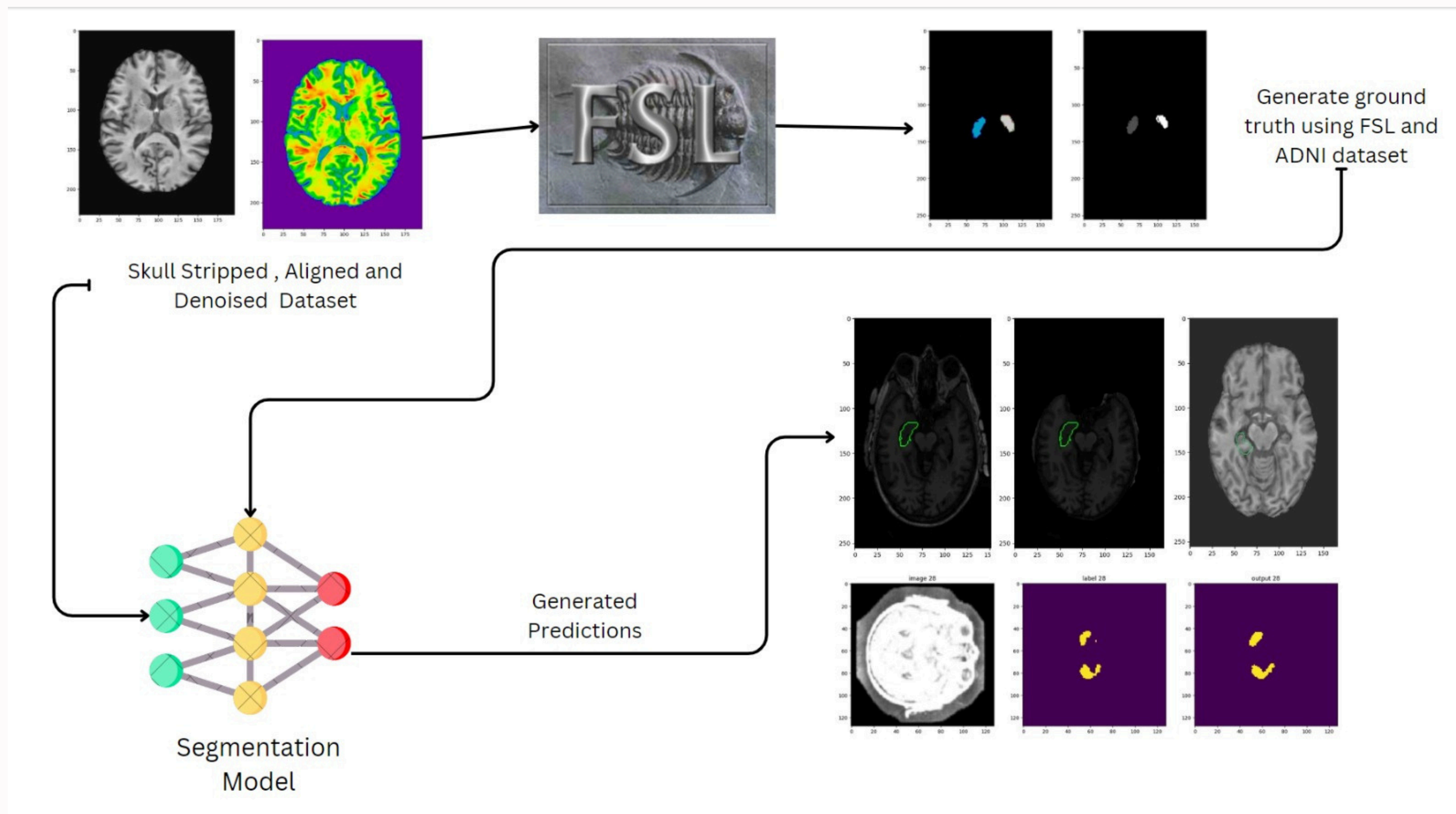
- **Brightness of Fat and White Matter:** In T1-weighted images, tissues with shorter T1 relaxation times, like fat and white matter, appear bright. This makes T1-weighted images useful for visualizing anatomical structures.
- **Dark Appearance of Fluid:** Fluids, such as cerebrospinal fluid (CSF), which have longer T1 relaxation times, appear dark. This contrast makes T1-weighted images good for visualizing the boundaries between soft tissues and fluid spaces.
- **Good Anatomical Detail:** T1-weighted images provide excellent contrast between different soft tissues, making them useful for structural and anatomical imaging, particularly of the brain and spinal cord.



