# Semantic Segmentation using BRATS 2020 Dataset:

Sachin Maurya Roll No.: CS24MS002
Abhishek Omprakash Sharma Roll No.: CS24MS003

***Abstract*—Brain tumor segmentation in magnetic resonance imaging (MRI) plays a crucial role in diagnosis, prognosis, and treatment planning. This study presents an automated deep learning approach utilizing a 2D U-Net architecture for segmenting brain tumor sub-regions in the BraTS2020 dataset. The dataset comprises multi-modal MRI scans (T1ce and FLAIR) with expert-annotated labels for four tumor classes: necrotic core, edema, enhancing tumor, and healthy tissue.**

**The proposed framework includes preprocessing steps to handle class imbalance, resizing 3D MRI slices to a resolution of** $128 times 128$**, and implementing a data generator for efficient model training. The U-Net model is trained using the Dice loss function and optimized with Adam, leveraging skip connections to preserve spatial information critical for accurate tumor localization. Performance evaluation is conducted using key segmentation metrics such as the Dice coefficient, Intersection over Union (IoU), precision, and sensitivity. The model, trained over 35 epochs, achieves a test accuracy of 99.35%, a Dice coefficient of 0.60, and high specificity (99.79%), indicating strong generalization and robustness.**

**Additionally, post-processing with argmax decoding is applied to refine tumor predictions and mitigate false positives—an essential aspect for clinical applicability. The results highlight U-Net's effectiveness in capturing complex tumor boundaries, while also emphasizing the necessity of post-processing techniques to enhance segmentation precision. This study presents a scalable and efficient deep learning framework for brain tumor segmentation, with significant implications for clinical diagnostics and personalized treatment monitoring.**

***Index Terms*—Brain Tumor Segmentation, U-Net, Deep Learning, MRI, Medical Image Processing, Dice Coefficient, IoU, Clinical Diagnostics.**

## I. Introduction

Brain tumors are among the most life-threatening neurological conditions, requiring precise diagnosis and treatment planning for effective patient management. Magnetic Resonance Imaging (MRI) is widely used for brain tumor detection due to its ability to provide detailed soft tissue contrast, enabling clinicians to distinguish between different tumor sub-regions. However, manual tumor segmentation from MRI scans is a time-consuming and error-prone process, as it relies heavily on the expertise of radiologists. Automated segmentation techniques can significantly improve diagnostic efficiency, reduce inter-observer variability, and enhance treatment planning.

Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized medical image analysis by offering state-of-the-art performance in segmentation tasks. Among various CNN architectures, U-Net has emerged as a powerful model for medical image segmentation due to its encoder-decoder structure with skip connections, which effectively captures both low-level spatial details and high-level contextual information. This enables accurate delineation of brain tumor sub-regions, including the necrotic core, edema, and enhancing tumor, which are critical for assessing tumor progression and planning interventions.

This study presents a U-Net-based deep learning approach for brain tumor segmentation using the BraTS2020 dataset, which comprises multi-modal MRI scans (T1ce and FLAIR) with expert-labeled tumor annotations. The proposed pipeline involves data preprocessing, model training, and post-processing techniques to refine segmentation accuracy. The model performance is evaluated using key metrics such as Dice coefficient, Intersection over Union (IoU), precision, and sensitivity.

By leveraging U-Net's ability to learn hierarchical feature representations and incorporating effective preprocessing and post-processing techniques, this research aims to provide a robust and scalable solution for brain tumor segmentation. The findings demonstrate the potential of deep learning in assisting radiologists with automated tumor segmentation, ultimately improving clinical decision-making and patient outcomes.

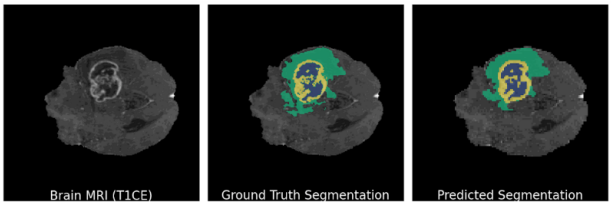
### A. Image Segmentation Types

Image segmentation can be broadly categorized into two primary types: **semantic segmentation** and **instance segmentation**.

