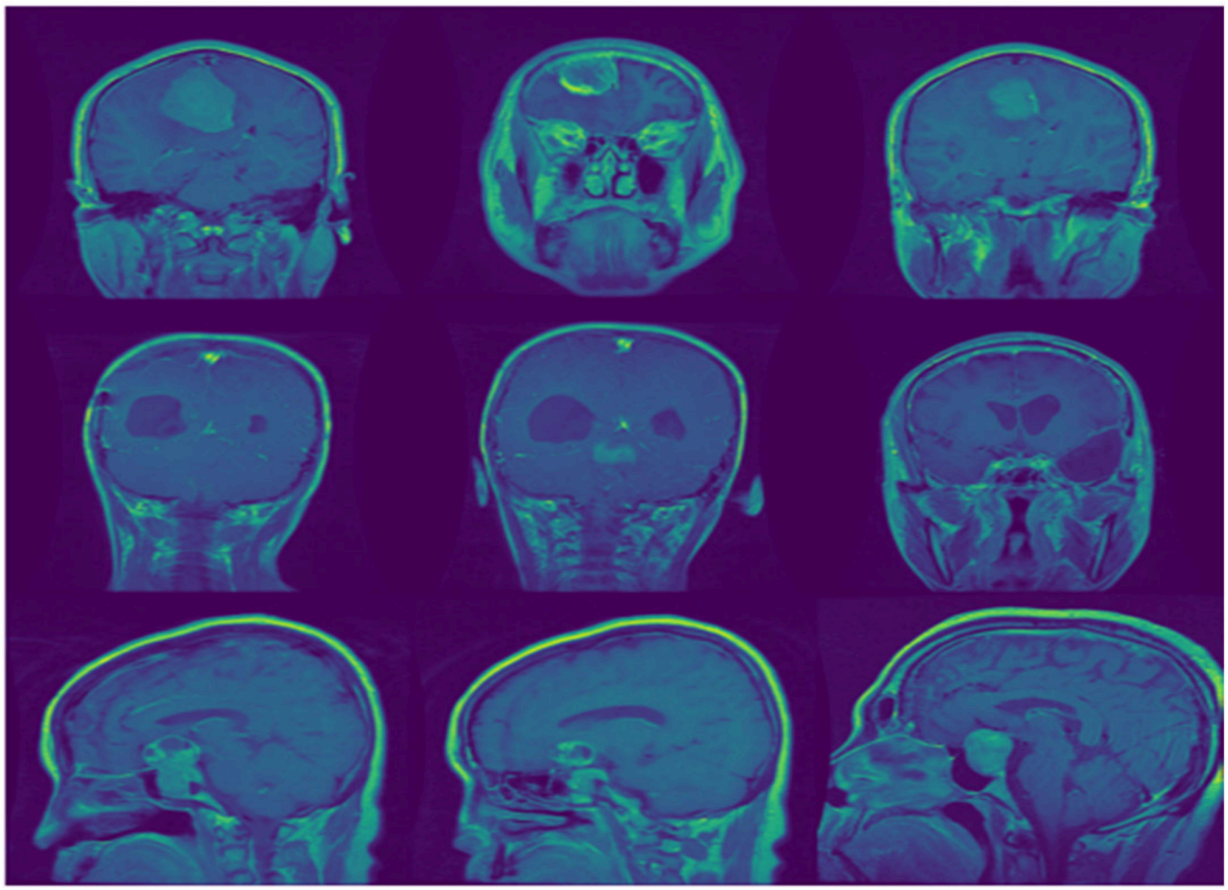
Sabra Mabrouk

Brain Tumor Classification



Introduction

A brain tumor is an abnormal growth of cells within the brain. Tumors can either be benign or malignant. Malignant brain tumors are more dangerous and aggressive due to their potential to invade surrounding brain tissue and spread to other parts of the body. Brain tumors can develop in any area of the brain, and their effects can vary depending on their type, location, and size. Some common types of brain tumors include Gliomas, Meningiomas, and Pituitary Adenomas. Brain tumor classification refers to the process of categorizing different types of brain tumors based on medical imaging data (such as MRI scans) to assist in diagnosis, treatment planning, and prognostication. The classification of

brain tumors is a critical task in the medical field, as the appropriate treatment strategy often depends on accurately identifying the type, size, and location of the tumor

