Documentation on implementation of ResNet – 50 architectures for Brain Tumor Classification

Background:

A brain tumor is a growth of brain cells or cells close to the brain. The tissue of the brain can develop brain tumors. Near the brain tissue, brain tumors are also possible. The pituitary gland, pineal gland, and membranes that surround the surface of the brain are nearby structures.
Magnetic resonance imaging (MRI) is one of the procedures used to identify brain tumors, however it can be difficult to determine the tumor. Different tumor types respond differently to treatment.

Dataset:

In Kaggle, a brain MRI Image Dataset was created. The dataset is divided into two categories, Training and Testing, which house various Brain MRI scans. Glioma, meningioma, no tumor, and pituitary are the four subfolders that make up the Training folder. A total of 5712 photographs are in the Training folder, while 1310 images are in the Testing folder.

Data Preparation:

Data Augmentation: Data augmentation uses existing data to create modified copies of datasets, which are then used to artificially increase the training set. It comprises making little adjustments to the dataset or creating new data points using deep learning.
There were not enough samples to train the neural network because of the limited size of the dataset. Additionally, data augmentation helped to address the problem of data imbalance.

- The collection originally had 5712 brain MRI scans.
- The dataset currently includes 7022 brain MRIs because of data augmentation.
- The testing dataset is unaffected, and these numbers are for the training dataset.

Data Preprocessing: The following preprocessing procedures were used on each image:

- Cut out the area of the photo that solely shows the brain, which is the most significant portion of the photo:
- Because the images in the collection come in various sizes, resize the image to fit the following shape: (224, 224, 3) = (image_width, image_height, number of channels). To feed it as input to the neural network, all images must have the same shape.
- Apply Normalization: converting pixel values to a scale between 0 and 1.

Data Split: The following ways in which the data were divided:

- 80% of the information is for training.
- 20% of the data are used for development and validation.
- The testing dataset is kept and used separately.

The Neural Network Architecture:

ResNet -50

What is the ResNet – 50 model?

He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian's 2015 study "Deep Residual Learning for Image Recognition" developed the convolutional neural network (CNN) variant known as ResNet. Applications using computer vision frequently use CNNs.

One MaxPool layer, one average pool layer, and 48 convolutional layers make up the 50-layer convolutional neural network known as ResNet-50. Artificial neural networks (ANNs) that use residual blocks to build networks are known as residual neural networks.

Advantage of using ResNet - 50:

Deeper networks: ResNet50 can be deeper than other models because it uses residual connections to skip over some layers. This allows for the creation of deep neural networks that are easier to optimize and can achieve better performance.

Better performance: ResNet50 has achieved state-of-the-art performance on many image classification tasks. This is partly due to its deeper architecture, but also because it uses batch normalization, which helps to stabilize the training process and improve generalization.

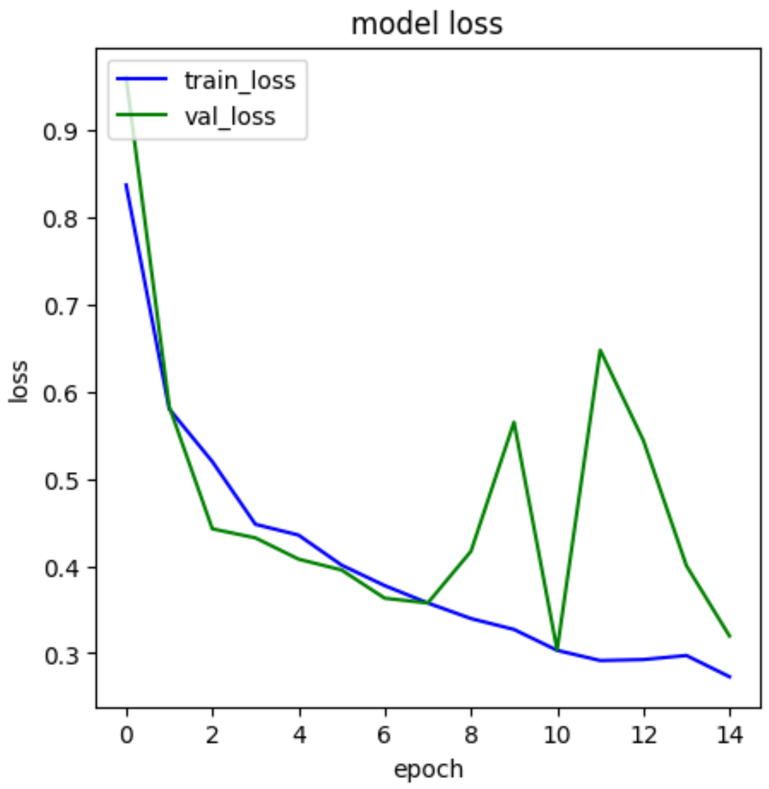
Faster convergence: ResNet50 has been shown to converge faster than other deep neural network models, meaning that it requires less time to train and can achieve better results with less data.