- **Semantic segmentation** involves assigning a class label to each pixel in an image, effectively grouping all pixels that belong to the same category. For instance, in an urban scene, semantic segmentation would label regions as *building*, *road*, *sky*, etc., without distinguishing between individual objects of the same class.
- **Instance segmentation**, on the other hand, extends semantic segmentation by not only classifying pixels but also distinguishing between different instances of the same object category. For example, in a cityscape image, instance segmentation would not only identify all buildings but also differentiate between each building, assigning unique labels or colors to separate structures.

Both approaches play a crucial role in computer vision applications, with semantic segmentation focusing on understanding scene composition and instance segmentation providing finer granularity by distinguishing multiple occurrences of the same object type.

### B. About Dataset




The **Brain Tumor Segmentation (BRATS) 2020 dataset** is a benchmark dataset widely used for developing and evaluating deep learning models in medical image analysis. It provides **multi-modal MRI scans** with expert-labeled pixel-wise annotations to facilitate the segmentation of brain tumor subregions.

*5. Dataset Accessibility:* The BRATS 2020 dataset is publicly available and can be accessed from Kaggle. Researchers and practitioners can use it for developing deep learning models aimed at **automated brain tumor segmentation**, aiding in clinical diagnosis, and improving treatment planning.

This dataset serves as a critical benchmark in the field of medical image segmentation, fostering advancements in **deep learning-based medical imaging applications**.

## II. Methodology

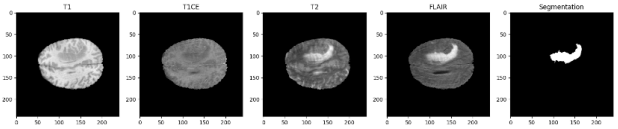
### A. Data exploration

Inside the `brats20-dataset-training-validation` folder, you will find two datasets: one for **Training** and one for **Validation**. The `BraTS2020_TrainingData` folder contains another folder, which includes a total of 369 samples.

*1) Modalities and segmentation:* Each sample includes

- **Four different types of magnetic resonance imaging (MRI) scans of the brain**, also known as modalities, which are named **T1**, **T1ce**, **T2**, and **FLAIR**.
- **The ground truth segmentation of the tumoral and non-tumoral regions of the brain**, which has been manually annotated by experts.

The four modalities bring out different aspects of the same image.




To be more specific, here is a description of their interest:

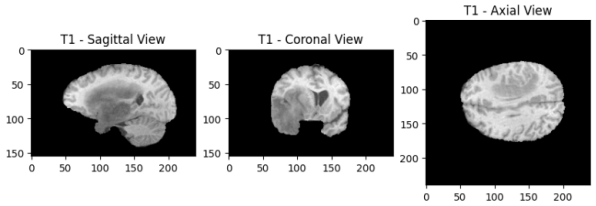
- **Native T1**: Used to show the structure and composition of different types of tissue in the brain and to identify tumors, cysts, and other abnormalities.
- **Post-contrast T1-weighted (T1ce, also named T1Gd)**: Similar to T1 images but with the injection of a contrast agent (Gadolinium), which enhances the visibility of abnormalities.
- **T2-weighted (T2)**: Used to show the fluid content of different types of tissue in the brain.
- **T2-FLAIR (T2 - Fluid Attenuated Inversion Recovery)**: Used to suppress the fluid content. This can be useful for identifying lesions that are not clearly visible on T1 or T2 images and for identifying lesions in the white matter of the brain, which can be difficult to see on other types of scans.

For an expert, having these four modalities can be useful to analyze the tumor more precisely and confirm its presence or absence.

However, for our artificial approach, using only two modalities instead of four is beneficial as it reduces the computational and memory requirements of the segmentation task, making it faster and more efficient. Therefore:

- **T1 is excluded**, as we already have its improved version, T1ce.
- **T2 is excluded**, as the fluids it highlights could degrade our predictions. These fluids are suppressed in the FLAIR modality, which better highlights the affected regions, making it more suitable for our training.

