MRI SEGMENTATION

DATASET -

Brain tumour (BT) diagnosis is a lengthy process, and great skill and expertise are required from radiologists. As the number of patients has expanded, so has the amount of data to be processed, making previous techniques both costly and ineffective.
For the purpose of developing and testing BT segmentation and diagnosis algorithms, the brain tumour segmentation (BraTS) dataset was produced.
In the BraTS dataset, mri files are present as *.nii.gz* files.
.nii : This indicates that the files contain medical imaging data stored in the NIfTI (Neuroimaging Informatics Technology Initiative) format. NIfTI is a widely used standard format for storing and sharing neuroimaging data.
.gz : This indicates that the file has been compressed using gzip compression, which helps reduce the file size and makes it easier to store and transfer large volumes of medical image data.
We will be using this dataset to perform Brain MRI segmentation.

This study discusses different MRI pulse sequences used for medical imaging in the context of the BraTS dataset, a dataset used for investigating brain tumors (BTs). As shown in Figure 2, the modalities discussed include T1-weighted (T1W), T2-weighted (T2W), fluid-attenuated inversion recovery (FLAIR), and T1-weighted with contrast enhancement (T1CE) MRI.

PREPROCESSING -

We know that MRI is a volumetric scan, which implies it provides us with 3D data. However, this 3D data is provided in the form of multiple 2D images (or slices) stacked one after another.
The model we will be creating would work on 2D images. In each file there are 155 2D slices, so for simplicity we will be selecting the middle slice for our model.

Now, as there are 4 pulse sequences present in the dataset for a single MRI. Each pulse sequence provides different kinds of information, and combining them can lead to better insights and more accurate results.
So, to create a multichannel input for our model, we need to stack all these together.

MODEL DESCRIPTION

The U-Net architecture is a Convolutional Neural Network (CNN) designed to yield precise segmentation results, especially in medical imaging.
The U-Net architecture consists of a symmetric encoder-decoder structure with a series of convolutions, pooling, and upsampling operations, designed in a U-shape.
In this model we will just be using 3x3 convolutional layers.
It has three main parts:

1. Contracting Path (Encoder)
   The encoder is responsible for capturing the context. The encoder path reduces the spatial resolution and it also increases the depth of the feature maps thereby capturing increasingly abstract representations of the input. This contracting path is similar to the feedforward layers in other convolutional neural networks
2. Bottleneck
   At the bottom of the "U", the bottleneck connects the encoder and decoder.
3. Expanding Path (Decoder)
   The decoder is responsible for precise localisation using transposed convolutions.
   The decoder works on decoding the encoded data and locating the features while maintaining the spatial resolution of the input. The decoder layers in the expansive path

upsample the feature maps, while also performing convolutional operations.
The skip connections from the contracting path help to preserve the spatial information lost in the contracting path, which helps the decoder layers to locate the features more accurately.

MODEL ARCHITECTURE

1) Encoder Path:
Each encoder block consists of two Convolutional layers followed by Batch Normalization and ReLU activation.
Max-pooling is applied after each encoder block to reduce the spatial dimensions of the feature maps.
2) Bottleneck:
The bottleneck layer represents the lowest level of the U-Net, where the feature maps are at their smallest spatial dimensions but have the most channels.
3) Decoder Path:
Each decoder block consists of an up-convolution (transposed convolution) to upsample the feature maps followed by concatenation with the corresponding feature maps from the encoder path (skip connections). Each concatenated feature map is then processed through a convolutional block similar to the encoder.
4) Final Convolution:
A 1x1 convolutional layer is used at the end to map the output to the desired number of channels.

Forward Pass

1. Encoder Forward pass
   The input passes through the encoder blocks and pooling layers.
2. Bottleneck:
   The encoded feature maps are processed through the bottleneck block.
3. Decoder Forward Pass:
   The decoder starts by upsampling the feature maps and concatenating them with the corresponding encoder feature maps (skip connections). The concatenated feature maps are then processed through the decoder blocks.
4. Final Convolution:
   The final output is generated by passing the feature maps through a 1x1 convolutional layer to get the desired number of output channels.

Utility Method
The _block method is a static method that defines a convolutional block used in both the encoder and decoder. Each block consists of:

1. Two convolutional layers with 3x3 kernels and padding of 1.
2. Batch normalisation layers after each convolution.
3. ReLU activation functions for non-linearity.

Training the model

To train the model:

1. Prepare the Dataset and DataLoader : Call the BraTSDataset and store X and y in the dataset. Then, use the inbuilt DataLoader function from the module class.

2. Initialise Model, Loss Function, and Optimiser : Initialise the model as UNET(), define the loss function as nn.CrossEntropyLoss(), and initialise the optimiser.

   CrossEntropyLoss $-\sum_{c=1}^{N} y_c log(p_c)$

3. Train and Save the Model : Implement the training loop with 10 epochs, then save the trained model.

Testing

The plot_segmentation_masks function is designed to display three images side by side for visual comparison: the input MRI image, the Predicted Segmentation Mask, and the Ground Truth Segmentation Mask.

Performance metrics

When evaluating segmentation models, especially in medical imaging, several performance metrics are commonly used to measure how well the model performs. Metrics used for evaluation are:

1. Accuracy: Measures the proportion of correctly predicted pixels out of the total pixels.
   Accuracy = Number of correctly predicted pixels / Total number of pixels

2. Dice Coefficient: The Dice coefficient, also known as the Sørensen-Dice index, is a statistical tool used to gauge the similarity between two sets of data.

3. $Dice(A,B) = 2|A \cap B| / (|A|+|B|)$
   where,
   A and B are two sets. $|A \cap B|$ is the number of elements common in sets A and B.
   $|A|$ and $|B|$are the number of elements in sets $A$and B, respectively