# Preprocessing Pipeline



# Project Pipeline



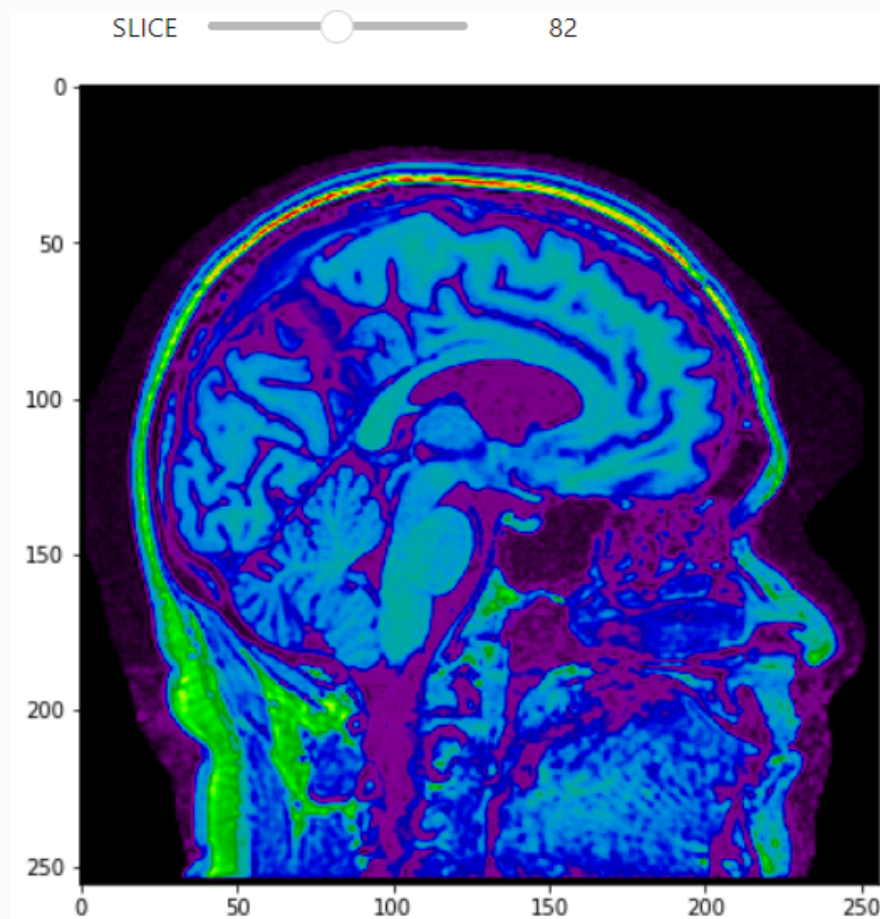


# MRI Image Processing

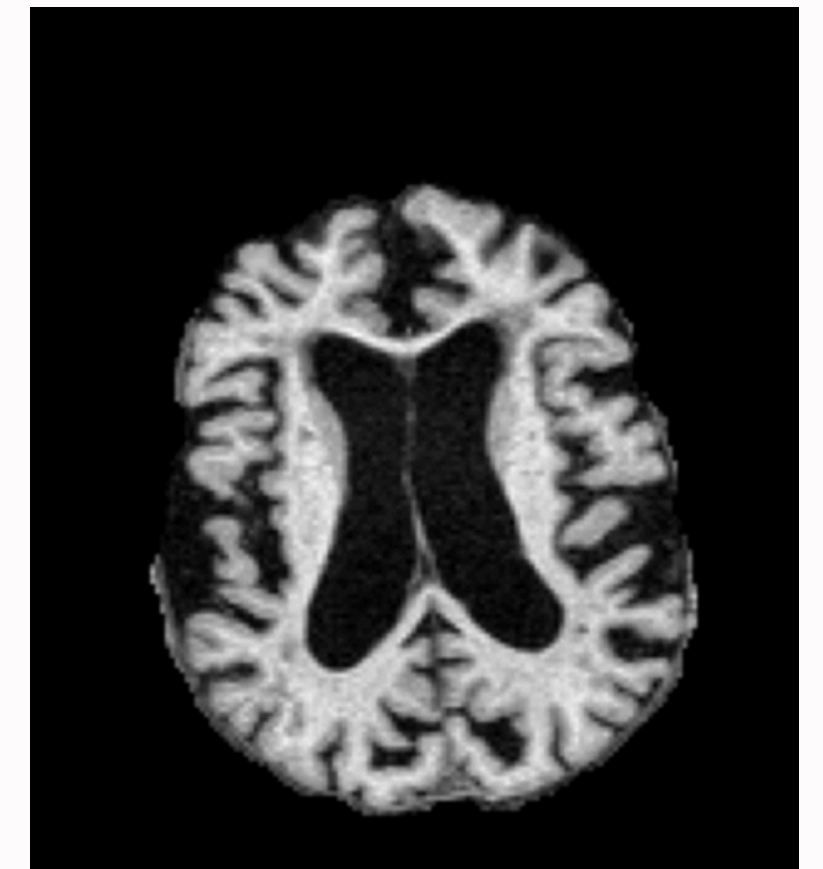
## Steps of Pre-processing

- Visualize Image using AntspyNet
- Denoising the Images.
- Choose Appropriate Mask for registration , in this case MNI 152 Atlas mask.
- Register the image to the mask using WarpAffine for spatial alignment.
- Outline a mask from the registered image for extraction of the brain.
- Generate Skull Stripped Image.
- Normalize image to (256,256,160)