Problem statement

The goal of a brain tumor classification project is to develop an automated system capable of accurately classifying brain tumor types using medical imaging data (e.g., MRI scans). Brain tumors, which can be either benign or malignant, present a significant challenge for healthcare professionals due to the complexity of the human brain and the variety of tumor types. Early and accurate diagnosis is crucial to improving patient outcomes, and computational tools based on artificial intelligence (AI) can assist radiologists by providing decision support for diagnosis and treatment planning.

Objective

To develop a deep learning-based model that can automatically classify brain tumors into multiple categories (such as glioma, meningioma, and pituitary) using MRI scan images to assist in the early detection, diagnosis, and treatment planning for patients.

Motivation

Brain tumors, both malignant and benign, pose significant health risks, and timely detection is crucial for effective treatment. However, manual interpretation of medical images by radiologists is time-consuming and prone to errors, particularly when the tumor is located in a complex or hard-to-reach part of the brain. Moreover, the increasing number of MRI scans being produced in healthcare settings necessitates an automated solution to assist clinicians in diagnosing brain tumors accurately and efficiently.

Dataset

Source: ***https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset***

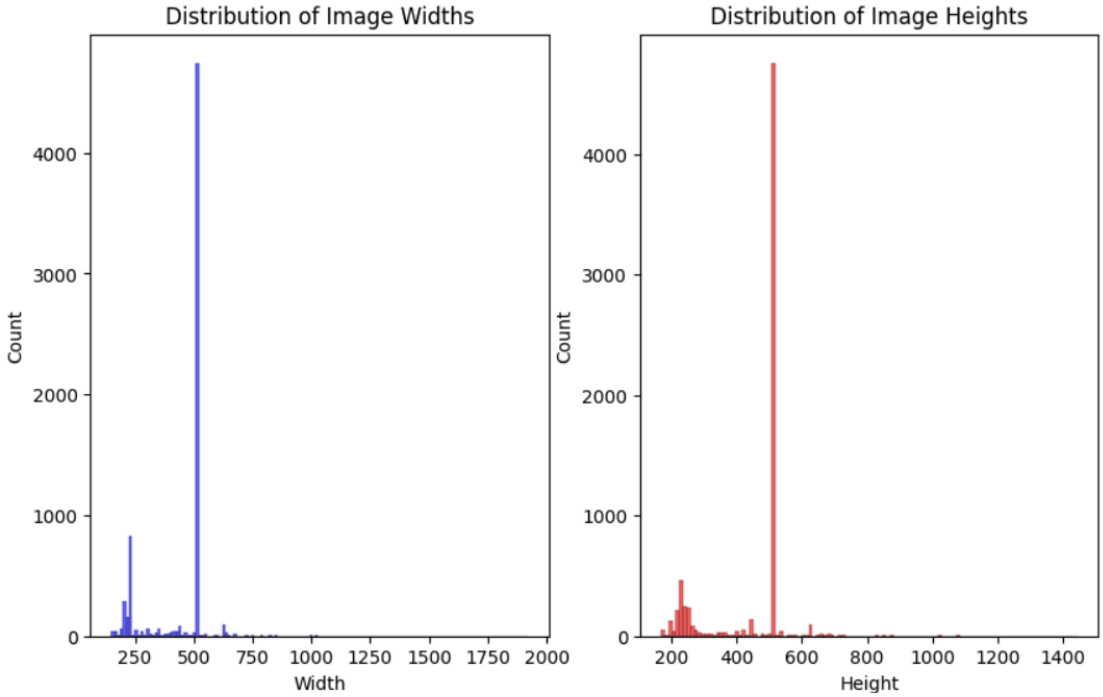
The dataset contains 7023 human brain MRI images which are classified into 4 classes: glioma, meningioma, pituitary, and no tumor. The dataset is divided as:

- Training dataset: 5712 images
- Testing dataset: 1311 images

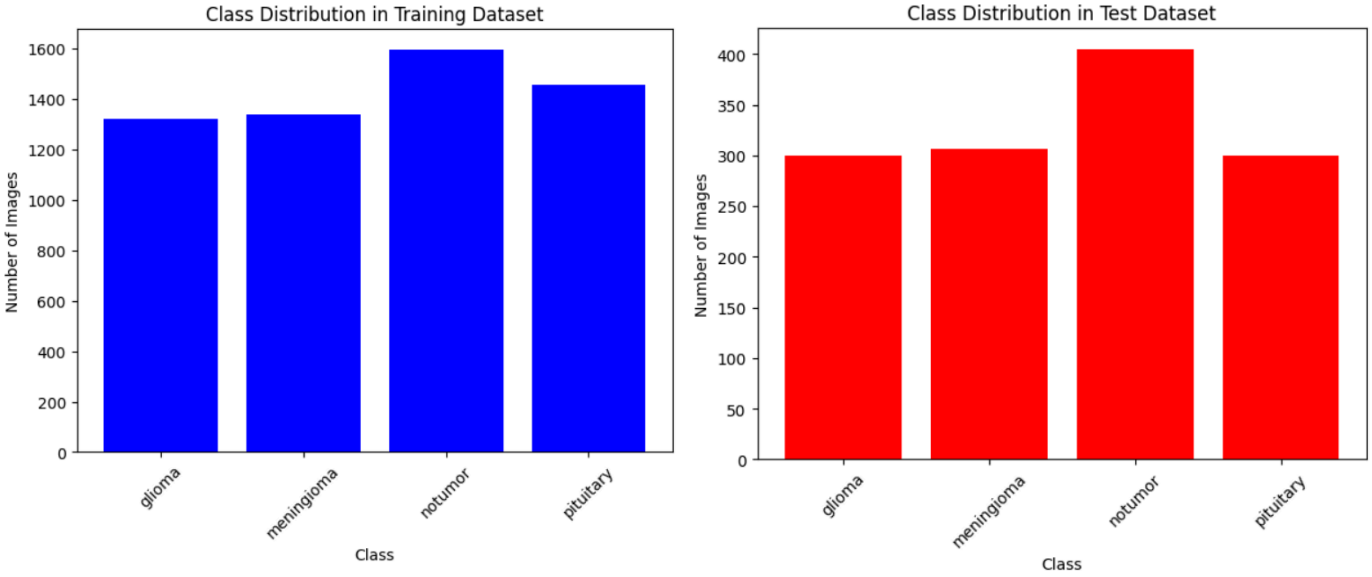
Data wrangling and EDA

We first checked for corrupt or unusually small images, and removed any image smaller than 50x50 pixels. No corrupted images were found.

