MRI Brain Tumor Detection using Convolutional Deep Learning Methods

Nassim Ali-Chaouche, Kartik Mukkavilli, Quan Gu, Nail Enikeev

Department of Applied Data Science, San Jose State University

DATA 255: Deep Learning Technologies

Simon Shim, PhD

Abstract

Brain tumors, a severe affliction that affects both children and adults, present significant diagnostic challenges due to their complex nature and varied presentation in MRI scans. Given the rising incidence of brain tumors and their high mortality rate, accurate and timely diagnosis is crucial. This project aims to harness the power of Deep Learning to create an automated system for detecting and classifying brain tumors from MRI scans, potentially offering a faster and more accurate diagnostic tool for clinicians worldwide, especially in regions lacking skilled radiologists. A fine-tuned InceptionV3 model yielded the best results. The model struggles in the recall of glioma tumors, but demonstrates good performance when considering every class.

1. Introduction

In the ever-evolving landscape of medical diagnostics, the utilization of Magnetic Resonance Imaging (MRI) for brain tumor detection stands as a critical juncture between technology and healthcare. The accurate identification and classification of brain tumors via MRI scans are paramount, not only for effective patient treatment but also for advancing our understanding of neurological disorders. As deep learning technologies have progressively infiltrated the domain of medical imaging, they have opened new avenues for more precise and efficient diagnostic methods. This study focuses on dissecting the myriad of deep learning models that have been applied in recent years for brain tumor detection in MRI scans. By examining the nuances of these models, from the foundational convolutional neural networks (CNNs) to the emerging vision transformers and hybrid systems, this review aims to distill the essence of their advancements and evaluate their efficacy. The goal is to ascertain which model, or combination thereof, demonstrates the most promising results in terms of accuracy, reliability, and practicality in clinical settings, thereby contributing to the enhancement of diagnostic procedures in the field of neuro-oncology.

2. Literature Review

The realm of medical imaging has witnessed a pivotal shift with the advent of deep learning methodologies, particularly in the detection and classification of brain tumors via MRI scans. This literature review encapsulates the essence of cutting-edge research in this field, scrutinizing a spectrum of methodologies ranging from established convolutional neural networks (CNNs) to burgeoning technologies like vision transformers and multifaceted hybrid models.

2.1 Convolutional Neural Networks (CNNs)

The application of CNNs in the realm of brain tumor classification has garnered significant scholarly attention. Saeedi et al. (2023) ventured into this domain by deploying convolutional deep learning techniques in conjunction with select machine learning methods, thereby reinforcing the efficacy of CNNs in brain tumor detection. Complementing this, the research conducted by Vankdothu, Hameed, and Fatima (2022) delves into a novel approach integrating CNN with Long Short-Term Memory networks, thus underscoring the synergy between these technologies in augmenting classification precision.

2.2 Hybrid and Advanced Models

The landscape of hybrid models presents a noteworthy progression. The study by Cinar, Ozcan, and Kaya (2022) introduced an innovative hybrid DenseNet121-UNET model, aimed at brain tumor segmentation. This model exemplifies the integration of DenseNet architecture with the UNET framework, yielding enhanced segmentation accuracy in MRI scans. Similarly, Anand et al. (2023) have made strides in this field by proposing a weighted average ensemble deep learning model, thereby accentuating the utility of ensemble methodologies in refining diagnostic precision.

2.3 Vision Transformers

The emergence of vision transformers in medical imaging signifies a groundbreaking development. Asiri et al. (2023) embarked on a comparative analysis of pre-trained models, focusing on advancing brain tumor classification through the application of fine-tuned vision transformers. Their research posits that vision transformers hold the potential to surpass traditional CNNs under specific scenarios. In a related vein, Asiri et al. (2023) investigated the implementation of a fine-tuned vision transformer, demonstrating its proficiency in the accurate and efficient detection of brain tumors, thereby highlighting the transformative impact of this technology in medical imaging.

2.4 Inception Modules and ResNet

The integration of inception modules into brain tumor segmentation processes has been spotlighted by Cahall et al. (2019), who elucidated that these modules significantly enhance segmentation capabilities. Further, Rathore et al. (2023) explored the application of ResNet-50 in tandem with image processing tools, contributing to the broader discourse on the effective application of deep learning in medical diagnostic procedures.

2.5 Refined Approaches in CNNs

AlTahhan et al. (2023) presented a refined methodology for the automatic classification of brain tumors, utilizing a hybrid convolutional neural network framework. Their findings underscore the dynamic evolution of CNNs, particularly in their adaptation and integration with other advanced technologies to improve performance in MRI scan analysis.

The current corpus of research in MRI-based brain tumor detection and classification is characterized by its diversity and the unique contributions of each study to the efficacy of different models. The transition from conventional CNNs to advanced vision transformers and hybrid models mirrors the rapid evolution of machine learning techniques within the sphere of medical imaging. Our project, aimed at identifying the best-performing model in this context, necessitates an evaluation that extends beyond mere accuracy, encompassing considerations of practicality and adaptability in clinical environments.

3. Selection of pre-trained models

3.1 VGG-19

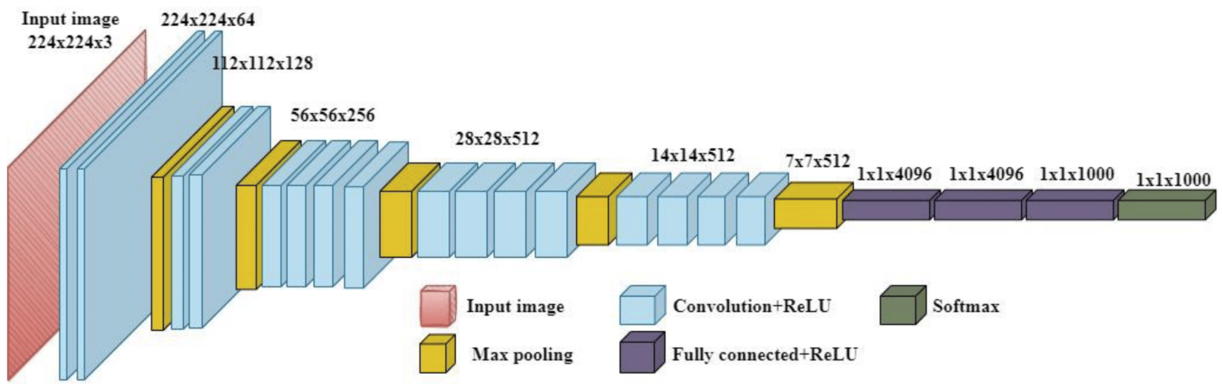
One of the high-performance models for image classification problems is VGG-19 - a leveled-up deeper version of VGG-16 architecture. According to the “VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease” research paper (Nguyen, 2022), the fine-tuned VGG-19 model performed at 99.72% accuracy in crop disease classification, which is the highest value among other fine-tuned experimented models, such as GoogleNet, ResNet-50, and AlexNet. The model contains 143 million parameters.