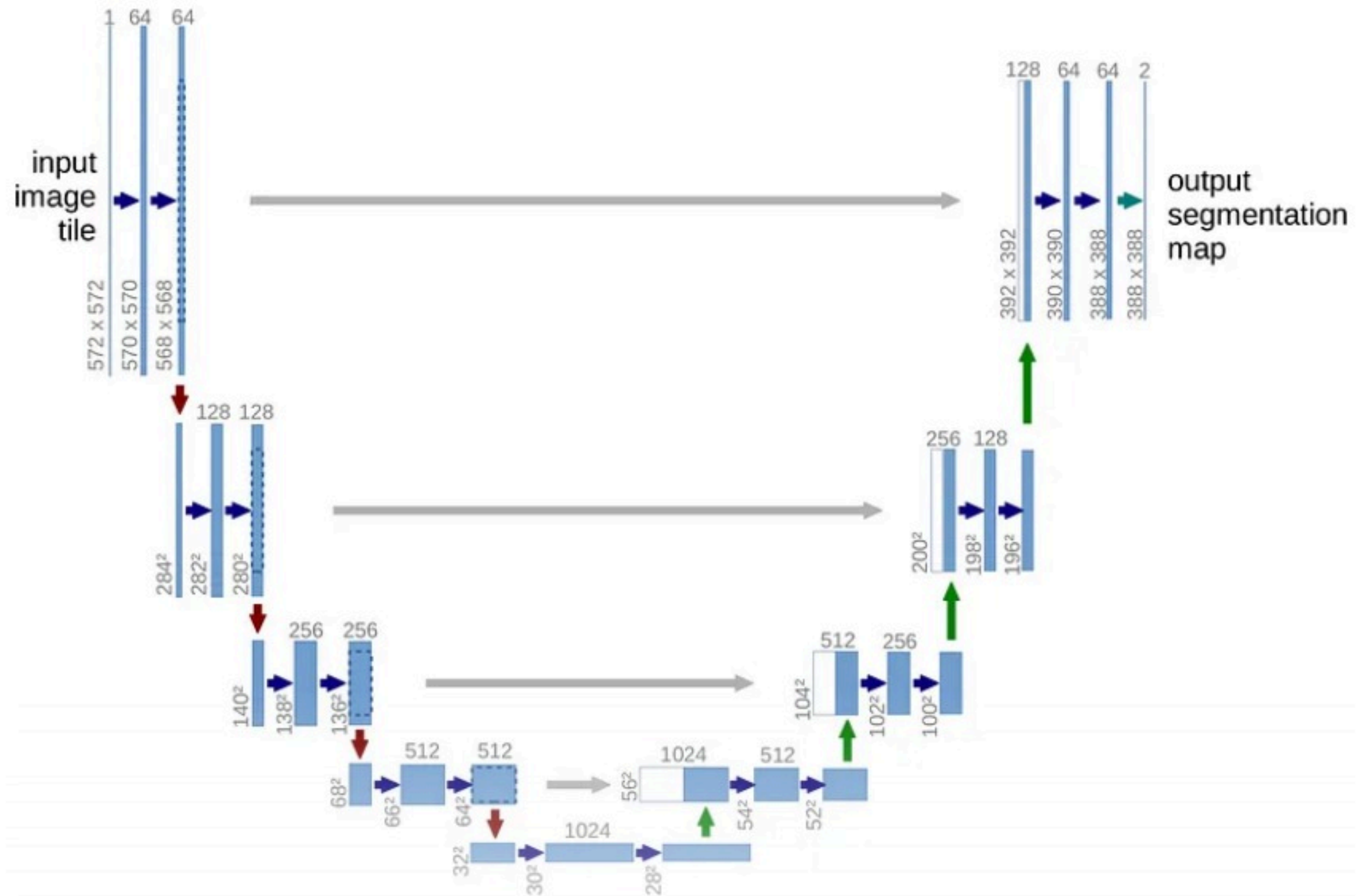
Raw Image

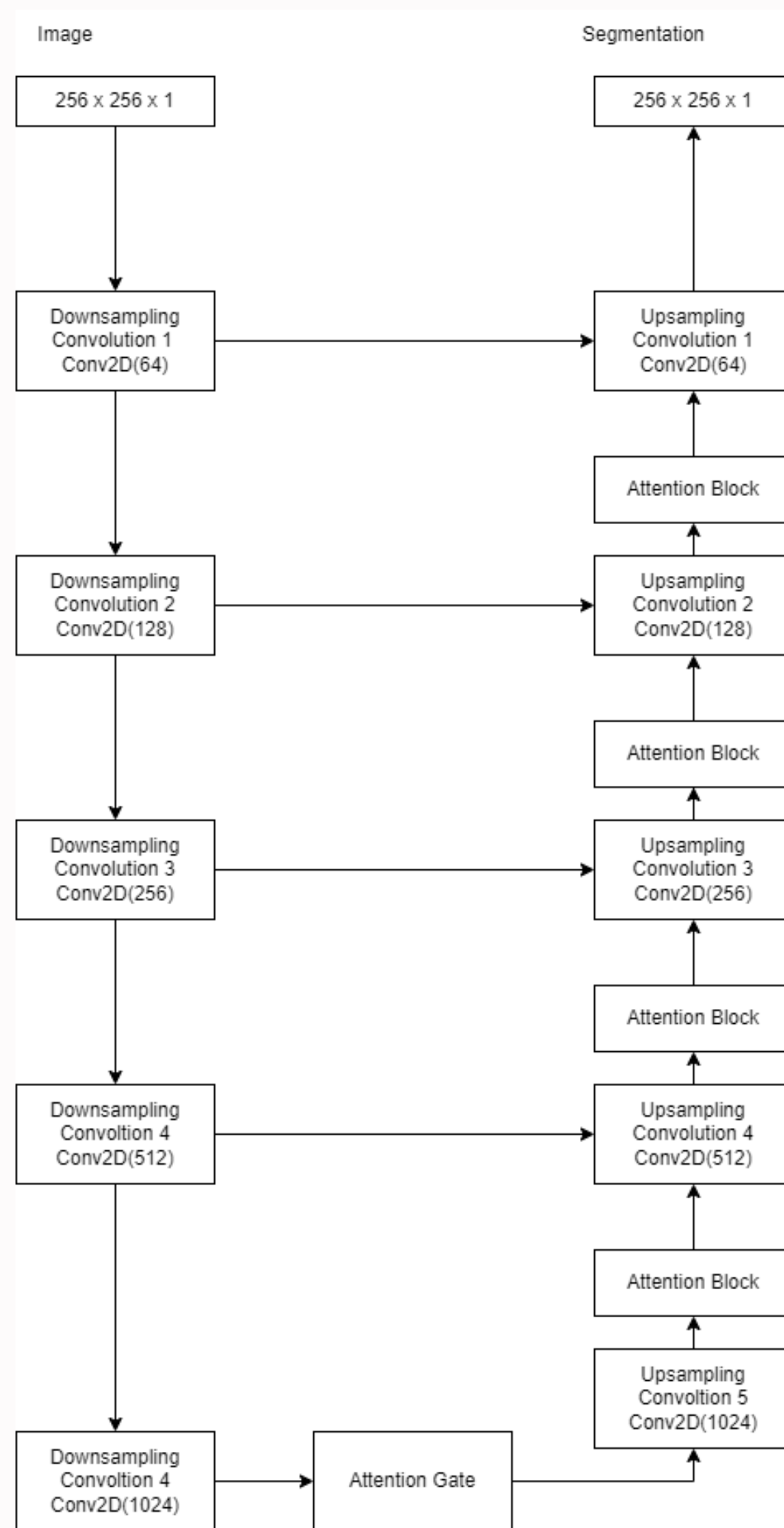


Skull Stripped  
Normalized Image

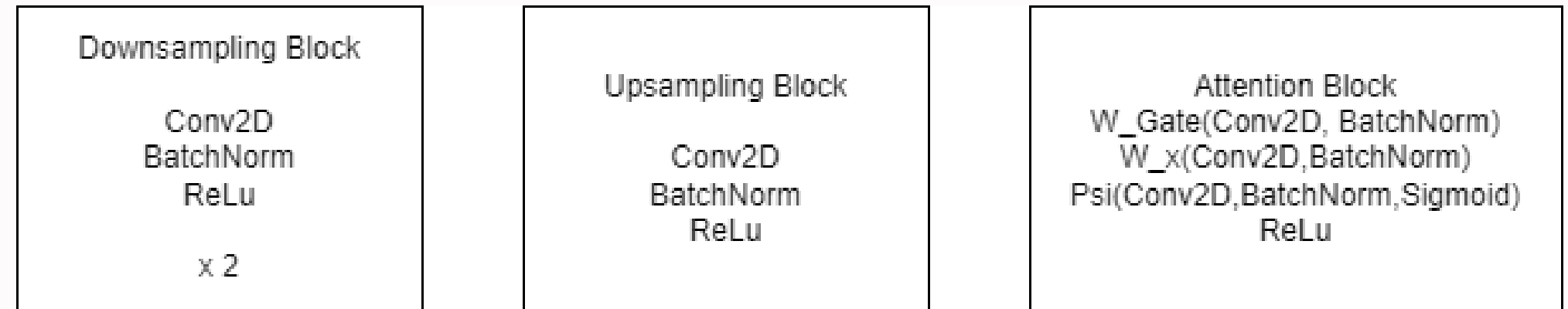


# U-Net Architecture





# Attention U-Net Architecture



# Algorithm for UNet Architecture

## UNet Algorithm

- i. Input images of size  $(H, W, C)$
- ii. Apply convolution followed by ReLU activation for down sampling

$$Y_{down} = \text{ReLU}(\text{Conv}(X, W) + b)$$

- iii. Compress information at the bottleneck using convolution layers.