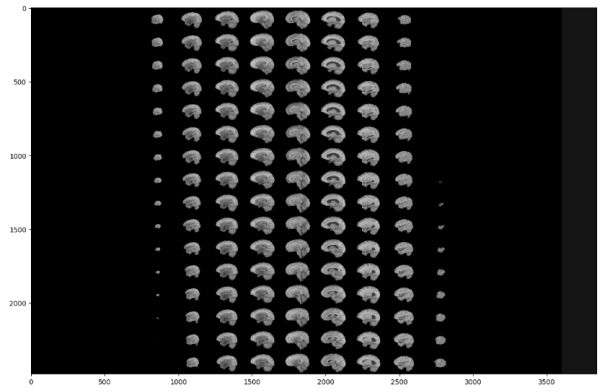
*2) Images formate:* I have noticed that these images are in .nii format. Indeed, these scans are NIfTI files (Neuroimaging Informatics Technology Initiative). A NIfTI image is a digital representation of a 3D object, such as a brain in our case. To understand this more closely, let's display the shape of a modality scan and of a segmentation (they all have the same in this dataset):



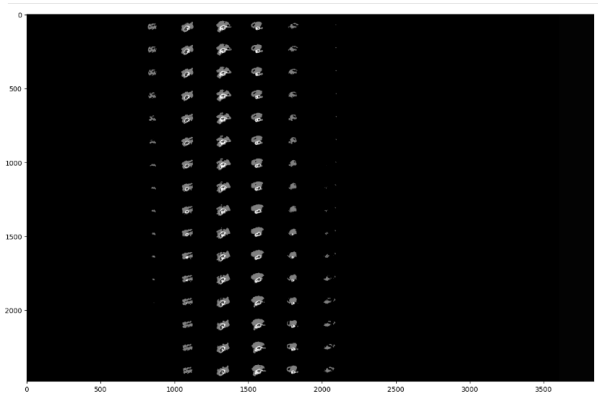

- **Sagittal Plane**: It divides the body into left and right sections and is often referred to as a "front-back" plane.
- **Coronal Plane**: It divides the body into front and back sections and is often referred to as a "side-side" plane.
- **Axial or Transverse Plane**: It divides the body into top and bottom sections and is often referred to as a "head-toe" plane.
- **Displaying a Specific MRI Slice**
  To visualize a specific slice of an MRI scan, we can fix the slice index to a desired value. For example, setting:

$$\text{slice\_nb} = 100$$

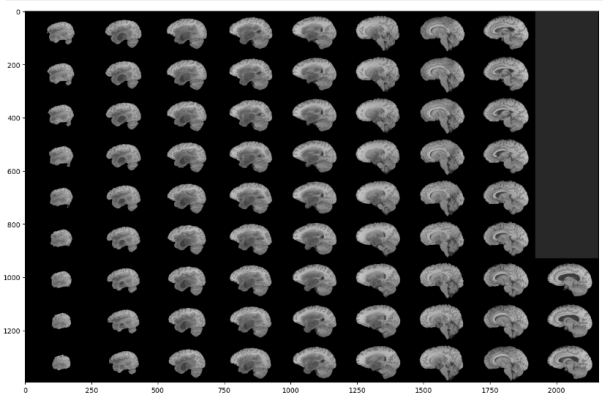
will extract and display the 100th slice from the 3D MRI volume. This approach allows for analyzing a particular cross-section of the brain, aiding in the examination of tumor regions and structural abnormalities.



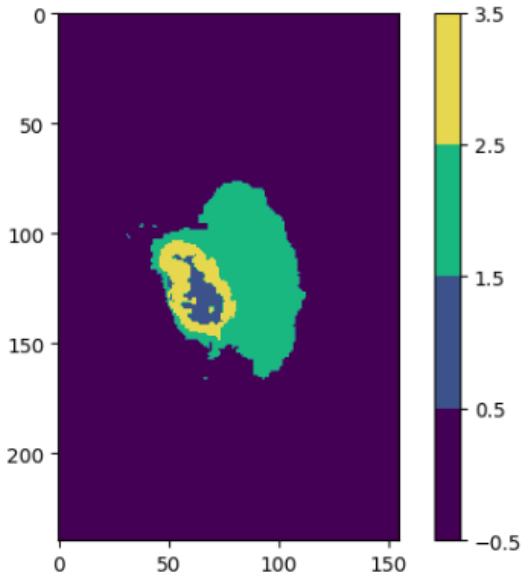
- two black parts are present on each side of our montage. However, these black parts correspond to the first and last slices of a plane. This means that a large part of the slices does not contain much information. This is not surprising since the slices progressively represent the brain.