The total number of layers is 19, compared to VGG-16, which has 16 layers. There are 16 convolutional and 3 fully connected layers. All the layers have weights. Convolutional layers extract features and learn the relationships between them. The reason behind using so many convolutional layers is that a cascade of them significantly increases the depth of feature extraction and results in catching even more complex relationships. Each convolutional layer has a 3x3 filter size - the smallest size that still can capture features. Between the convolutional layers, there are 2x2 size max-pooling layers with a stride of 2. The max-pooling layers reduce the spatial dimensions of the feature maps and play a significant role in preventing the network from overfitting (Stephen, 2023). As an input, the network takes fixed-size 224x224 images with 3 RGB channels. A cascade of convolutional layers starts with increasing the number of channels to 64 and keeping the same 224x224 image size. Then, each following max-pooling layer doubles the number of channels and reduces the spatial dimensions of the feature map. The cascade of convolutional layers is followed by 3 fully connected and one softmax layer. The fully connected layers increase the channel number to 4096, 4096, and 1000 for each layer respectively. As an output, the softmax layer provides a distribution of probabilities of the input for each class. The VGG-19 neural network architecture is shown in Figure 1.

Each convolutional and fully connected layer has a Rectified Linear Unit (ReLU) activation function. The ReLU provides positive output for positive input, and zero value for negative input. The ReLU introduces non-linearity to the model, which is crucial for capturing complex relationships. Compared to other activation functions, such as Tanh and Sigmoid, the ReLU is computationally efficient due to its simplicity. Thus, the ReLU decreases the chances of the gradient vanishing, which is a typical issue in deep neural networks. However, there is a disadvantage to using ReLU, known as “dying” ReLU. The functions assign zero value output to any negative input, which shuts down neurons that still might be valuable to the network. To solve this problem it is suggested to use the Leaky ReLU activation function, which assigns a small non-zero value for negative input. However, in the current project, experiments with base architecture layers are not considered since the model is fine-tuned on the actual data and extended by custom additional layers.

Figure 1

VGG-19 architecture



Note. *Image credit: https://doi.org/10.3390/agriengineering4040056*

To handle multi-class classification in VGG-19 sparse categorical cross entropy loss function is used. Sparse categorical cross entropy is the typical choice in image classification tasks as it defines one class out of all classes. Also, the loss function directly accepts integer values assigned to each class, without the need to preprocess the data with one-hot encoding. In addition, this reduces computation time, especially when dealing with a large number of classes. Since VGG-19 is a deep convolutional network reducing the computational complexity and memory usage is beneficial.

Applying 19 layers in comparison to 16 in VGG-16 is more computationally expensive, but on the other hand, the model does more robust feature extraction and, thus, increases the model performance. The model optimizer is Adaptive Moment Estimation (Adam), which holds adaptive separate learning rates for each parameter, providing more precise weight updates.

The softmax layer accepts logits and outputs a probability distribution, which is directly used to do multi-classification over four classes.

3.2 ResNet-50

The second common choice for multi-class image classification is ResNet-50 - an improved version of ResNet architecture. It has a unique structure of residual blocks and skip connections, which provides around 24 million parameters.

Deep residual learning is the main concept of the architecture, which was developed to overcome the vanishing gradient problem in deep learning. The ResNet-50 has 50 layers that are: convolutional, max-pooling, and fully connected layers. The network starts with a single 3x3 convolutional layer with a stride of 2, followed by a max-pooling layer with a depth of 64. Max-pooling applies downsampling and reduces spatial dimensions of the feature map for deeper feature extraction. As an input, the network takes fixed-size 224x224 images with 3 RGB channels. Four residual blocks are used to reduce complexity and training time. The design of each block is a 3-layer bottleneck: 1x1 convolutional layer reducing the dimension, 3x3 convolutional layer, and 1x1 convolutional layer, which expands the dimension. ResNet has four stages utilizing several residual blocks. Three, four, six, and 3 blocks for each stage respectively. The amount of filters gradually increases from 64, through 128, 256, 512, 1024, to 2048. Each residual block for each stage includes skip connections, that bypass one or more layers. The skip connections outputs are added to the outputs of stacked layers. Such an approach allows gradients to pass straight through skip connections during the backpropagation and support training through all convolutional layers avoiding the vanishing gradient problem. The average pooling layer follows the chain of residual blocks. A fully connected layer finalizes the network, with the size of four - the number of classes. The model architecture is shown in Figure 2.

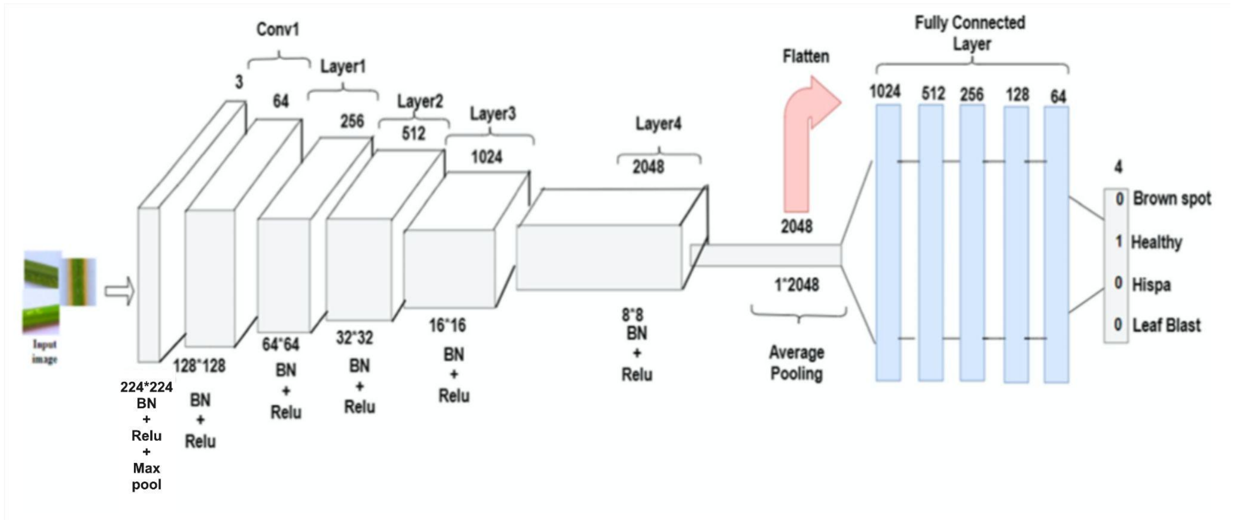
To handle multi-class classification in ResNet-50 sparse categorical crossentropy loss function is used. Sparse categorical cross entropy is the typical choice in image classification tasks as it defines one class out of all classes. Also, the loss function directly accepts integer values assigned to each class, without the need to preprocess the data with one-hot encoding. In addition, this reduces computation time, especially when dealing with a large number of classes. Since ResNet-50 is a deep convolutional network reducing the computational complexity and memory usage is beneficial.