$$Y_{bottleneck} = \text{ReLU}(\text{Conv}(Y_{down}, W) + b)$$

- iv. Upsample and concatenate feature maps from the contracting path.
- v. Apply a final convolution to generate the segmentation map.  
 $\hat{Y} = \text{Sigmoid}(Y_{final})$

## UNet with Attention Algorithm

- i. Input image of size  $(H, W, C)$
- ii. Apply convolution and ReLU to downsample the image.

$$Y_{down} = \text{ReLU}(\text{Conv}(X, W) + b)$$

- iii. Compute attention coefficients to focus on important features.

$$a = \sigma(W_a \cdot [Y_{down}, Y_{up}])$$

- iv. Multiply attention coefficients with the upsampled feature map.  $Y_{att} = a \cdot Y_{up}$

- v. Concatenate the attention-modulated output with the feature maps from the contracting path.  $Y_{final} = \text{Conv}(\text{Concat}(Y_{att}, Y_{down}), W) + b$

- vi. Generate the final segmentation map using a sigmoid function.



# MRI Image Processing

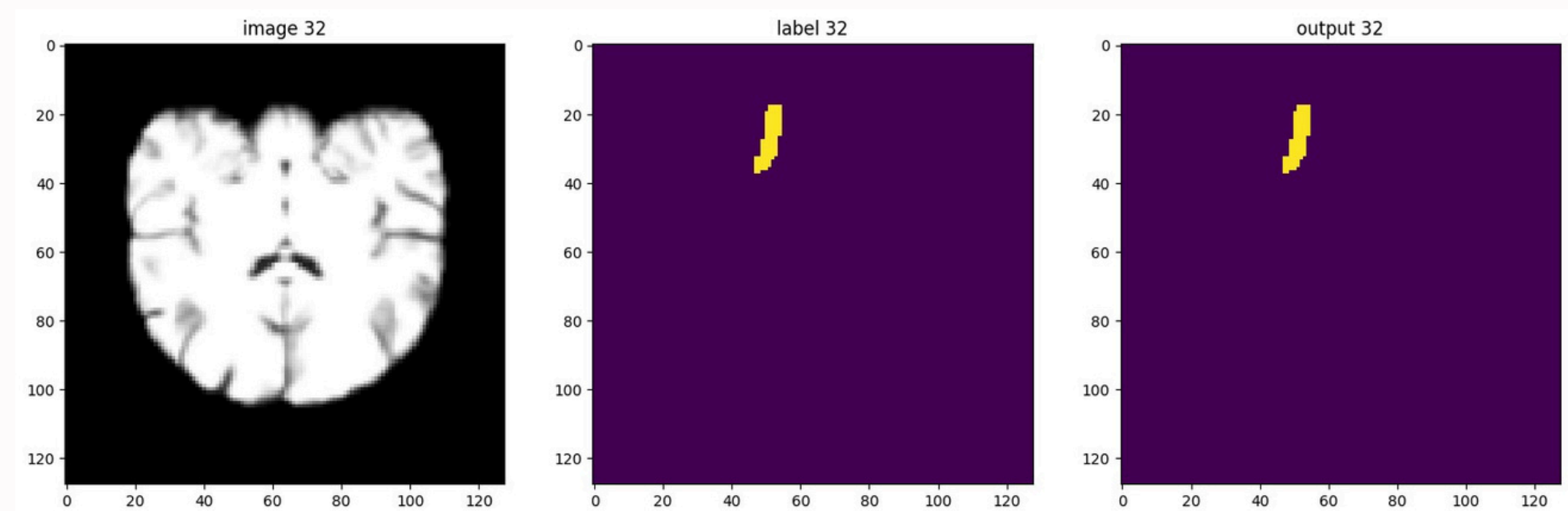
## Segmentation

- Generate Ground Truths using FSL
- Standardize the images and segmentations
- Load training and testing images into the Data Loader
- Define unet using Adam optimizer and 75 epochs
- Check and plot Accuracy with testing images
- Scale the model for larger dataset

## UNet specification

- Model: 3D Unet
- Spatial Dimensions :3
- Optimizer:Adam with specific settings like learning rate and weight decay.
- Activation function:Sigmoid

Final Segmented results in comparison with the ground truth



# Algorithm for SWINUNet Architecture

## Patch Embedding Layer

- Purpose: The PatchEmbedding layer converts the input image into patches, a necessary step for transformer models that operate on sequences rather than spatial data.
- Mechanism:
  - The Conv2d layer divides the input into patches with a kernel size and stride of (4,4), which reduces the spatial resolution by a factor of 4, effectively creating 16x smaller patches.
  - This layer outputs patches that serve as tokens for the Swin Transformer.

# Algorithm for SWINUNet Architecture

## Encoder

The encoder captures multi-scale features by passing the patches through a series of Swin Transformer blocks and Patch Merging layers.