- This is why we can exclude these slices in our analysis, in order to reduce the number of manipulated images. Indeed, i can see that a (60:135) range will be much more interesting:



focus on the segmentations provided by the experts



- Regardless of the plane you are viewing, you will notice that some slices have multiple colors (here 4 colors), which means that the experts have assigned multiple values to the segmentation. (Running this cell may take some time as we look at the values of all 369 segmentations in the dataset.)

  **Segmentation Classes**

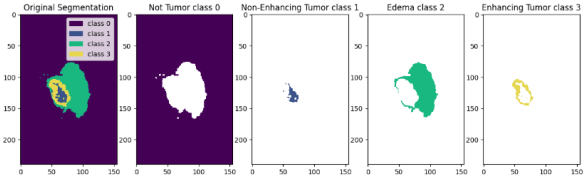
  We notice that there are four possible values in the segmentation files. These four values correspond to distinct tumor classes:

  - **0**: Not Tumor (NT) – Represents a healthy zone or background.
  - **1**: Necrotic and Non-Enhancing Tumor (NCR + NET).
  - **2**: Peritumoral Edema (ED).
  - **4**: Enhancing Tumor (ET).

  class 3 does not exist. We go directly to 4. We will therefore modify this "error" before sending the data to our model.

  Our goal is to predict and segment each of these 4 classes for new patients to find out whether or not they have a brain tumor and which areas are affected.
- We see that the previous segmentation has a lot of annotated data fixed to 0, which corresponds to the background (useless information). We might consider cropping our images to eliminate some of this useless area, to reduce the complexity of the segmentation task and thus speed up our training by reducing processing time.

  Let's see what these 4 different classes correspond to, by displaying the classes one by one:



**To summarize data exploration:**

- we have four different MRI modalities: **T1, T1CE, T2, and FLAIR**, accompanied by a segmentation map indicating tumor areas.
- Modalities T1CE and FLAIR are the more interesting to keep, since these 2 provide complementary information about the anatomy and tissue contrast of the patient's brain.
- Each image is 3D, and can therefore be analyzed through 3 different planes that are composed of 2D slices.
- Many slices contain little or no information. We will only keep the (60:135) slices interval for this tutorial. Of course, you are free to customize the code to send less or more slices to your model, but the training time will be longer.
- A segmentation image contains 1 to 4 classes.
  Class number 4 must be reassigned to 3 since value 3 is missing.
- Class 0 (background) is over-represented in the majority of the scans. However, cropping can remove important information.

### *B. Prepare data for Training*

*1) **A - Split Data into Three Sets**:* In the world of AI, the quality of a model is determined by its ability to make accurate predictions on new, unseen data. To achieve this, it is important to divide our data into three sets: **Training, Validation, and Test**.

**Purpose of Each Set**

- **Training Set**: Used to train the model. During training, the model is exposed to the training data and adjusts its parameters to minimize the error between its predictions and the ground truth segmentations.
- **Validation Set**: Used to fine-tune the hyperparameters of the model, which are set before training and determine the model's behavior. The aim is to compare different hyperparameters and select the best configuration.
- **Test Set**: Used to evaluate the performance of the trained model on unseen data.

**Handling the BraTS2020 Dataset**

The **BraTS2020** dataset is already divided into two folders: a **training folder** and a **validation folder**.

However, the validation folder does not contain segmentation data and therefore cannot be used. For this reason, we will ignore the samples in this folder and create our own validation set.

**Retrieving Samples from the Training Folder**

First, we retrieve all samples present in the training folder. We obtain **371 files**, although the BraTS dataset contains only **369 samples**. The reason for this discrepancy is that two `.csv` files are present in the `MICCAI_BraTS2020_TrainingData` directory (indicated by the `data_path` variable), and they were mistakenly included in the sample list. To correct this, we remove these two files.