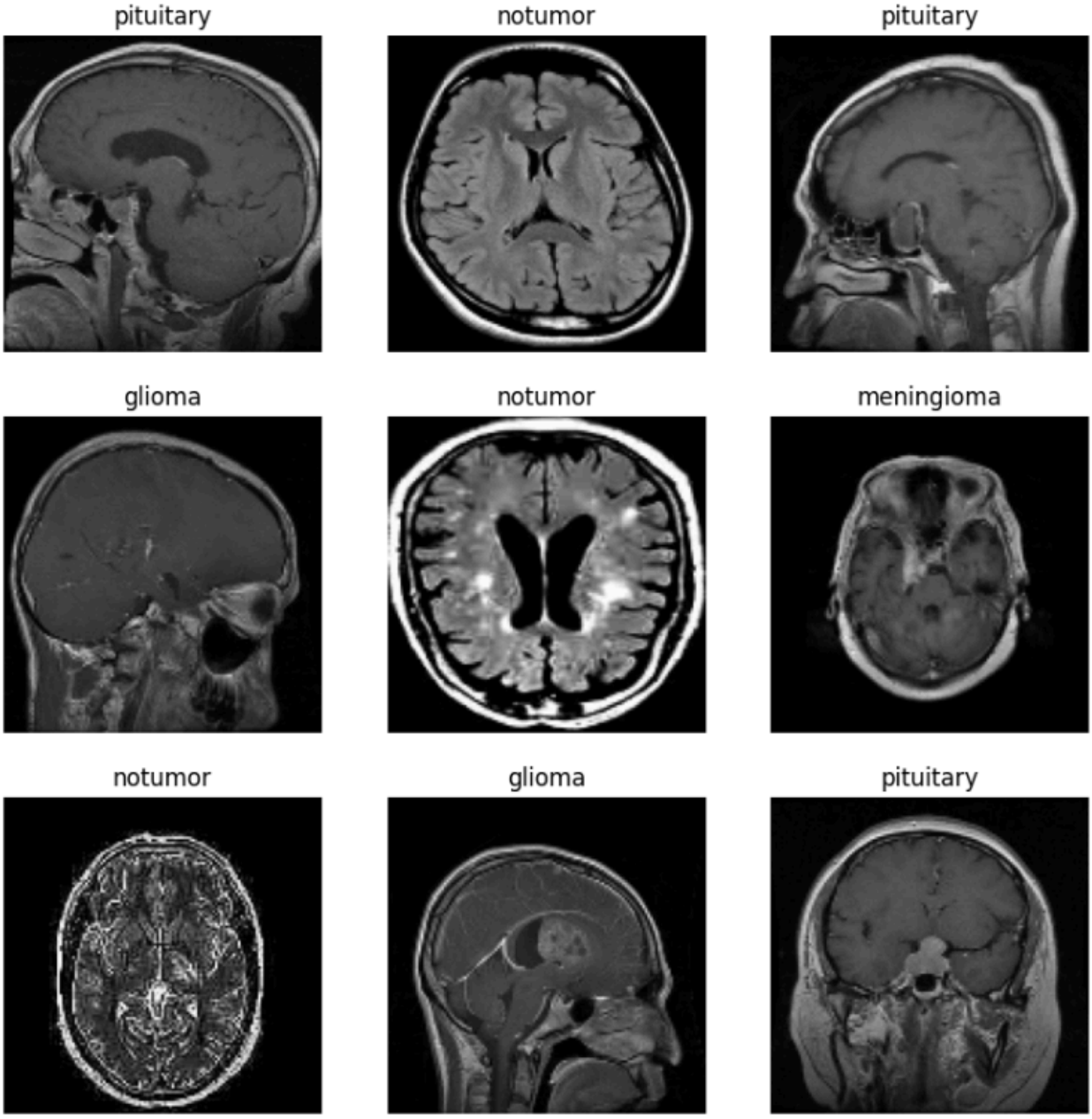
We then visualize the distribution of image dimensions, images in the dataset have different dimensions.



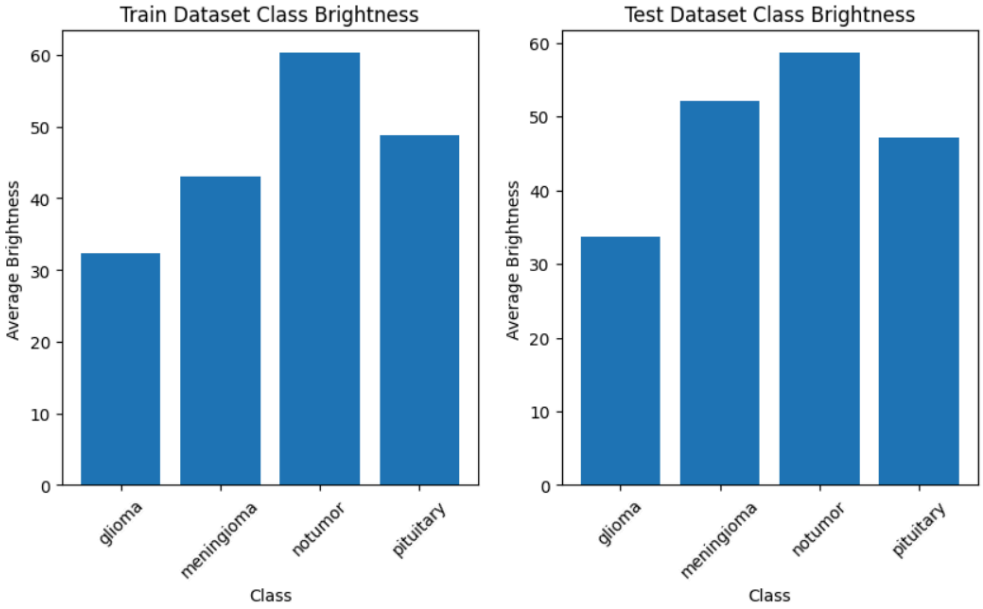
We checked the class distribution in the training and test sets