- Swin Blocks: Each SwinBlock has two SwinTransformerBlock layers.
  - SwinTransformerBlock is a key Swin Transformer component, built with WindowAttention and a multi-layer perceptron (MLP).
  - WindowAttention:
    - Divides the input into small windows (local regions).
    - Each window gets processed separately with self-attention, making this layer computationally efficient.
    - The qkv (query, key, value) projection layers project the input features for attention, while proj maps the attention output back to the original feature dimension.
    - The Softmax function calculates the attention scores, highlighting important features within each window.
  - LayerNorm normalizes input features to stabilize training, and the MLP further processes these normalized features.
    - The MLP contains a GELU activation for non-linearity and a second LayerNorm.
  - Patch Merging: Between Swin blocks, the PatchMerging layers reduce spatial resolution by combining neighboring patches and doubling the feature dimension. This downsampling creates a hierarchical representation, akin to UNet's downsampling.

# Algorithm for SWINUNet Architecture

## Bottleneck

The bottleneck is a crucial part of the model's feature extraction.

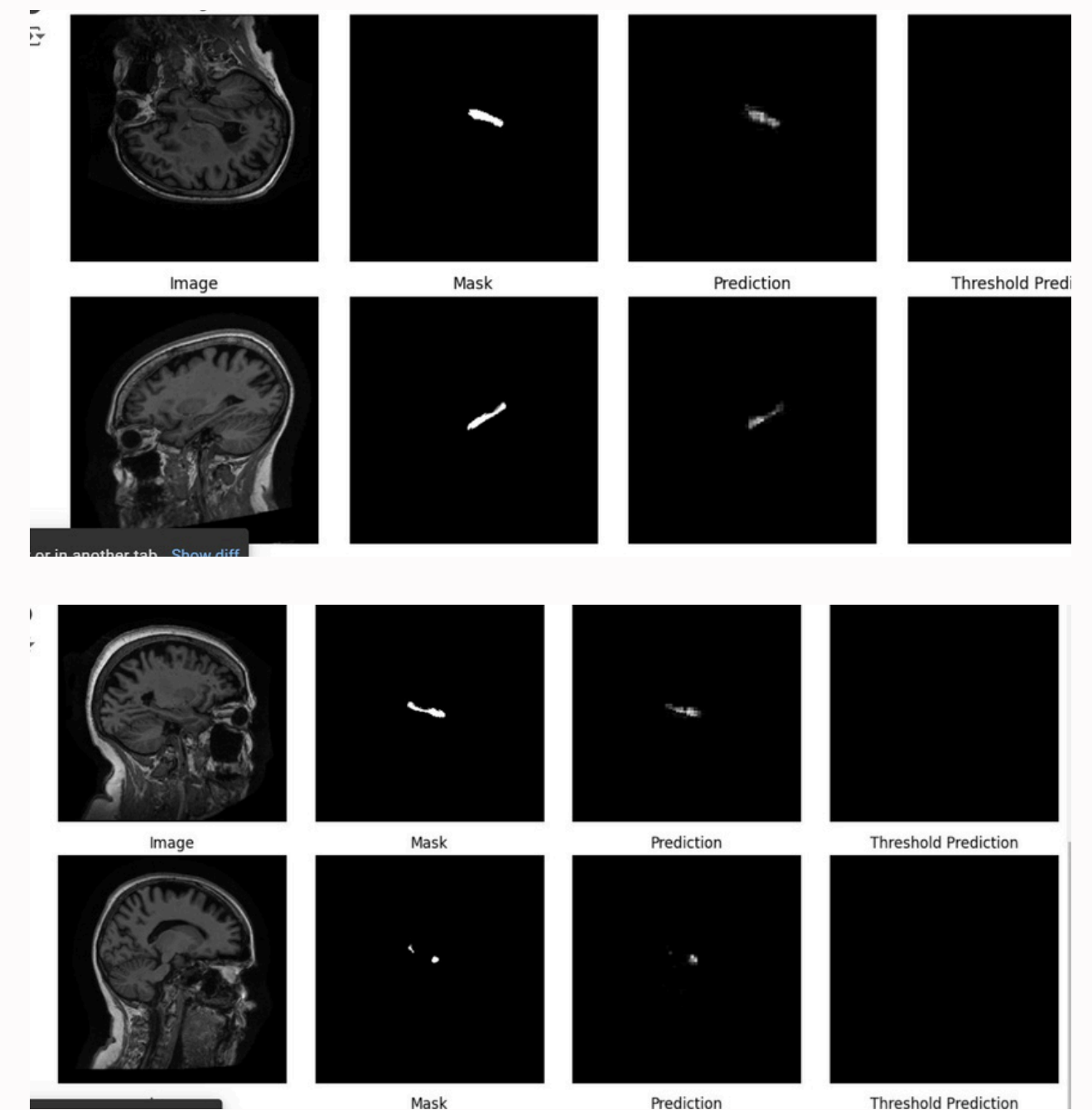
- The bottleneck consists of a single SwinBlock that operates at the lowest resolution (largest receptive field) with the highest number of channels (feature richness).
- This block captures the deepest features, akin to the "base" in standard UNet architectures.
- The SwinTransformerBlock here has the same components: WindowAttention, MLP, and LayerNorm, but with increased feature dimensions (512 channels) and a larger MLP capacity (2048 output channels), allowing it to encode a high-level understanding of the input.