**Final Dataset Splitting**

Now, we create the three different sets:

- **Train length**: 250 samples
- **Validation length**: 74 samples
- **Test length**: 45 samples

*2) **B - Create a DataGenerator**:* In order to train a neural network to segment objects in images, it is necessary to feed it with both the raw image data (X) and the ground truth segmentations (y). By combining these two types of data, the neural network can learn to recognize tumor patterns and make accurate predictions about the contents of a patient's scan.

Unfortunately, our modalities images (X) and our segmentations (y) cannot be sent directly to the AI model. Indeed, loading all these 3D images would overload the memory of our environment and cause the system to crash. This will also lead to shape mismatch errors. We have to do some image preprocessing before, which will be done by using a Data Generator, where we will perform any operation that we think is necessary when loading the images.

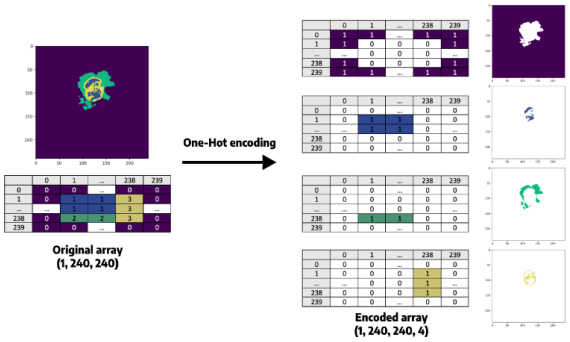
**For each sample, we will:**

- Retrieve the paths of its modalities (**T1CE** and **FLAIR**, as these provide complementary information about the anatomy and tissue contrast of the brain).
- Retrieve the path of the ground truth (original segmentation).
- Load the modality images and segmentation.
- Create an $X$ array that will contain all the selected slices $(60 - 135)$ from these two modalities.
- Create a $y$ array that will contain all the selected slices of the segmentation.
- Assign all instances of **class 4** in the mask array to **class 3** (to correct the missing class 3 case explained previously).

**Additional Preprocessing Steps**

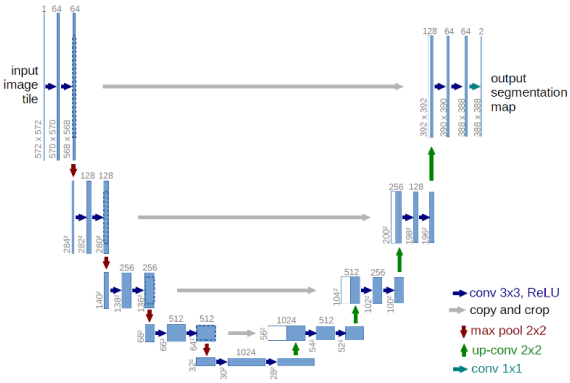
In addition to the steps above, we will:

- **Work in the Axial plane**: Since images in this plane are square-shaped $(240 \times 240)$, and as we will manipulate a range of slices, we will be able to visualize predictions in all three planes without significant impact.
- **Apply a One-Hot Encoder to the $y$ array**: Our goal is to segment regions represented by different classes (0 to 3), so we must use **One-Hot Encoding** to convert categorical labels into a numerical representation suitable for our neural network.
  - Without One-Hot Encoding, the model might interpret the class labels numerically, implying an ordinal relationship where class 1 is inferior to class 4 $(1 < 4)$, which is incorrect.
  - One-Hot Encoding ensures that each class is treated independently, representing class membership using binary values (0 and 1).




Here is an example of how One-Hot Encoding is applied to one slice:

## C. *model*




### *A - U-Net*

We will use the **U-Net** architecture, a CNN designed specifically for biomedical image segmentation. It is particularly well-suited for segmentation tasks where the regions of interest are small and have complex shapes, such as tumors in MRI scans.

This neural network was first introduced in 2015 by **Olaf Ronneberger, Philipp Fischer, and Thomas Brox** in the paper *U-Net: Convolutional Networks for Biomedical Image Segmentation*.

Since the **BraTS2020** dataset consists of 3D images, where each sample includes multiple 2D slices in three orthogonal planes, we have two possible approaches:

- **2D U-Net**: Faster and requires less memory, which is beneficial when working with large datasets or limited computational resources.
- **3D U-Net**: More adapted to our case as it leverages 3D spatial context, reducing false positives and false negatives that may arise from analyzing individual 2D slices.