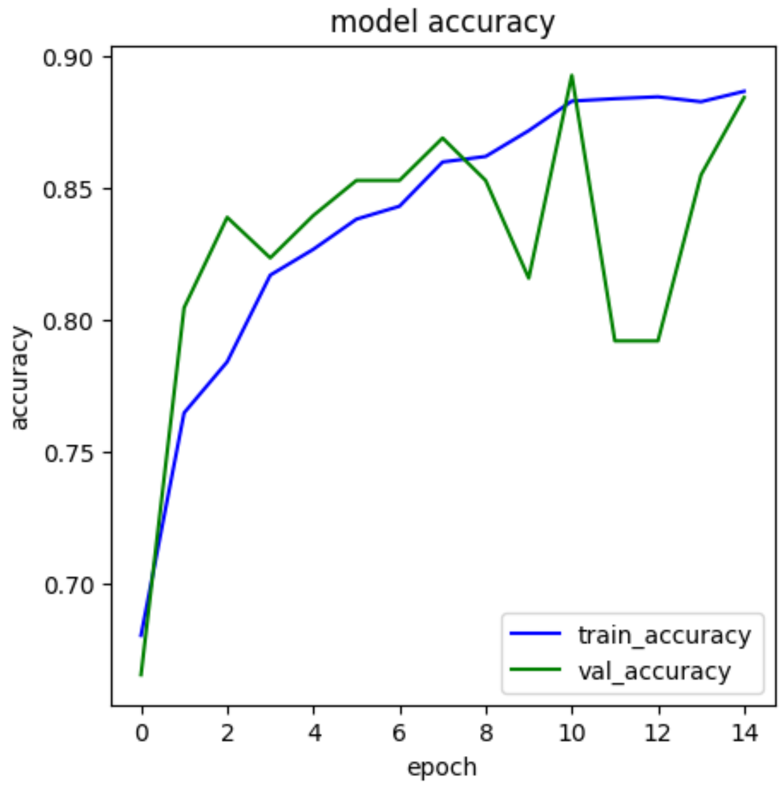
Improved accuracy: ResNet50 has been shown to achieve higher accuracy on image classification tasks than other deep neural network models, such as VGG and Inception. This is partly due to its use of residual connections, which enable the model to learn more complex features.

Transfer learning: ResNet50 has been pre-trained on a large dataset, which makes it a good starting point for transfer learning. Transfer learning allows you to fine-tune the pre-trained model on a smaller dataset, which can save time and resources while still achieving good performance.

Training the model:

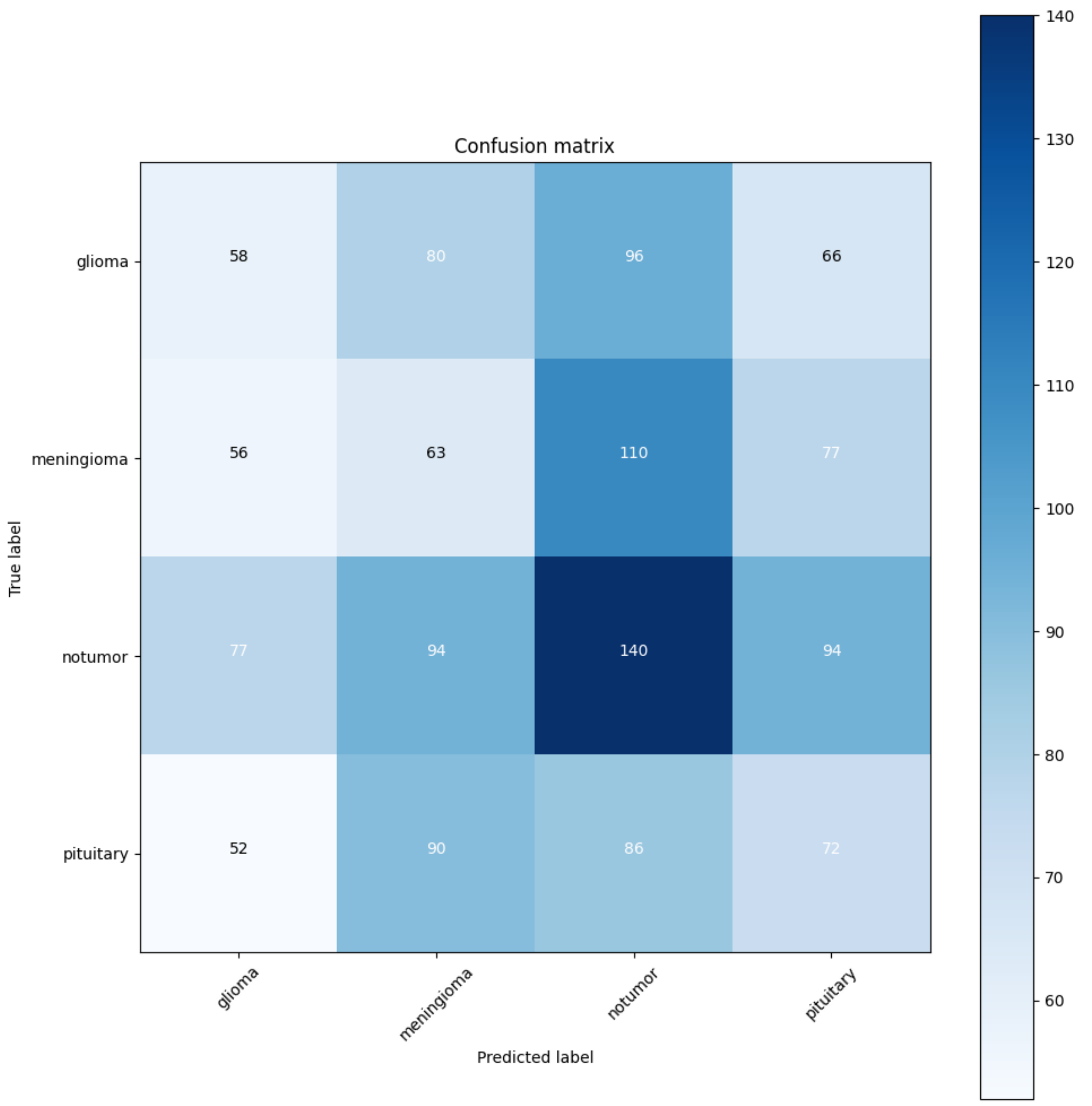
The model was trained for 15 epochs, and these are the loss & accuracy plots:



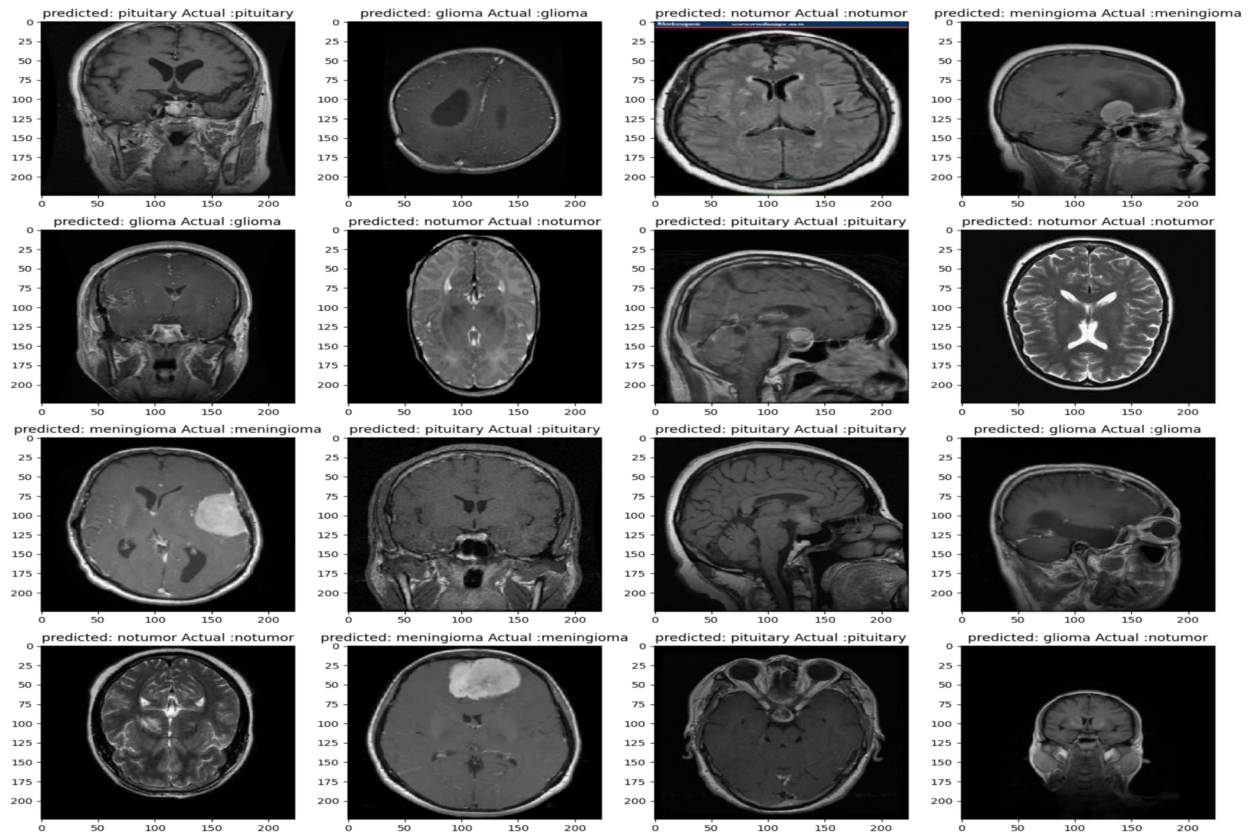


<u>**Confusion Matrix:**</u>

- We generate it to visualize important predictive analytics like recall, accuracy, specificity, and precision.
- The diagonal elements present are the 'true predictions' that we were able to make.



<u>Predicted v/s Actual for ResNet – 50:</u>


predicted: pituitary Actual :pituitary
predicted: glioma Actual :glioma
predicted: notumor Actual :notumor
predicted: meningioma Actual :meningioma
predicted: glioma Actual :glioma
predicted: notumor Actual :notumor
predicted: pituitary Actual :pituitary
predicted: notumor Actual :notumor
predicted: meningioma Actual :meningioma
predicted: pituitary Actual :pituitary
predicted: pituitary Actual :pituitary
predicted: glioma Actual :glioma
predicted: notumor Actual :notumor
predicted: meningioma Actual :meningioma
predicted: pituitary Actual :pituitary
predicted: glioma Actual :notumor

Results:

Based on the information provided, it appears that the ResNet-50 model developed has achieved reasonably good accuracy rates.

- The training accuracy rate of 92.06% indicates that the model has learned to classify the data with a high degree of accuracy.
- The testing accuracy rate of 86.34% suggests that the model can generalize well to new, unseen data.
- The validation accuracy rate of 88.44% indicates that the model has not overfit the training data and is able to generalize well to new data.

2. **Visual Geometry Graph (VGG) -16**

- VGG is type of Convolution Neural Network (CNN) by Oxford University.
- It is the most widely used CNN architecture for image recognition tasks.
- This architecture consists of convolution layers followed by max pooling and fully connected layers.
- This project is based on the VGG 16 which consist of 13 convolution layers and 3 fully connected layers.

Architecture:

The VGG (Visual Geometry Group) architecture consists of a series of convolutional layers, followed by max pooling layers, and then fully connected layers at the end.
The architecture can be customized to have different numbers of layers, with the most famous versions being VGG16 and VGG19.

1. Input layer: The input to the network is a 224x224 RGB image.
2. Convolutional layers: There are 13 convolutional layers in VGG16, each with a 3x3 filter and a stride of 1. The number of filters

   in each layer varies from 64 to 512. All the convolutional layers use the ReLU activation function.

3. Max pooling layers: After each set of two convolutional layers, there is a max pooling layer with a 2x2 filter and a stride of 2.

   The purpose of these layers is to reduce the spatial size of the feature maps and prevent overfitting.

4. Fully connected layers: There are three fully connected layers at the end of the network. The first two have 4096 neurons each, and the third one has 1000 neurons, which corresponds to the number of classes in the ImageNet dataset. The last fully connected

   layer uses the softmax activation function to produce the final probability distribution over the classes.

In contrast to VGG16, the VGG19 design has 16 convolutional layers and 3 more convolutional layers before the first max pooling layer.
Modern performance has been attained by both VGG16 and VGG19 in a variety of image recognition tasks.