After each convolutional layer batch normalization is applied. This helps to prevent the gradient from vanishing or exploding, maintaining mean and variance as inputs for activation functions, per batch. As an activation function Rectified Linear Unit (ReLU) is utilized. The ReLU provides positive output for positive input, and zero value for negative input. The ReLU introduces non-linearity to the model, which is crucial for capturing complex relationships. Compared to other activation functions, such as Tanh and Sigmoid, the ReLU is computationally efficient due to its simplicity. Thus, the ReLU decreases the chances of the gradient vanishing, which is a typical issue in deep neural networks. The model optimizer is Adaptive Moment Estimation (Adam), which holds adaptive separate learning rates for each parameter, providing more precise weight updates. The softmax activation function outputs probabilities distribution over four classes.

ResNet-50 has a relatively complex structure. Although the model is optimized for efficient computation and memory usage, it still requires a significant amount of resources.

Figure 2

ResNet-50 architecture



Note. *Image credit: https://doi.org/10.1007/s00521-022-07793-2*

3.3 Inception V3

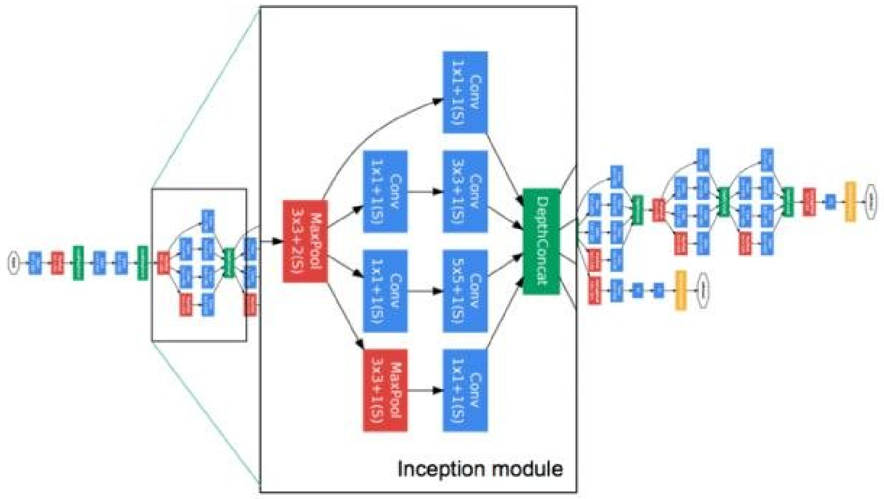
Inception V3 - convolutional neural networks, the third improved version of the Inception models. It was designed with two points in mind - accuracy and efficiency. It has around 24 million parameters. The architecture is highly complex. The main concept of the neural network is the idea of using inception modules - building blocks containing multiple types of filters, that the model is allowed to choose. Each inception module offers filters with different sizes, such as 1x1, 3x3, 5x5. Thus, the model can capture even more features utilizing a variety of spatial dimensions. The inception module is shown in Figure 3.

The first 1x1 convolutional layer reduces the spatial dimensions of the input to pass them for larger-size convolutions. The module also utilizes max-pooling layers. Then, after inputs are passed to different filters, their outputs are concatenated considering the channel dimensions to pass to the next layer. Thus, the network processes the data at a variety of dimensions simultaneously. The architecture consists of stacked inception modules, that allow to capture features even deeper. Also, while maintaining depth and complexity, by focusing on the important features at every scale the risk of overfitting is reduced. Additionally, to overcome overfitting, label smoothing is applied to the loss function. Inception V3 breaks down larger convolutional filters into smaller ones. Module after module the number of outputs increases, such an approach allows the network to capture more complex features at each layer. The model utilizes asymmetric convolutions, which means splitting 3x3 convolution into 1x3 and 3x1 sizes. The vanishing gradient is mitigated by auxiliary classifiers, that add gradients at lower layers while training. Each input is normalized by batch normalization to have zero mean and variance. Batch normalization is applied after convolutional layers before the activation function. This approach makes convergence faster, which means higher training speed.

As an activation function Rectified Linear Unit (ReLU) is utilized. The ReLU provides positive output for positive input, and zero value for negative input. The ReLU introduces non-linearity to the model, which is crucial for capturing complex relationships. Compared to other activation functions, such as Tanh and Sigmoid, the ReLU is computationally efficient due to its simplicity. Thus, the ReLU decreases the chances of the gradient vanishing, which is a typical issue in deep neural networks. To handle multi-class classification in Inception V3 sparse categorical cross entropy loss function is used. Sparse categorical cross entropy is the typical choice in image classification tasks as it defines one class out of all classes. Also, the loss function directly accepts integer values assigned to each class, without the need to preprocess the data with one-hot encoding. In addition, this reduces computation time, especially when dealing with a large number of classes. The model optimizer is Adaptive Moment Estimation (Adam), which holds adaptive separate learning rates for each parameter, providing more precise weight updates. The softmax activation function outputs probabilities distribution over four classes. The model architecture is shown in Figure 4.