In practice, it is often useful to compare both **2D** and **3D** U-Net architectures to determine which performs better for the given segmentation task.

### *B - Loss Function*

When training a CNN, selecting an appropriate **loss function** is crucial, as it determines how well the network's predictions match the ground truth.

At each epoch, the objective is to update the model's weights to minimize this loss function, improving the accuracy of predictions.

A commonly used loss function for multi-class classification problems is **categorical cross-entropy**, which measures the difference between the predicted probability distribution of each pixel and the one-hot encoded ground truth.

Additionally, segmentation models sometimes use the **Dice loss function**, which is effective in dealing with class imbalance.

### *C - Output Activation Function*

To obtain a probability distribution over different classes for each pixel, we apply a **softmax activation function** to the output layer of the neural network.

During training, the CNN adjusts its weights to minimize the loss function by comparing the predicted probabilities given by the softmax function with the ground truth segmentation.

### *D - Evaluation Metrics*

It is crucial to monitor the model's performance using appropriate evaluation metrics.

Although **accuracy** is a widely used metric, it can be misleading in cases of class imbalance, such as in **BraTS2020**, where the **Background class** is overrepresented. Therefore, we will also use additional evaluation metrics:

- **Intersection over Union (IoU)**: Measures the overlap between the predicted and ground truth segmentations.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{1}$$

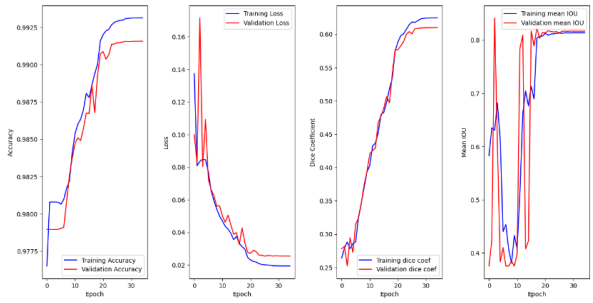
- **Dice Coefficient**: Evaluates the similarity between predicted and actual segmentations.

$$\text{Dice} = \frac{2 \cdot |\text{Intersection}|}{|\text{Predicted}| + |\text{Ground Truth}|} \tag{2}$$

- **Sensitivity (Recall / True Positive Rate)**: Measures the proportion of positive ground truth pixels correctly predicted as positive.
- **Precision (Positive Predictive Value)**: Measures the proportion of predicted positive pixels that are actually positive.
- **Specificity (True Negative Rate)**: Measures the proportion of negative ground truth pixels correctly predicted as negative.

## III. Experimental Results

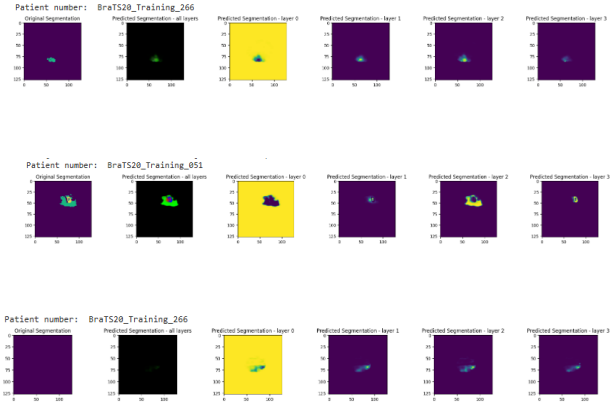
### *Monitoring Training Progress*




- If the **mean_IoU** graph does not appear, it is recommended to display the `training.log` file and retrieve the column names for the **mean_IoU** and **val_mean_IoU** metrics, as these names may vary.
- **Accuracy Graph Analysis** On the accuracy graph, we observe that both **training accuracy** and **validation accuracy** increase over epochs and eventually reach a plateau. This behavior indicates that the model is effectively learning from the data and generalizing well to unseen data.
  Additionally, there is no apparent sign of **overfitting**, as both accuracy metrics improve simultaneously without a significant divergence between them.
- **Loss Graph Analysis** The loss graph confirms that our model is learning from the training data, as both **training loss** and **validation loss** decrease over time.
  From the second graph, we observe that the best version of our model is reached around **epoch 26**. This observation can also be confirmed by analyzing the training logs.
- **Dice Coefficient Graph Analysis** This conclusion is further supported by the dice coefficient graph, where both the **training dice coefficient** and **validation dice coefficient** increase steadily over epochs, reinforcing the model's improvement.