Key Feature:

The key features of the VGG (Visual Geometry Group) architecture are:

- Simplicity: The architecture is based on a simple and uniform pattern of repeated convolutional and pooling layers, which makes it easy to implement and understand.
- Small filters: The VGG's convolutional layers use tiny (3x3) filters, enabling the network to pick up fine-grained characteristics from the input images.
- Deep architecture: VGG can learn hierarchical representations of the input data thanks to its deep architecture, which contains several layers.
- Transferability: By pre-training the VGG architecture on huge image classification datasets like ImageNet, the learnt features can be applied to various computer vision problems with a small quantity of training data.

- wState-of-the-art performance: VGG has attained modern performance in a number of image recognition tasks, including semantic segmentation, object detection, and image classification.

  The main drawbacks of VGG are its high computational cost and memory requirements, which limit its practical applications on low-resource devices.

Components:

- Input layer: The input to the network is a 2D array of pixels representing an image.
- Convolutional layers: VGG has multiple convolutional layers, each of which applies a set of filters to the input to extract

  features. Each convolutional layer is followed by a non-linear activation function, typically the Rectified Linear Unit (ReLU).

- Pooling layers: After each set of two or three convolutional layers, VGG uses a max pooling layer, which downsamples the

  feature maps by taking the maximum value within a small window.

- Fully connected layers: VGG has multiple fully connected layers, which map the extracted features to the final output classes.

  The fully connected layers are typically followed by a softmax function to produce a probability distribution over the output

  classes.

- Dropout: To prevent overfitting, VGG uses a dropout layer after each fully connected layer, which randomly drops out a fraction

  of the neurons during training.

- Batch normalization: VGG also uses batch normalization after each convolutional layer and fully connected layer, which

  normalizes the activations to have zero mean and unit variance, making the network more stable and reducing the effect of

  internal covariate shift.

- Weight initialization: The weights in VGG are initialized using a Gaussian distribution with a mean of zero and a standard

  deviation of 0.01.

Overall, the VGG architecture is characterized by its deep and homogeneous structure, which allows it to learn hierarchical representations of the input data, and its use of small filters, which enables it to capture fine-grained features in the input images.

Implementation

I. Import TensorFlow: This line imports the TensorFlow library, which is used to build and train the VGG16 model.

II. Define the VGG16 function: This function takes a single input, num_classes, which specifies the number of classes in the

classification problem. The function returns a Keras Sequential model that implements the VGG16 architecture.

III. Define the convolutional layers: The first 13 layers in the VGG16 architecture are convolutional layers. These layers use 3x3

filters and ReLU activation. The first convolutional layer has 64 filters, and each subsequent layer doubles the number of filters

until there are 512 filters in the last layer.

IV. Define the max pooling layers: After every two convolutional layers, a max pooling layer is added to reduce the spatial

dimensions of the output. The pooling layers use a 2x2 window and a stride of 2.

V. Define the fully connected layers: After the convolutional layers, there are three fully connected layers with 4096 units each.

These layers use ReLU activation and dropout regularization with a rate of 0.5.

VI. Define the output layer: The final layer is a dense layer with num_classes units and softmax activation. This layer outputs the

predicted probabilities for each class.

VII. Compile the model: Before training the model, it needs to be compiled with a suitable loss function, optimizer, and metrics. This

can be done using the compile method of the Keras model.

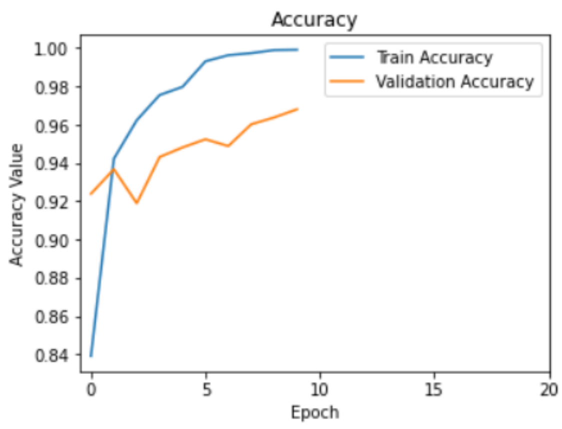
VIII. Train the model: Once the model is compiled, it can be trained on a dataset using the fit method. During training, the model updates its parameters to minimize the loss function and improve its accuracy on the training data.