Figure 3

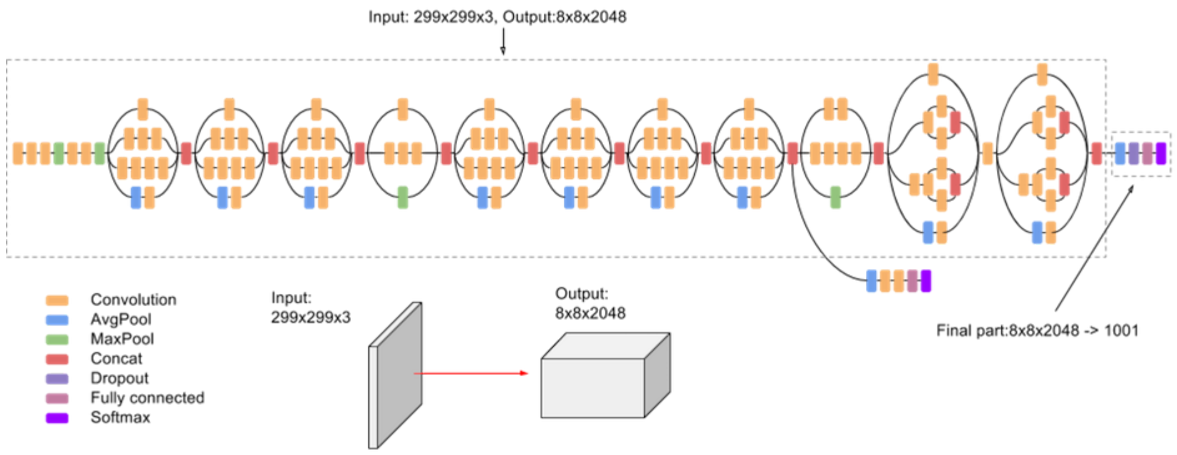
Inception module



Note. *Image credit: https://doi.org/10.35940/ijeat.B3226.129219*

Figure 4

Inception V3 architecture



Note. *Image credit: https://doi.org/10.14569/IJACSA.2019.0100107*

3.4 DenseNet121

DenseNet121 is a Dense Convolutional Network, part of the DenseNet models family. The main concept is having dense feed-forward connections, thus, each layer is linked to every other layer. The model has 121 layers in total, including convolutional, fully connected, and max-pooling layers. The network holds 7 million parameters.

The model consists of dense blocks, that have layers inside. Each layer has an extra input from all previous layers. These inputs are applied to the feature map of the current layer, which in its order, is connected to all subsequent layers. The amount of data that every layer contributes to the network state is controlled by the Growth Rate - a hyperparameter with value “k”, meaning each layer adds “k” feature maps to the total state of learning of the network. The model also utilizes bottleneck layers, which are convolutional layers with a 1x1 filter, followed by 3x3 convolution to reduce the spatial dimensions of the feature map at the next input, thus, reducing computational complexity. There are also transitional layers between dense blocks. Each transitional layer consists of batch normalization, a convolutional layer with filter 1x1, and an average pooling layer. Transitional layers control the feature map dimensions.

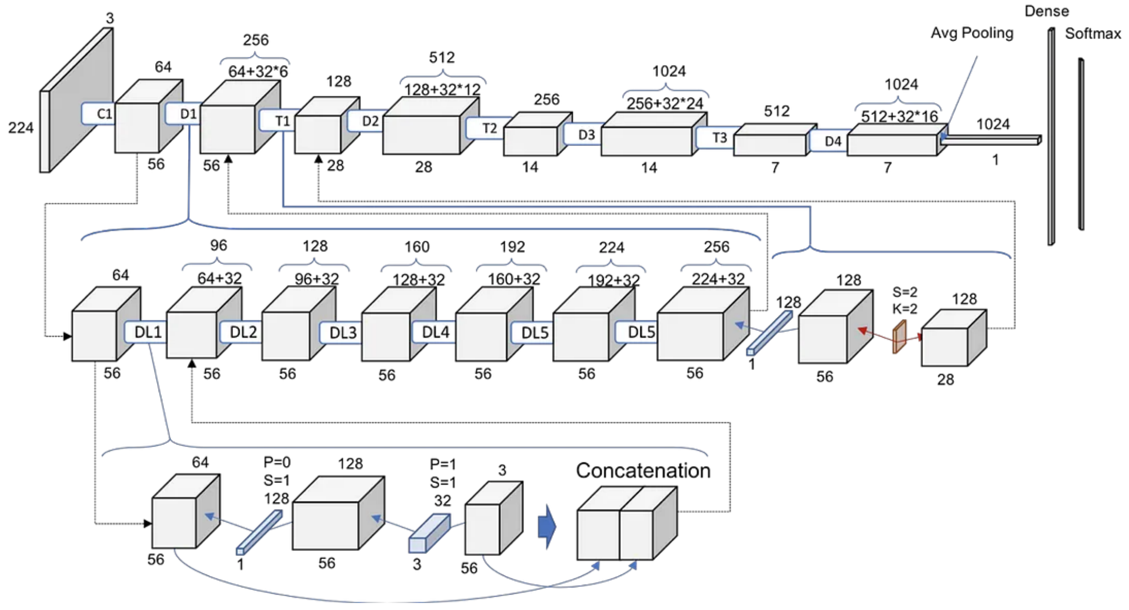
After each convolutional layer, followed by batch normalization, Rectified Linear Unit (ReLU) is utilized. The ReLU provides positive output for positive input, and zero value for negative input. The ReLU introduces non-linearity to the model, which is crucial for capturing complex relationships. Compared to other activation functions, such as Tanh and Sigmoid, the ReLU is computationally efficient due to its simplicity. Thus, the ReLU decreases the chances of the gradient vanishing, which is a typical issue in deep neural networks.

The last dense block is followed by average pooling, the purpose of which is to reduce each feature map to a 1x1 size. This drops the number of parameters for the following fully connected layer. Fully connected layers pass logits to the softmax activation function. The softmax activation function outputs probabilities distribution over four classes. The model architecture is shown in Figure 5.

To mitigate the vanishing gradient problem features are reused throughout the entire network, which improves the flow of the gradients. Such an approach allows the model to be compact and efficient in cases of limited computational resources. The model optimizer is Adaptive Moment Estimation (Adam), which holds adaptive separate learning rates for each parameter, providing more precise weight updates. To handle multi-class classification in DenseNet121 sparse categorical crossentropy loss function is used. Sparse categorical cross entropy is the typical choice in image classification tasks as it defines one class out of all classes. Also, the loss function directly accepts integer values assigned to each class, without the need to preprocess the data with one-hot encoding. In addition, this reduces computation time, especially when dealing with a large number of classes.

Figure 5

DenseNet121 architecture



Note. *https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a*

3.5 Transfer learning

To achieve better performance across all neural networks used in the project transfer learning concept is utilized. The transfer learning idea is to take the model that has been already trained on the quality large-size dataset and apply it to the related task, which is brain tumor detection. In this project four pre-trained models are used: VGG-19, ResNet-50, Inception V3, and DenseNet121. The idea is to use already established weights of the neural networks and update them using the current brain tumor dataset. This process is also called a fine-tuning of the model. There are several benefits of such an approach. First is efficiency, which means that neural networks don't need large amounts of labeled data for fine-tuning. Second, since the model has already learned features at depth before fine-tuning, this speeds up the training process on the current dataset. Third, such an approach typically leads to a much better performance of the model. Transfer learning is very popular in areas, such as computer vision and natural language processing.