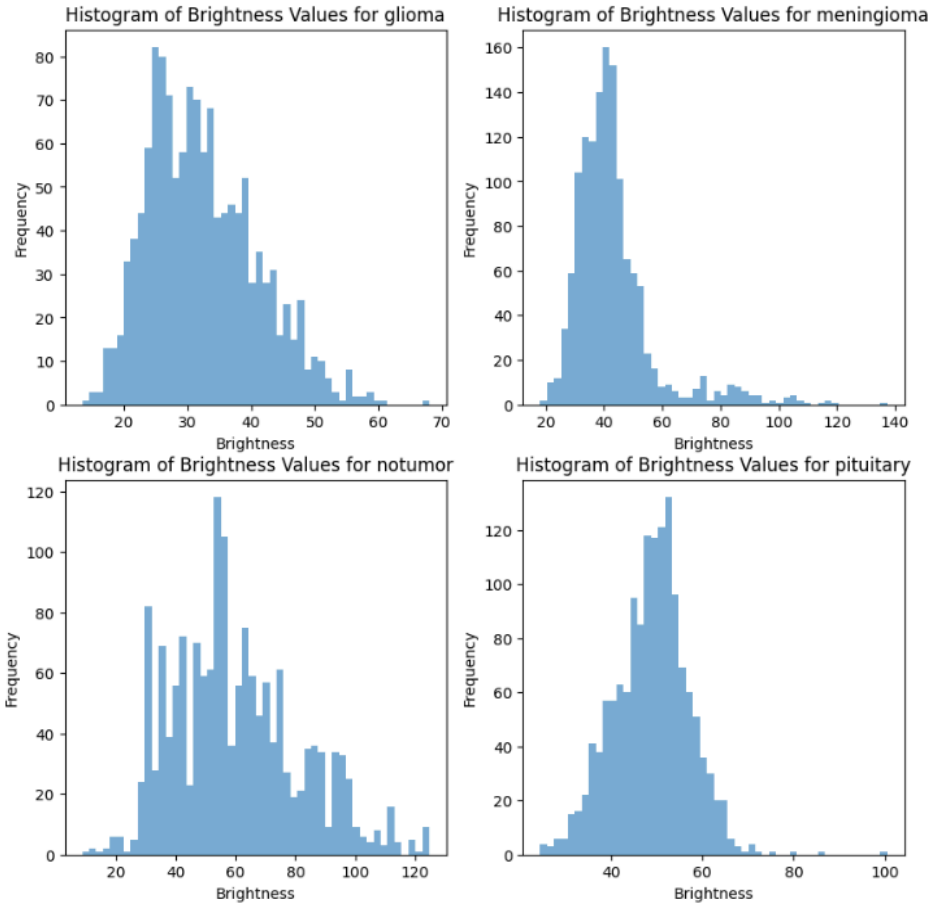
The positive classes have a similar number of images, the No-Tumor class has the largest number of images in both the training and testing set.
We propose to visualize a sample of images from different classes in the dataset.



To calculate brightness in images, we converted the image to grayscale and computed the mean brightness per class.