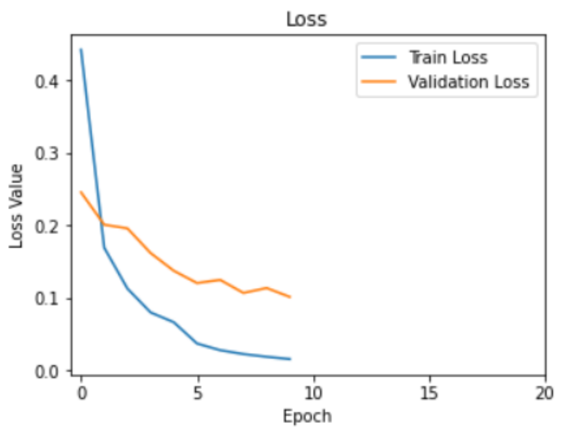
IX. Evaluate the model: After training, the model can be evaluated on a separate validation dataset to measure its performance. This

can be done using the evaluate method of the Keras model.

Model Training:

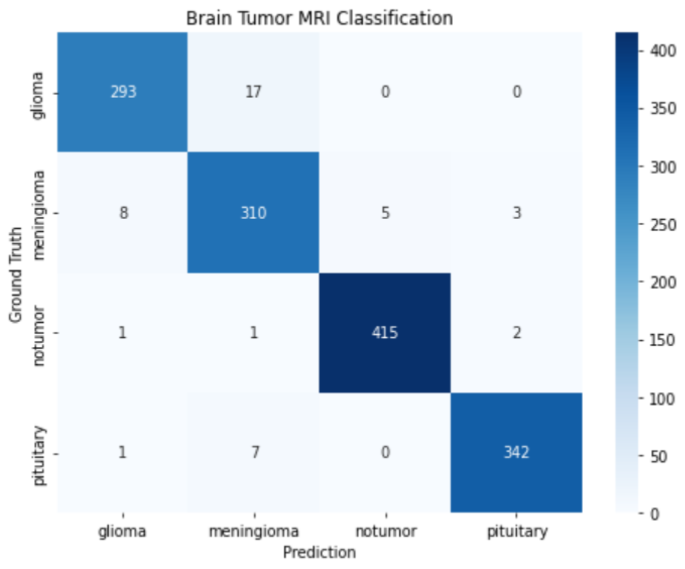
I have trained for 20 epochs and got this accuracy and loss graph.





Confusion Matrix:

- We generate it to visualize important predictive analytics like recall, accuracy, specificity, and precision.
- The diagonal elements present are the 'true predictions' that we were able to make.

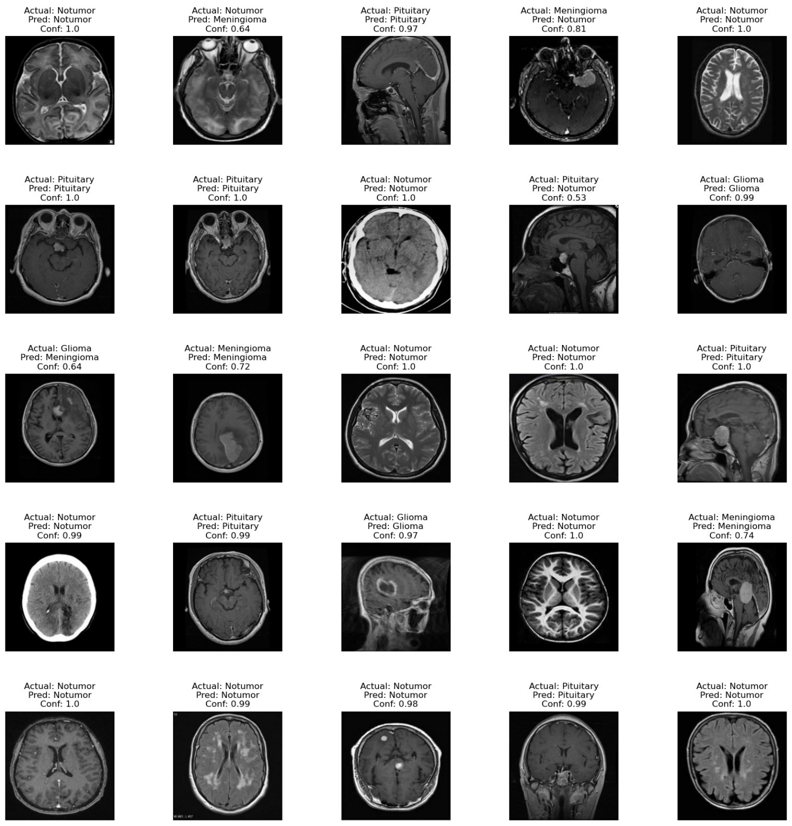


Ground Truth \ Prediction	glioma	meningioma	notumor	pituitary
glioma	293	17	0	0
meningioma	8	310	5	3
notumor	1	1	415	2
pituitary	1	7	0	342

Results:

Based on the information provided, it appears that the ResNet-50 model developed has achieved reasonably good accuracy rates.