# Algorithm for SWINUNet Architecture

## Decoder

The decoder mirrors the encoder structure but in reverse, progressively upsampling the feature maps back to the original resolution.

- Decoder Swin Blocks: These SwinBlocks process the upsampled features at each stage, utilizing similar SwinTransformerBlock layers to refine features.
  - As with the encoder, each block contains two SwinTransformerBlocks, each with WindowAttention, MLP, and LayerNorm.
  - The attention mechanism in each block focuses on different regions of the feature map at each level, ensuring the decoder retains relevant spatial information while upsampling.



# Evaluation Metrics

- F Measure : F1 Score (Dice similarity coefficient)
- Scores the overlap between predicted segmentation and ground truth.
- For 75 epochs dice coefficient for:
  1. UNET is 0.96
  2. Attention UNET is 0.70
  3. SWIN Transformer is 0.73

Metric	Definition	Formula
Sensitivity (Recall)	The proportion of actual positives correctly	$\frac{TP}{TP + FN}$
Accuracy	The overall correctness of the model in classifying both positives and negatives.	$\frac{TP + TN}{TP + TN + FP + FN}$
<u>IoU</u> (Jaccard Index)	Measures the overlap between predicted segmentation and the ground truth.	$\frac{TP}{TP + FP + FN}$
Dice Coefficient	A similarity metric between the predicted segmentation and the ground truth, commonly used in segmentation.	$\frac{2 \times TP}{2 \times TP + FP + FN}$



# Our Custom Model's Architecture

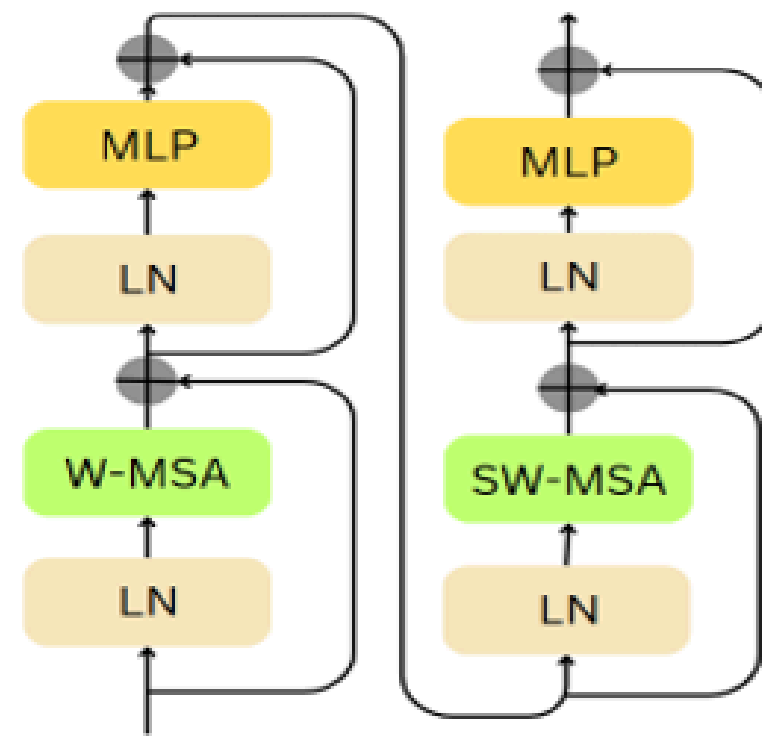
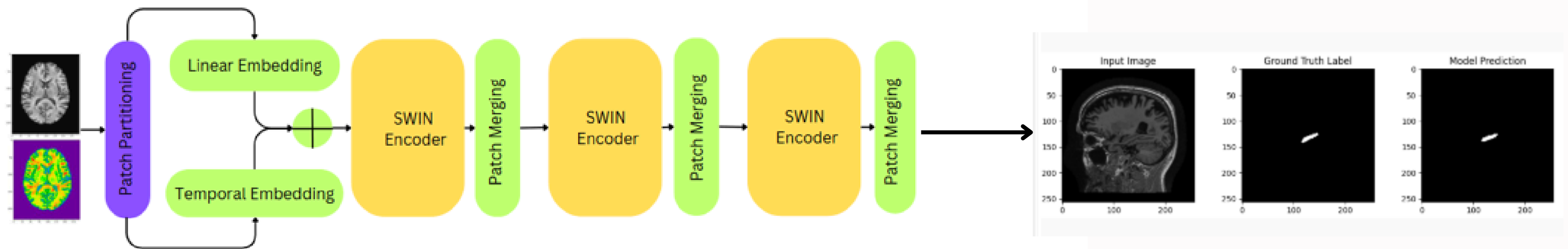


Fig 2(A). Two consecutive SWIN encoders



# What does our model brings to the table?

**Effective Use of Specific Segmentation Image Slices** – Optimized selection of slices for training, focusing on critical regions for segmentation.

**Reduction of False Negatives Using Temporal Embeddings** – Improved feature retention across consecutive slices, reducing misclassifications.