We then plot the histograms of brightness values for each class to see the distribution of brightness within each class



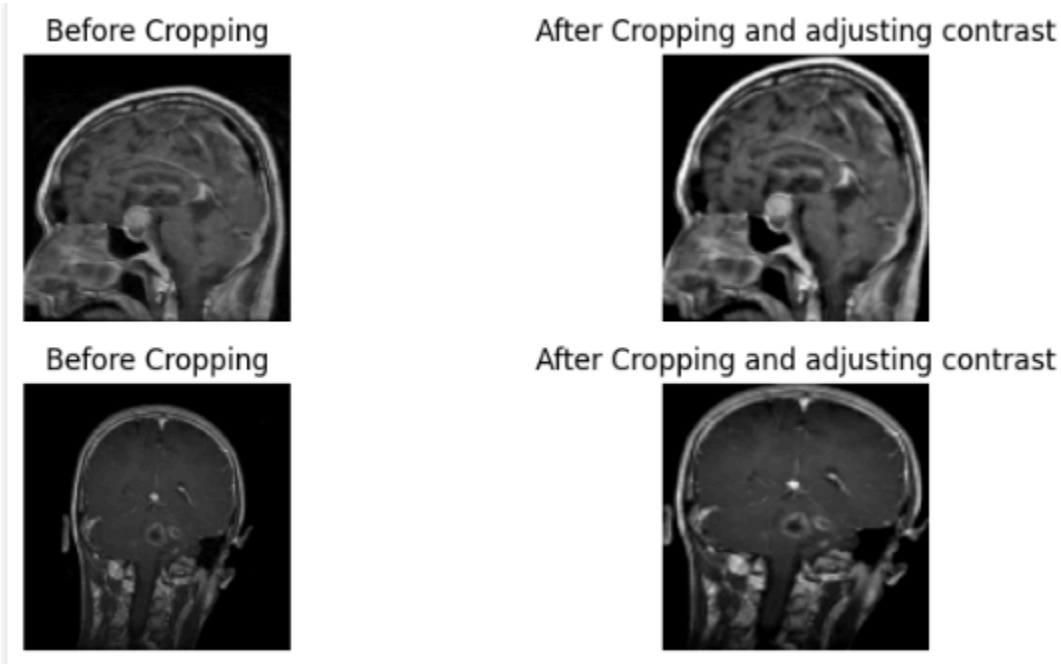
Images in the meningioma and pituitary classes exhibit low contrast. Poor contrast makes the edges and boundaries of structures (brain tissues or tumors) less defined, which can affect feature extraction. Image should be preprocessed to enhance the contrast.

Pre-processing and modeling

We stored the data from the previous step in a TFRecordDataset format. The first step in the data pipeline was to parse and preprocess the data. We created a function to parse the TFRecord files, which decodes the serialized image data. Each image was then resized to a consistent 150x150 pixel resolution to standardize the input size for the neural network. Additionally, the labels were one-hot encoded to represent the multi-class classification problem. After preprocessing the images and labels, we batched the data into batches of size 32.

Image transformation

We applied a 5x5 gaussian filter to images to remove noise and cropped them to help focus on the brain area by reducing the background. Next we adjusted the contrast of images to improve their quality. In the figure below, we show examples of images before and after preprocessing.



Model Definition

We have defined a Convolutional Neural Network (CNN) structured as follows:

1. Input Layer: The model is initialized with an input shape of (150, 150, 3). This corresponds to images of size 150x150 pixels with 3 color channels (RGB).
2. First Convolutional Block: this block is composed of
    - Convolutional layer that applies 256 filters of size 3x3 to the input images, the ReLU (Rectified Linear Unit) activation function introduces non-linearity to the model, allowing it to learn complex patterns. L2 regularization is applied to prevent overfitting by penalizing large weights
    - MaxPooling Layer: This layer performs max pooling with a pool size of 2x2. It reduces the spatial dimensions (height and width) of the feature maps, helping the model to become more invariant to small translations in the input. Pooling also reduces computational complexity and helps to prevent overfitting.
    - Dropout Layer: This layer randomly drops 30% of neurons during training, which helps prevent overfitting by encouraging the model to generalize better.
3. Second Convolutional Block: it contains
    - A second convolutional layer applies 128 filters of size 5x5. The number of filters is doubled compared to the first convolutional layer to capture more complex patterns. The ReLU activation again introduces non-linearity and L2 regularization to prevent overfitting.
    - Max pooling and dropout are also applied to reduce dimensionality and prevent overfitting
4. Third Convolutional Block: The third convolutional block has
    - A convolutional layer that applies 64 filters of size 7x7, further increasing the depth of the network to capture even more complex features. The ReLU activation is used to retain non-linearity.
    - Max pooling and dropout are applied again
5. Flatten Layer: After the convolutional and pooling layers, the output is a multi-dimensional tensor. The Flatten layer reshapes this tensor into a 1D vector so that it can be passed to fully connected layers (Dense layers).

