The contributions of the authors for the diagnosis and classify of brain tumors in the manuscript are as follows:

 Preprocessing of MRI images with Otsu thresholding:

Preprocessing of MRI (Magnetic Resonance Imaging) images is a crucial step in image analysis to enhance image quality, reduce noise, and facilitate more accurate and meaningful analysis. Otsu thresholding is a specific image segmentation technique that can be employed during the preprocessing of MRI images. Here's a brief overview of the purpose of preprocessing with Otsu thresholding:

1. Image Segmentation: Otsu's method is a popular image segmentation technique used to separate an image into different regions or objects based on intensity values. In the context of MRI images, segmentation is often employed to distinguish between different tissues or structures within the image, such as distinguishing between gray matter, white matter, and cerebrospinal fluid in brain MRI.

2. Thresholding: Otsu's method calculates an optimal threshold value to separate pixels into two classes, typically foreground and background, based on the pixel intensity distribution in the image. This thresholding helps in emphasizing the boundaries between different structures or tissues, making subsequent analysis tasks more effective.

3. Noise Reduction: MRI images can be affected by various types of noise, including random noise and artifacts. Otsu thresholding can help in reducing noise by focusing on the most significant intensity values and ignoring low-intensity noise. This can enhance the overall quality of the image for further analysis.

4. Feature Extraction: After segmentation with Otsu thresholding, it becomes easier to extract meaningful features from specific regions of interest within the MRI image. These features can be crucial for subsequent analysis tasks such as tumor detection, tissue classification, or volumetric measurements.

5. Improved Visualization: Otsu thresholding can enhance the contrast and visibility of structures within an MRI image, making it easier for clinicians and researchers to visually interpret and analyze the images.

 Extracting features of brain tumors with：

It was explained earlier.

 DenseNet121:

The key idea behind DenseNet is dense connectivity, where each layer is connected to every other layer in a feed-forward fashion. In traditional CNN architectures, information flows sequentially from one layer to the next. However, in DenseNet, each layer receives the feature maps from all preceding layers. This dense connectivity has several advantages, such as reducing the vanishing gradient problem, promoting feature reuse, and improving the flow of information through the network. DenseNet121 specifically refers to a DenseNet model with 121 layers. The number "121" indicates the total number of layers in the network, including convolutional, pooling, normalization, and fully connected layers. Key components and characteristics of DenseNet121:

Dense Blocks: DenseNet is composed of dense blocks, where each layer in a block receives feature maps from all preceding layers in the same block. This dense connectivity leads to a highly parameter-efficient network.

Transition Blocks: Between dense blocks, transition blocks are used to reduce the spatial dimensions (width and height) of the feature maps, typically through pooling, and also reduce the number of channels. This helps in controlling the model complexity and computational cost.

Bottleneck Layers: DenseNet121, like other DenseNet variants, employs bottleneck layers within each dense block. These bottleneck layers consist of 1x1 convolutional layers to reduce the number of input channels before the 3x3 convolutional layer.

Global Average Pooling (GAP): Instead of using fully connected layers at the end, DenseNet architectures often use global average pooling, which averages the spatial dimensions of each feature map, producing a fixed-size output. This reduces the number of parameters in the model and helps prevent overfitting.

Batch Normalization and ReLU Activation: DenseNet uses batch normalization and rectified linear unit (ReLU) activation functions to stabilize and activate the network during training.

 Convolutional layers in bridge layer:

We use double Convolution with double layer optimization for input and output of bridge, each Convolution section has ELU activation, The Exponential Linear Unit (ELU) is an activation function used in artificial neural networks. It was introduced to address some of the limitations of the widely used Rectified Linear Unit (ReLU) activation function. The ELU function is defined as follows:

$$ELU(x) = \begin{cases} x & ,if \quad x \geq 0 \\ \alpha \cdot (e^x - 1), if & x \geq 0 \end{cases}$$

In this formula:

x is the input to the function.

α is a hyperparameter controlling the slope of the function for $x < 0$.

e^x is the exponential function.

Smoothness and Continuity: Unlike ReLU, which has a sharp transition at zero, ELU is smooth and differentiable everywhere, including at $x = 0$, This smoothness can be beneficial during gradient-based optimization.

Avoiding "Dead Neurons": In ReLU, neurons with negative outputs are inactive (dead neurons) and do not contribute to the learning process. ELU helps mitigate the issue of dead neurons by allowing negative values and ensuring non-zero gradients for all inputs.

Asymmetry: ELU is asymmetric around the origin, with negative values approaching −α as x approaches negative infinity. This asymmetry helps prevent the vanishing gradient problem and allows the network to learn representations with negative values.

Parameter α: The parameter α controls the value to which the ELU function saturates for negative inputs. A common choice for α is 1.0, but it can be adjusted based on the characteristics of the data and the network.

Range of Output: The output of the ELU function is not bounded, which means it can take on both positive and negative values. This can be advantageous in certain scenarios, but it may also introduce challenges in networks that require outputs to be within specific ranges.

 Optimized Unet++:

U-Net++ is an extension of the original U-Net architecture, which is widely used for semantic segmentation tasks, such as image segmentation or medical image analysis. The U-Net++ architecture was proposed to enhance the segmentation performance by addressing some of the limitations of the original U-Net. Key features of U-Net++ include:

Nested Skip Pathways: U-Net++ introduces nested skip pathways to capture multi-scale contextual information. It adds multiple levels of skip connections between encoder and decoder blocks, allowing the network to capture features at different resolutions.

Skip-Harmonic Connections: U-Net++ uses skip-harmonic connections to harmonize the features extracted by different levels of the network. This is intended to improve the integration of information from multiple scales.

Hyperconnected Skip Pathways: Hyperconnected skip pathways are introduced to connect layers not only within the same resolution level but also across multiple resolution levels. This helps in capturing richer contextual information.

Reduced Semantic Gap: The design of U-Net++ aims to reduce the semantic gap between encoder and decoder blocks, allowing for more effective feature propagation and segmentation.

 Classification of images in the fully connected layer of Unet++:

In a U-Net architecture, the fully connected layer is typically not present in the traditional sense as it is in fully connected neural networks. U-Net is a convolutional neural network (CNN) designed for semantic segmentation, where the goal is to classify each pixel in an image into different classes. The network consists of an encoder path that captures context and a decoder path that performs upsampling to generate a segmentation mask.

In U-Net, the final output is a segmentation mask, not a classification for the entire image. Each pixel in the output corresponds to a particular class, and the softmax activation function ensures that the probabilities across all classes sum to 1 for each pixel.

So the output layer usually consists of a convolutional layer with a softmax activation function. This final layer provides pixel-wise probabilities for each class in the segmentation task.