**Better Context Awareness** – Temporal embeddings provide a sequential context, improving segmentation consistency across slices.

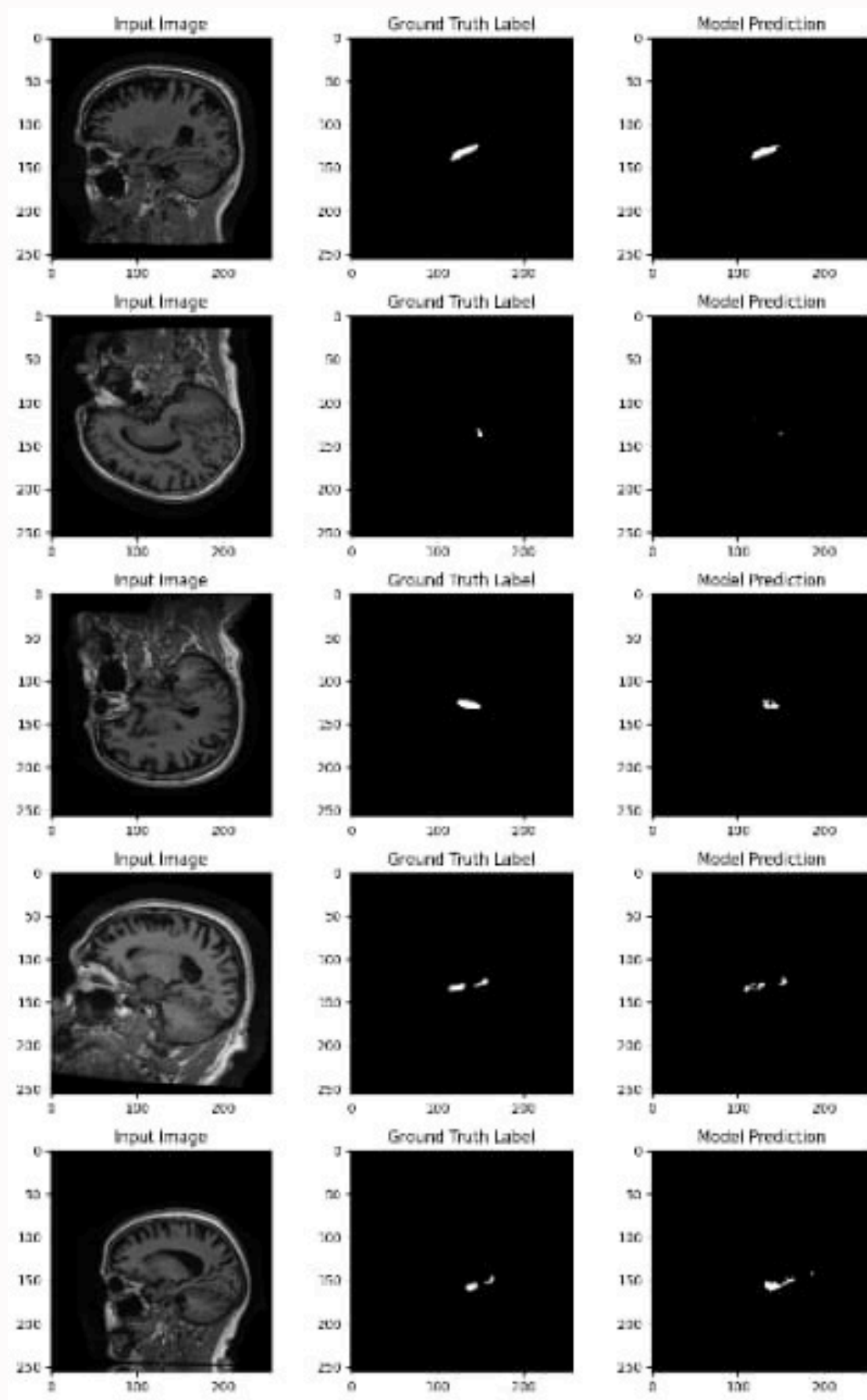
**Enhanced Edge Preservation** – Helps in maintaining structural continuity across adjacent slices, leading to better anatomical delineation.

**Adaptive Learning from Sequence Patterns** – Enables the model to capture temporal dependencies, reducing noise-induced misclassifications.

**Robustness to Motion Artifacts** – Reduces the impact of minor shifts in image acquisition by leveraging temporal coherence.

# Our Metrics

Metric	Value	Explanation
Dice Coefficient	0.78	Measures overlap between predicted and actual segmentations. Higher values indicate better segmentation accuracy.
IoU (Intersection over Union)	0.75	Ratio of intersection to the union of predicted and actual regions. A higher value means better segmentation alignment.
Precision	0.79	Measures how many of the predicted positives are actually correct (i.e., low false positives).
Recall	0.74	Measures how many of the actual positives were correctly identified (i.e., low false negatives).
F1-Score	0.77	Harmonic mean of precision and recall, balancing both metrics for overall performance.



## Sample Outputs

