**Brain Tumor Segmentation**

**Problem Statement:**

The goal is to segment brain tumours from MRI scans using deep learning. Each scan is multimodal, offering multiple views of the same brain region, each with unique imaging characteristics. The objective is to identify and classify different tumour regions accurately.

**Dataset Information:**

**Dataset Link:**  [**https://www.kaggle.com/datasets/awsaf49/brats2020-training-data**](https://www.kaggle.com/datasets/awsaf49/brats2020-training-data)

These are **.nii** files and are in 3D format.

These are multimodal scans.  
Multimodal → 4 different views from the same region (i.e., image with different modality).

Modality refers to different types of imaging techniques to retrieve different information.  
Ex: One modality might focus on the brain’s structure, while another focuses on blood flow.

**Annotations:**

* Label 0 → Unlabelled volume
* Label 1 → Necrotic & non-enhancing tumour
  + Necrotic → Tumour that has died due to lack of blood supply
  + Non-enhancing tumour → Less aggressive tumour tissue that is likely to respond to certain treatments
* Label 2 → Peritumoral edema
  + Peritumoral edema → Fluid around the tumour
* Label 4 → GD-enhancing tumour
  + Aggressive part of the tumour
* Label 3 → No pixels  
  ∴ We need to reassign the label 3 to label 4.

**We have 4 modalities:**

* **T1** → Brain anatomy, difference between the grey and white matter
* **T1CE** → Where the blood-brain barrier might be damaged, such as tumours or inflammation
* **T2 weighted** → Highlights the cerebrospinal fluid and identifies areas of abnormal fluid accumulations (edema)
* **FLAIR** → Removes cerebrospinal fluid to make abnormalities in the brain visible

We are not using T1, as we get more additional features from the T1CE + T2 information.

All the datasets were manually segmented and approved by the neuroradiologists.

**Data Preprocessing:**

**Step 1:**

The datasets are in the file format of **.nii** (NIFTI). The **.nii** (NIfTI) is a file format for storing medical images like MRI or CT scans. It contains:

* 3D image data (like brain scans)
* Info about image size, spacing, orientation

At first, these 3D images need to be normalized. Normalization can be done using the **MinMaxScaler()** but it only accepts 2D data and not 3D.

Hence, at first, 3D images are converted into 2D images and then normalized. After normalization, the scaled images are restored back into the same 3D shapes.

**Step 2:**

As we have 4 types of modalities, we have also noticed that the information we can get from T1 (modality) can also be obtained from the T1CE modality with additional information too. So, to increase computational efficiency, we are not going to use T1 for the training purposes.

The process of normalization is done on all three modalities (FLAIR, T1CE, T2) and they were combined into a single NumPy file by stacking one on another.

**Step 3:**

When we are looking into each image, most of the areas are black (which are not part of the brain). So, these areas may affect the model training and its accuracy. Hence, we are going to crop the images to some extent such that black (vast areas) are removed.

And we are using the data/images which contain at least 1% of valid image information compared to the entire image.

**Summary:**  
These steps are performed for the entire dataset and found 344 valid data points in the whole set of 369 data points.

**Custom Data Generator:**

Why custom data generator?  
Default Keras data generator doesn't handle the npy arrays. And we are handling the npy arrays. So, we are creating a custom data generator that loads the data into the model for training.

So, at the end, the custom data generator will help you yield a set of batches for training and each batch contains (an image, respective mask). The structure of the batch would be [(1, height, width, channels), (1, height, width, channels)] [if batch\_size=1, and in the shape, 1 represents the batch size].

**Visualization of the Inputs:**

**If we have:**

- Dataset with 100 patients

- Each scan of size 240×240×155 pixels

- 3 MRI modalities (FLAIR, T1CE, T2)

**The code might:**

- Select patient #45

- Choose slice #78

- Display a 2×2 grid showing:

- FLAIR image of slice 78

- T1CE image of slice 78

- T2 image of slice 78

- Corresponding segmentation mask where different colours represent different tumour regions

**Model Training, Loss Function, Optimizer Setup:**

As we are training the model in the Kaggle environment, we may face a challenge where during the training we may not get a correct corresponding mask for a respective input image. Hence, we built a load\_aligned\_data, which will help us to get the image and its respective mask correctly.

* **Categorical Focal Loss** → Focuses more on the hard-to-classify area. Adds more contribution toward the loss of hard (features) misclassified samples than the loss of easy (features) misclassified samples. Controlled by the 'γ' parameter.
* **Dice Loss** → Loss function commonly used in the image segmentation process. Measures the similarity between the predicted segmentation and the ground truth. It’s useful for imbalanced data where the foreground is much smaller than the background.
  + **Foreground**: Area with the object of interest (e.g., tumour)
  + **Background**: Everything else (non-tumour)
* **IOU** → Used in the accuracy of object detection or image segmentation process. It measures the overlap between the predicted region and the ground truth region. Higher the value, higher the accuracy.
* **Adam Optimizer** → Combination of two optimizers (Momentum + RMSprop) to achieve fast convergence and robust performance. Moving average of the gradient and moving average of the squared gradient.

**Example**: For a brain tumour image:

* Dice Loss ensures good overlap between predicted and actual tumour regions
* Focal Loss helps identify small tumour regions
* IOU Score of 0.8 would mean 80% overlap between prediction and ground truth
* Adam optimizer adjusts learning rates automatically for better convergence

**Defining the Architecture**

* ReLU → Activation function for the architecture to avoid vanishing gradient descent problem
* he\_uniform → A specific weight ini tialization technique working well along with the ReLU
* MaxPooling3D → Used to reduce the spatial dimension of the input volume while maintaining only the important features

**U-Net Architecture:**

[To have a detailed understanding of the U-Net and also the implementation, refer to the Notebook]

U-Net is a deep learning model used mainly for image segmentation, where the goal is to classify each pixel in an image.

* Encoder → Down-sampling using convolutional layers by detecting patterns
* Decoder → Up-sampling using the transposed convolutional layers with the help of skip connections (encoder's feature)
* Skip Connections → Connects the encoder layer to the decoder layer. Helps to preserve spatial information lost during the down-sampling.

**Metrics After Training:**

* Accuracy: 98.50%
* IOU Score: 0.6946
* Dice Loss: 0.8059
* Mean IOU: 0.5183

**Saving the Model**

* **.h5 format**: Traditional format storing both model and architecture
* **TensorFlow (.tf) format**: Useful for deployment and portability