3.6 Ensembling

Since the project utilizes four different neural networks it is reasonable to try to aggregate predictions from each model and try to achieve better accuracy. This approach is called ensemble. There are different types of ensembling: bagging, boosting, and stacking. Bagging is training several models each on an individual subset of the dataset. Boosting is the sequential training of models in such a way, that every next model is working on errors from the previous model. Stacking is using predictions from other models and combining them as input for another model to make final predictions. Ensembling provides several benefits, such as improved accuracy, reduced overfitting due to averaging predictions, and finally, all combined leads to highly accurate predictions considering different types of data and noise levels. Downturns are the increased computational complexity and time spent on training, and models must be selected mindfully to achieve a good combination for ensembling.

4. Model Evaluation Methods

For brain tumor detection and classification several different metrics can be used, such as accuracy, macro precision/recall/F1-score, and macro AUC-ROC (adapted for multi-class classification). When models have been trained and tested these metrics will represent a comprehensive evaluation of the models' performance.

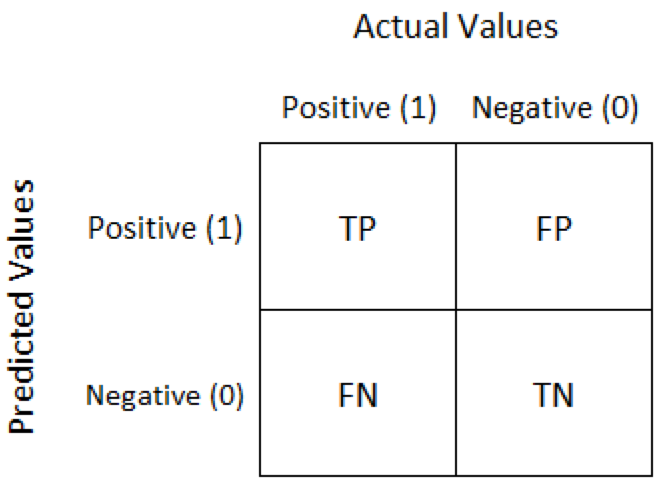
One of the most commonly used metrics to evaluate how well the model assigns correct labels in multi-classification problem is accuracy. The accuracy is the ratio of correct predictions to the total number of instances in the dataset. The definition of accuracy is shown in (1):

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Instances} \qquad (1)$$

Another way to represent the models' distribution of predictions to the ground truth labels is using the Confusion Matrix. The Confusion Matrix provides visually comprehensive distribution across multi-classes and estimates the portion of confused predictions. There are four categories shown, as follows: true positives, true negatives, false positives, and false negatives. The diagonal of the matrix corresponds to the correct predictions. Thus, the confusion matrix shows the errors made by the neural networks. The confusion matrix is shown in Figure 6.

Figure 6

Confusion matrix



For multi-class classification precision/recall/F1-score are calculated for each class. Precision is the ratio of true positives to the sum of true positives and false positives. Macro precision is the average precision determined across all classes. Classes are considered equal, despite the differences in frequency in the dataset. Thus, precision tells how much the model is trustworthy when predicting class as positive. The definition of precision is shown in (2):

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2}$$

Recall is the ratio of true positives to the sum of true positives and false negatives. Similarly to macro precision, macro recall is the average recall across all classes. It also considers all classes as equal. Thus, recall measures the ability of the model to capture all positive instances in the dataset. The definition of recall is shown in (3):

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3}$$

F1-score is a widely used evaluation metric that takes into account both precision and recall. It is useful when dealing with an imbalanced dataset. The macro F1-score is the harmonic mean of macro precision and macro recall, thus it involves taking each class as unweighted and considers classes of different sizes equally. The definition of F1 score is shown in (4):

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

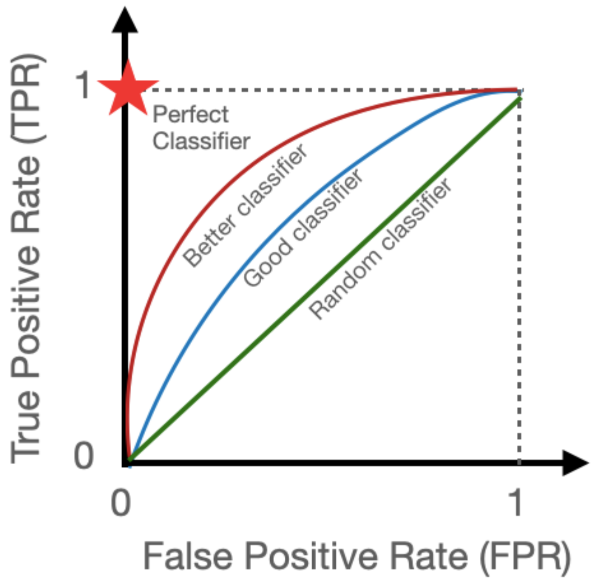
Also, it is possible to determine the weighted averages for precision/recall/F1-score, assigning weights corresponding to the size of each class. For the imbalance dataset, it might be more suitable to apply weighted averages. The closer precision/recall/F1-score metrics are to 1, the better the model's ability to assign the correct label for each instance.

The AUC-ROC, or the area under the receiver operating characteristic curve, score is a robust metric to evaluate model performance at various thresholds. The true positive rate is plotted against the false positive rate at a variety of thresholds. The true positive rate is also known as recall, and the false positive rate is defined as the ratio of false positives to the sum of both the number of false positives and true negatives. For the multi-class classification problem, the macro AUC-ROC score can be calculated, by taking the averages of individual AUC-ROC scores for each class. There are two approaches to taking individual AUC-ROC scores before averaging: one versus rest or one versus one. A completely inaccurate model would have an

AUC-ROC score of 0.5, while a robust model would have a score of 1. The area under the receiver operating characteristic curve (AUC-ROC) is shown in Figure 7.

Figure 7

Area under the receiver operating characteristic curve (AUC-ROC)



5. Data Exploration and Processing

The image data for this project comes from the “Brain Tumor Classification (MRI)” dataset on Kaggle. The dataset is divided into two folders: “Testing” and “Training”. Within each folder, there are four folders, which contain images for the classes “glioma_tumor”, “meningioma_tumor”, “no_tumor”, and “pituitary_tumor”. Thus, the task for this project involves multiclass image classification comprising four distinct classes.