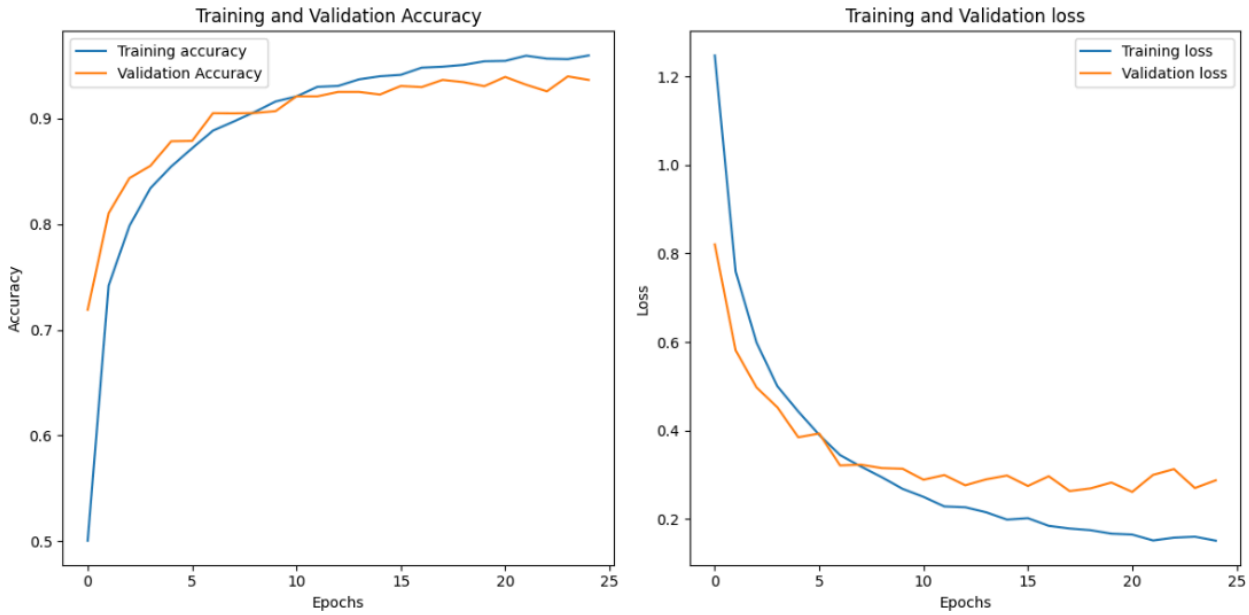
6. Fully Connected Dense Layer: This is a fully connected (dense) layer with 128 neurons and ReLU activation. It allows the model to combine the learned features from the convolutional layers and make more abstract predictions.

7. Another dropout with a rate of 50%, helping prevent overfitting by randomly dropping half of the neurons in this layer during training

8. Output Layer: The output layer has 4 neurons, one for each class. The softmax activation function ensures that the model outputs a probability distribution over the 4 classes. The class with the highest probability is the model's predicted class for the input image.

The optimizer Adam is used to adjust the weights of the model during training to minimize the loss function.

A 5-fold cross-validation is applied while training, we plot the training and validation accuracy and the training and validation loss over epochs.