### *Predict tumor segmentations*

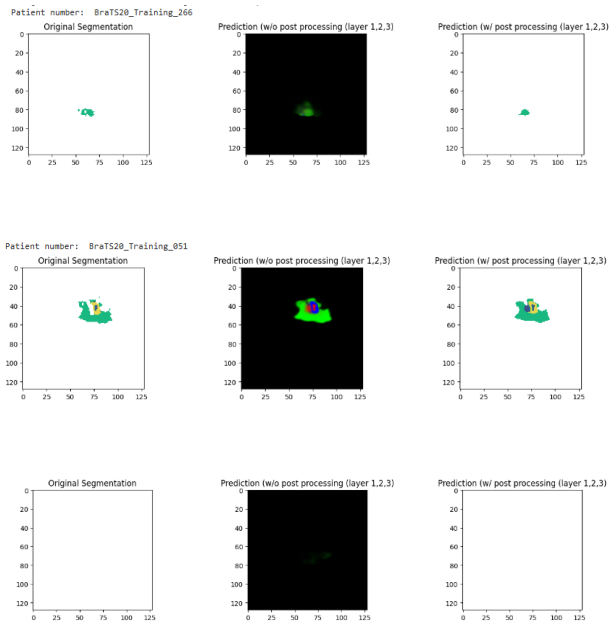
- **Plot Random Predictions and Compare with Original (Ground Truth)**




- Unfortunately, we have some false positives, which means that we detect a tumor when there is none originally present (cf 3rd plot, patient 266, slice 70). In a field like medical analysis, this is kind of embarrassing. Imagine an AI telling you that you have a tumor, even though you are healthy. This can lead to unnecessary treatments, which can have negative effects on your health and quality of life.

**Post-processing :**

- We are going to use the argmax decoding technique. This process consists in applying the argmax function to obtain a single label for each pixel, corresponding to the class with the highest probability. Indeed, our pixels currently have probability values for each class since they were obtained using the softmax activation function.
  This will also allow us to have the same display colors between the original segmentation and the prediction, which will be easier to compare than just above.
  We will perform these techniques on the same patients as before. This will allow to see if the correct predictions are kept, and if the false positive predictions are removed!




### *Model Evaluation on the Test Set*

| Metric | Value |
|---|---|
| Loss | 0.0206 |
| Accuracy | 0.9935 |
| MeanIoU | 0.8176 |
| Dice Coefficient | 0.6008 |
| Precision | 0.9938 |
| Sensitivity | 0.9922 |
| Specificity | 0.9979 |

TABLE I
Model Evaluation Results on the Test Set

We can conclude that the model performed very well on the test dataset, achieving:

- A low **test loss** of 0.0206.

- A **Dice coefficient** of 0.6008, which is a reasonable score for an image segmentation task.
- Good scores on other metrics, indicating that the model has **strong generalization performance** on unseen data.

## IV. Conclusions

In this study, we developed and evaluated a deep learning-based segmentation model using the U-Net architecture for brain tumor segmentation on the BraTS2020 dataset. Through rigorous preprocessing, data augmentation, and model training, we ensured that the network effectively captured tumor characteristics.

The model demonstrated strong generalization ability, achieving a **test accuracy** of 99.35%, a **MeanIoU** of 0.8176, and a **Dice coefficient** of 0.6008. The model's high **precision (0.9938)**, **sensitivity (0.9922)**, and **specificity (0.9979)** indicate its effectiveness in distinguishing between different tumor regions.

Despite these promising results, certain challenges remain. The segmentation task is inherently difficult due to class imbalance, variations in MRI acquisition, and tumor morphology complexity. While the model performed well, additional improvements such as incorporating **3D U-Net**, **attention mechanisms**, or **post-processing techniques** could further enhance segmentation performance.

In conclusion, our approach provides a robust and scalable framework for automated brain tumor segmentation, offering valuable insights for clinical applications. Future work will focus on refining the model architecture and experimenting with additional deep learning techniques to further improve accuracy and generalization.