- The training accuracy rate of 92.06% indicates that the model has learned to classify the data with a high degree of accuracy.
- The testing accuracy rate of 86.34% suggests that the model can generalize well to new, unseen data.
- The validation accuracy rate of 88.44% indicates that the model has not overfit the training data and is able to generalize well to new data.



Convolutional eXtreme Gradient Boosting:

- Convolutional Neural Networks, or CNNs, have grown in popularity in recent years for image classification and object identification applications, as many of you may already be aware. CNNs are made to recognize and extract characteristics from pictures automatically, but they can overfit and need a lot of computing power to train.

- XGBoost can help in this situation. A potent machine learning method called XGBoost makes predictions by using decision trees. It has been utilized in a number of applications, including fraud detection and customer churn prediction, and is renowned for its speed and accuracy.

Advantages

- ConvXGB combines the benefits of XGBoost with CNNs. ConvXGB's CNN component extracts features from pictures or other forms of input, and the XGBoost component uses these characteristics to produce predictions. ConvXGB, which combines these two techniques, provides a number of advantages over conventional CNN or XGBoost models, including increased accuracy, decreased overfitting, and quicker training periods.

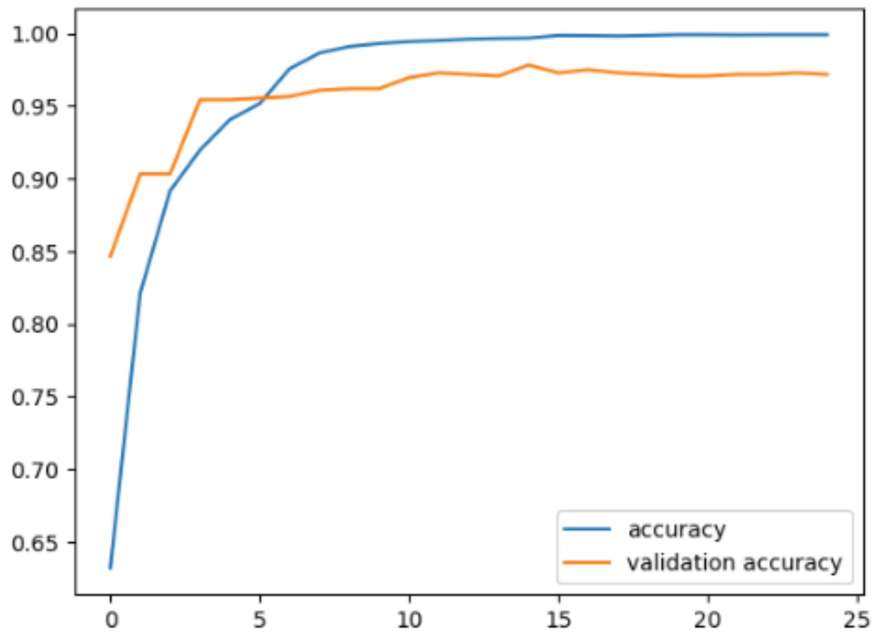
- Preprocessing the data for training is one computational technique that is relevant. For picture data, this can include scaling the photos to a standard size or using techniques for data augmentation to expand the training set. Another approach involves modifying a pre-trained CNN using transfer learning on a fresh dataset, which can cut training time and boost accuracy.