After training, we evaluated the model on the test set, the accuracy is given by:

```
Test Loss: 0.2509309947490692
Test Accuracy: 0.9320312738418579
```

Hyperparameter Tuning

In order to enhance our model performance, we propose to tune our model using Keras Tuner, while applying cross validation. The tuned parameters are:

- The number of filters in the 3 convolutional blocks
- The dimensionality of the output space in the dense layer
- The learning rate of the optimizer “adam”

The best parameters are: {'filters_1': 64, 'filters_2': 64, 'filters_3': 512, 'dense_units': 192, 'learning_rate': 0.0006759502243542753}

The results of the Tuning are given in the table below.

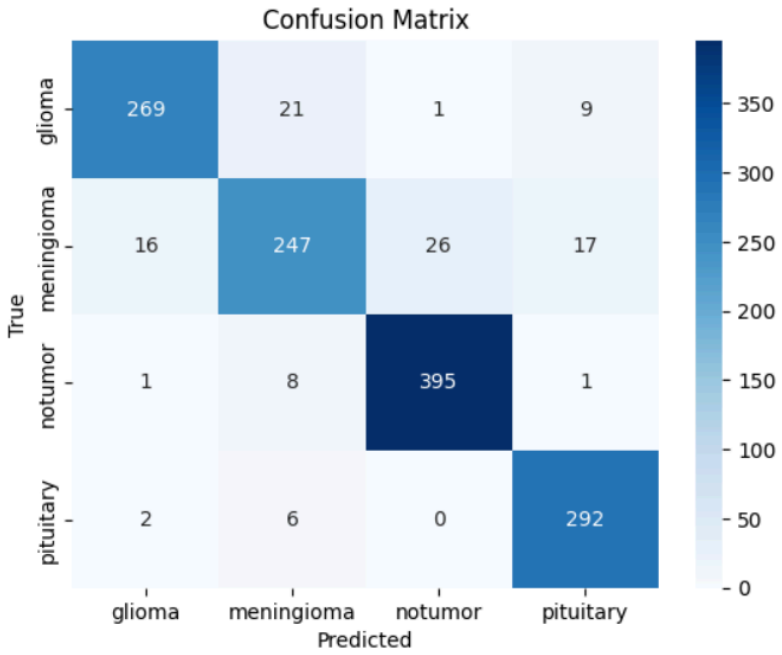
Mean validation accuracy cross folds	Standard deviation of validation accuracy across folds
0.96	0.024

The test accuracy of the best model is equal to 0.915

Evaluation of the best model

In the context of brain tumor classification, we need to ensure that the model is both accurate and reliable, particularly in detecting tumors and minimizing false negatives. To thoroughly assess our model, we computed the precision, recall, F1 score, the roc-auc score and the confusion matrix, results are given as follows.

```
Precision : 0.9153234403784606
Recall : 0.9131245461147421
F1-Score : 0.9134548014117674
AUC-ROC : 0.9855789551328838
```



The proposed model achieved a high accuracy rate (above 90%), it is performing with very high precision and recall rates giving an AUC of 0.98.

Conclusion

In this project, we are interested in classifying brain Tumor MRI. The dataset provides human brain MRI and their labels, and we aim to fit a model capable of accurately classifying these images into 4 classes. The images were filtered and cropped before being fed to a CNN model. This model is composed of three convolutional layers with progressively increasing filter sizes and depths, followed by max pooling layers to reduce spatial dimensions and retain important features. The model incorporates Dropout layers to help mitigate overfitting, especially during training. The L2 regularization applied to the convolutional layers further helps prevent overfitting by penalizing large weights. After the convolutional blocks, the model flattens the features into a one-dimensional vector and passes it through a dense layer with ReLU activation. Finally, a softmax output layer is used for classification. Some hyperparameters were tuned and the best model succeeded to give a high accuracy rate of 91% and a recall rate of 91%, an indicator of reliability in this context. In future work, we aim to improve the model performance by :

- Using data augmentation techniques to generate more data which can help the model generalize better and improve its ability to detect positive cases.
- Increasing model complexity, either by adding more layers or more neurons in the dense layers which can help the model learn better feature representations.
- Tuning more hyperparameters like batch size and dropout rate to explore different configurations and select the one that works best for improving recall.