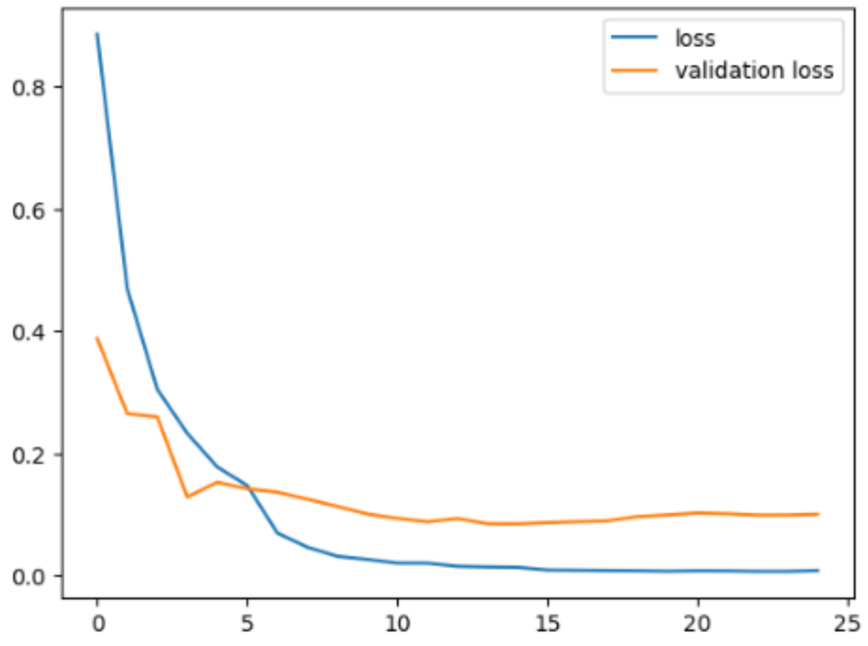
Model Implementation

Implementing ConvXGB includes several steps.

i. Preprocessing: Preprocessing is the process of getting data ready for training. For picture data, this can entail scaling the photos to a standard size or using techniques for data augmentation to expand the training set.
ii. Training the CNN: The CNN part of the model must first be trained before moving on to ConvXGB. In order to do this, the training data must be fed into the CNN, and the weights must be updated depending on the discrepancy between the expected and real labels.
iii. Extracting Features: The next stage is to utilize the CNN to extract features from the training data after it has been trained. This entails running the CNN with the training data and capturing each layer's output.
iv. Training the XGBoost model: The XGBoost part of the model must be trained as the last stage in the ConvXGB training process. This entails feeding the XGBoost algorithm with the characteristics that were retrieved from the CNN.
v. Evaluation: It's critical to assess the model's effectiveness on a validation set once it has been trained. Metrics like accuracy, precision, recall, and F1 score can be used to achieve this.

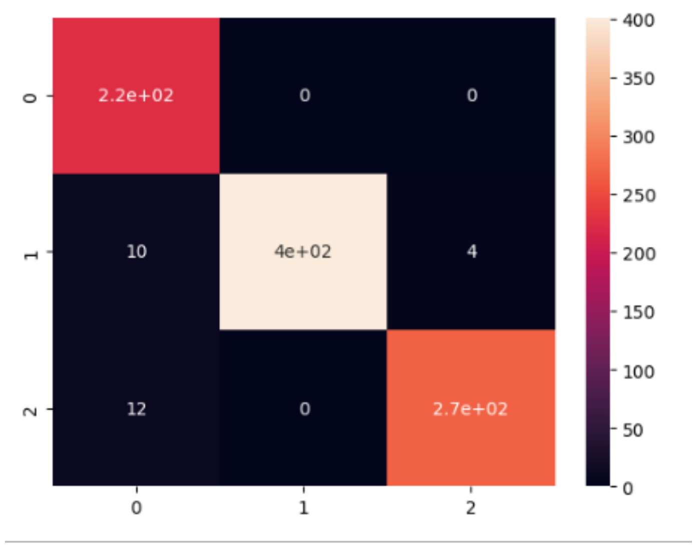
Accuracy and Loss



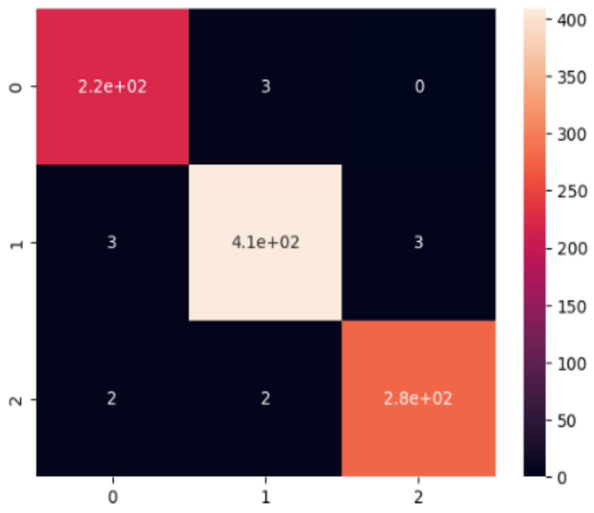


Confusion Matrix

CNN



XGBoost



Results

Based on the information provided, it appears that the ConvXGB model developed has achieved good accuracy rates.

- Training accuracy rate of 99.30 %.
- Testing accuracy rate of 97.17 %

The validation accuracy rate of 98.58 %.