5.1 Data Cleaning

The first approach that was taken was to perform multiple data cleaning steps to verify the integrity of the data and to ensure there will not be any issues during the modeling phase. The images were checked using the PIL library’s ‘verify’ function, and by using the OpenCV library.

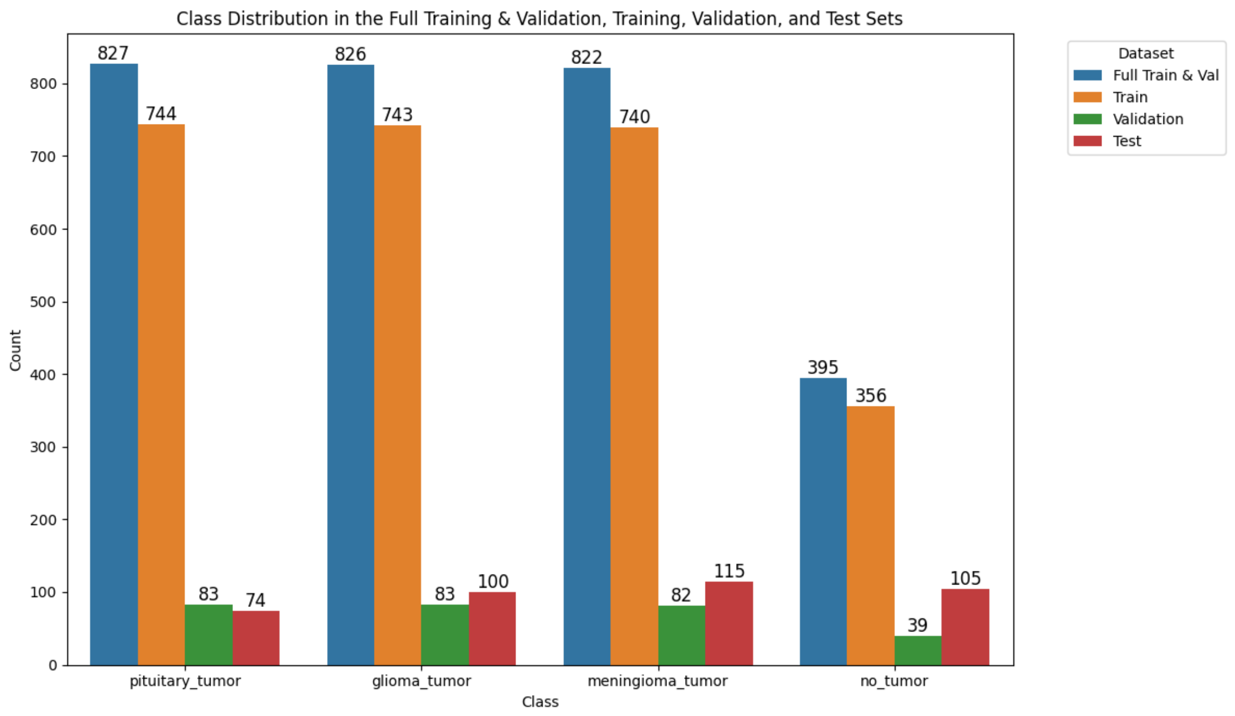
All of the image files were found to be valid (not corrupted). Next, all images that were less than 32x32 or greater than 2048x2048 were removed since they might cause issues during modeling (no images met these conditions). Lastly, all files that were not images were removed.

5.2 Creation of a Validation Set and Class Distributions

In order to monitor overfitting in the later modeling phase, 10% of the images from the original train set (from the "Training" folder) are set aside to be used as a validation set and the remaining 90% are used as a new train set, and both of these sets maintain the same class distribution from the original "Training" folder. The "Testing" folder is not modified and is used as the test set. The class distributions of the original "Training" set, the new training and validation sets, and the test set are shown in the figure below:

Figure 8

Class Distribution of Every Dataset

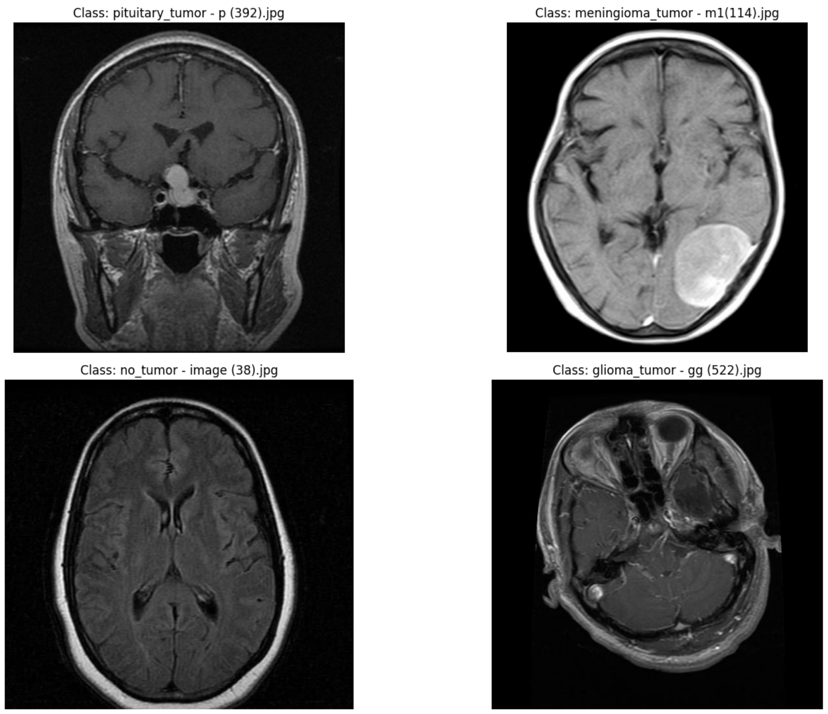


5.3 Displaying Sample Images

A sample image from each class - “glioma_tumor”, “meningioma_tumor”, “no_tumor”, and “pituitary_tumor”, is shown in the figure below:

Figure 9

Sample Image from each Class

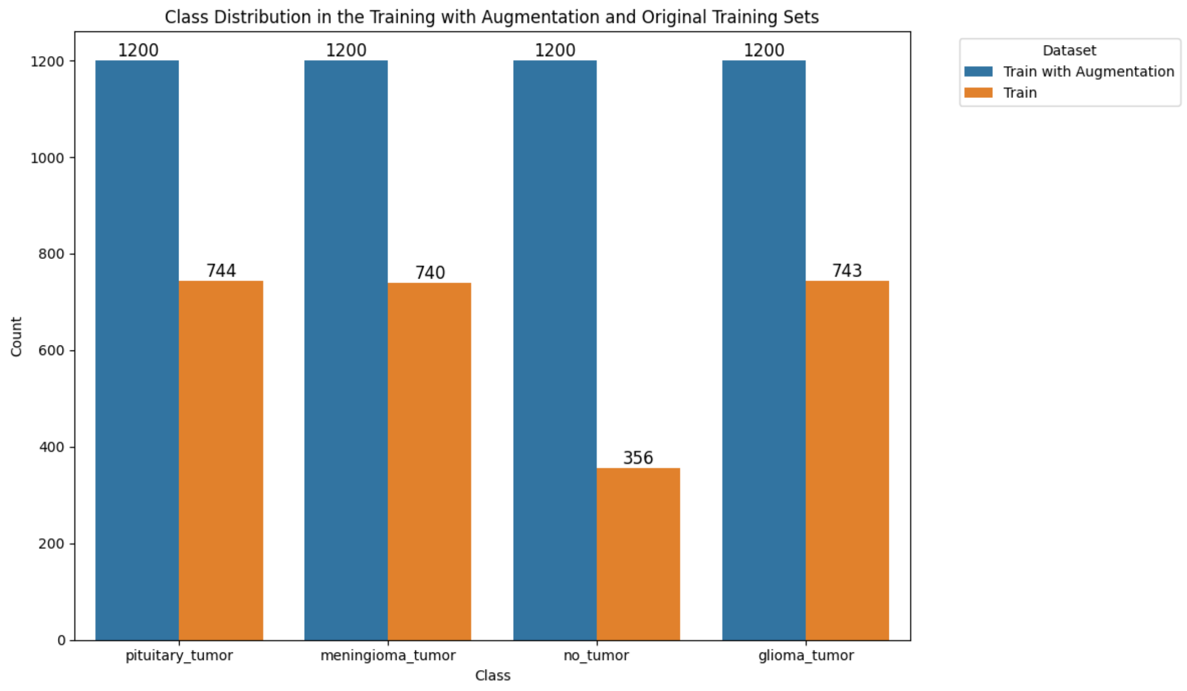

Class: pituitary_tumor - p (392).jpg
Class: meningioma_tumor - m1(114).jpg
Class: no_tumor - image (38).jpg
Class: glioma_tumor - gg (522).jpg

5.4 Data Augmentation

In order to increase the size of the data and counter the imbalance present in the class variable, we utilized data augmentation on the train set. Augmentation was not performed on the validation and test sets since doing so may lead to artificially high results for the model. Using data augmentation techniques on the train set produces more diverse types of data, which may lead to the model generalizing to a greater number of unseen images. The distribution of the class variable in the train set before and after augmentation is shown in the figure below:

Figure 10

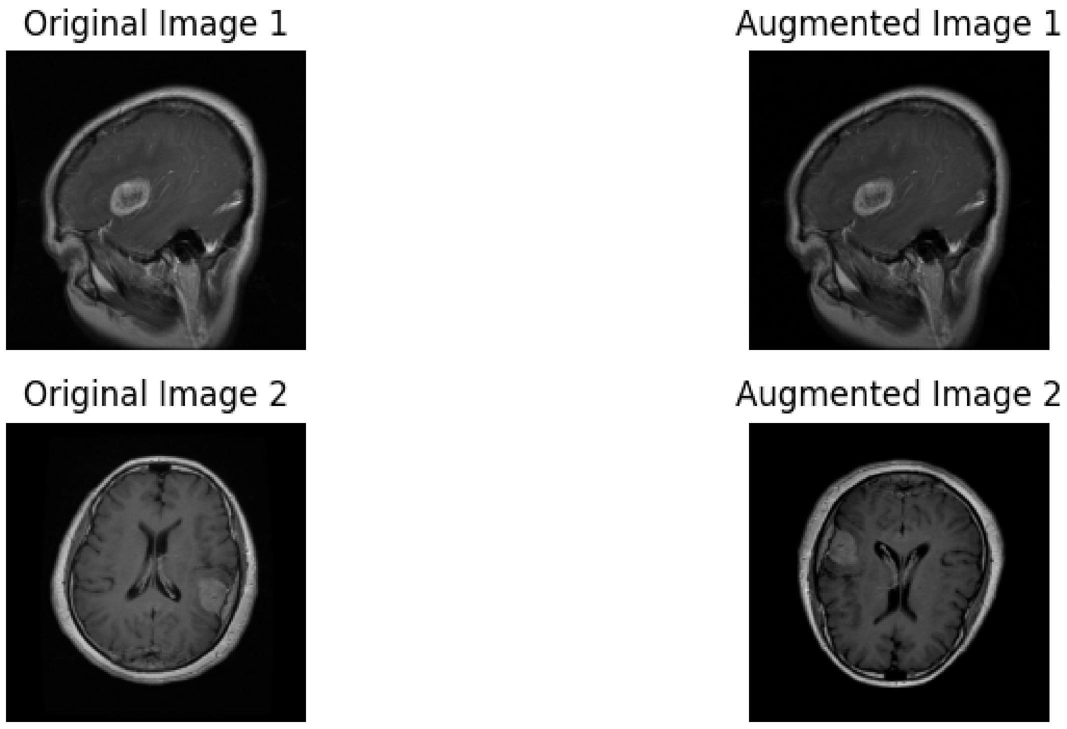
Distribution of the Class Variable in the Train Set, Before and After Augmentation



The images in each class are augmented to reach 1200 images. The specific augmentations that are performed include random 90-degree rotations, contrast, and brightness changes. The rotations are chosen at random and can constitute a 0-degree rotation (no rotation), or rotations by a factor of 90 degrees. These rotations were utilized so that the model can be more robust to varying rotations and hopefully generalize better to new images. The changes in contrast and brightness were chosen at random from a small range, specifically a factor of 0.2 for contrast and 0.1 for brightness (e.g. the images can at max be darkened or brightened by a factor of 10%). The contrast and brightness adjustments were kept to a small range in order to maintain the integrity of the medical images. These augmentations were used to help the model generalize better to MRI images which may have some slight variability in their contrast and brightness levels. An example of a couple of original training images, and their augmented counterparts is shown below:

Figure 11

Original Images and Augmented Images from the Train Set

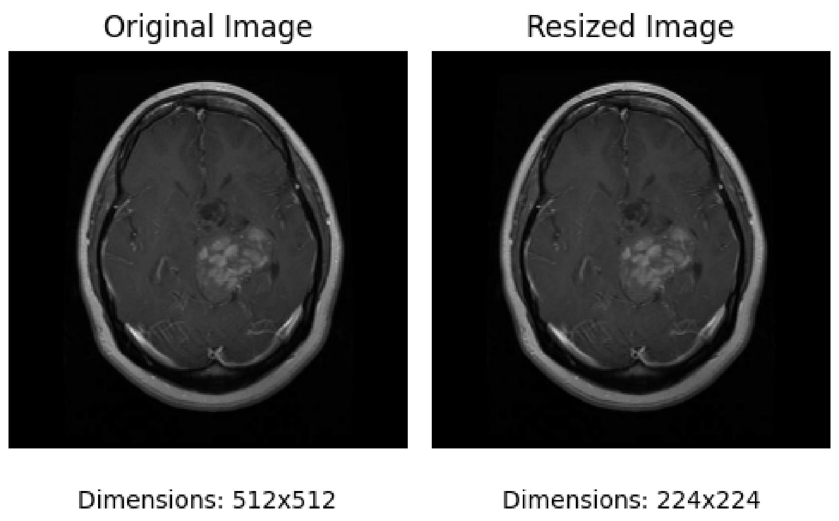


5.5 Compatibility with Pre-Trained Models and TensorFlow

The next step is to prepare the train, validation, and test sets to be compatible with the pre-trained models, namely VGG-19, Inception V3, ResNet-50, and DenseNet121. The images in these datasets are of varying sizes. The pre-trained models were trained on datasets with specific image dimensions, and thus the images in each dataset need to be resized in order to be compatible with the pertinent pre-trained model. For instance, in the case of VGG-19, the images need to be resized to have dimensions 224x224, and for Inception V3, the images should have dimensions 299x299. The images in the dataset are not fairly large and can be resized to lower dimensions without a significant loss of detail, and thus techniques like padding and antialiasing are not necessary. An example of an original image, and a resized image is shown below:

Figure 12

Original Image and Resized Image side-by-side



After resizing, the pixel values of the images need to be adjusted to match what the pre-trained model expects. This was done using each pre-trained model's "preprocess_input" function within their specific module in TensorFlow. For instance, for VGG-19, "tensorflow.keras.applications.VGG-19.preprocess_input" was used. The steps to preprocess the datasets to be compatible with the pre-trained models were wrapped into a function. The train, validation, and test datasets were converted to be compatible with training in TensorFlow using the 'Dataset' API. The "from_tensor_slices" function was used to create a dataset from the image path and label tuples (the labels were encoded with a custom mapping), and then the aforementioned preprocessing function was used to process the images and get them in a state where they are compatible for the pertinent pre-trained model. The training set was shuffled to ensure that bias or irrelevant patterns were not introduced/acquired in the model training phase due to the order of the images. The training, validation, and test sets were batched, cached, and prefetched with TensorFlow's 'AUTOTUNE' to increase the efficiency of the computations, and to decrease the time at which the model converges. With the datasets prepared for training in TensorFlow and compatible with each pre-trained model, the next step is the modeling phase.

6. Data Modeling

6.1 Initial Training

Each of the base pre-trained models was loaded using their respective modules in TensorFlow. For instance, the base VGG-19 model was instantiated by calling "tensorflow.keras.applications.VGG-19.VGG-19". For each of the pre-trained models, the 'weights' parameter was left as 'imagenet', and the 'include_top' parameter was set to False to remove the last few fully-connected layers at the top of each network. The input_shape parameter was set to the value that was used during data preprocessing, namely the input size that the pre-trained model expects (for example (224, 224, 3) for VGG-19). After the base pre-trained model was instantiated, all of the model's layers were frozen by setting the "trainable" variable to False. Afterward, a GlobalAveragePooling2D layer was added (to reduce the dimensions of the feature maps and retain the most important information), along with different combinations of Dense layers with a varying number of neurons and activation functions, and Dropout layers to reduce overfitting. The last layer is a Dense layer with 4 neurons corresponding to the number of classes, and a 'softmax' activation function due to the task being multiclass classification. The optimizer used is "adam", and the loss function is "sparse_categorical_crossentropy" since the class labels were integer encoded instead of one-hot encoded. A callback with an early stopping mechanism was utilized to monitor if the model was overfitting. The validation loss was monitored with a patience level of 3. Thus, if the validation loss did not improve after 3 epochs, the training process was terminated. The loss and accuracy on the test set were then obtained to determine how well the model was performing to completely unseen data. Initially, the accuracy of each model on the test set was not very notable (for each model, a dropout rate of 0.2 after the GlobalAveragePooling2D layer without additional dense layers seemed to produce the best results on the test set, but the accuracy levels were all below 75%). Fine-tuning of the models was then employed in an attempt to achieve a better performance of the models.

6.2 Fine-Tuning and Model Ensembles

Fine-tuning involves unfreezing some of the top layers of the base pre-trained model, recompiling the model so that the changes can take effect, and to continue training the model after the initial training (when all of the layers were frozen). Fine-tuning allows the model to adapt better to the specific features of a dataset.

The initial epoch during the fine-tuning phase was set to the last epoch during the initial training. The same optimizer (with a lower learning rate since the model is much larger and can overfit easier), loss function, and early stopping mechanism were used during the fine-tuning stage. The number of frozen layers for each model was experimented with. Choosing the number of frozen layers requires some consideration since having a larger number of trainable layers is computationally more expensive and may lead to overfitting (since the model may become very adapted to the training data and may not generalize well to new images). After the four fine-tuned models were finalized after some experimentation with the number of frozen layers, model ensembles were created. One ensemble involved using soft-voting (where the final class prediction is made by averaging the class probabilities from each model), and the other involved majority voting (where each model returns a class label and the class with the most votes is returned as the final prediction). The results of the individual models with the best combinations of parameters during initial training and fine-tuning, and the ensemble models are shown in the table below.

Table 1

Results of the Individual Models and Ensemble Models

Model	**Initial Test Accuracy**	**Fine-tuned/ Final Test Accuracy**
VGG-19	67.51%	78.68% (17/22 layers frozen)
Inception V3	69.54%	**84.01%** (200/311 layers frozen)
ResNet-50	74.11%	81.98% (100/175 layers frozen)
DenseNet121	66.75%	79.95% (300/427 layers frozen)
Ensemble (Soft Voting)	N/A	81.73%
Ensemble (Majority Voting)	N/A	81.98%

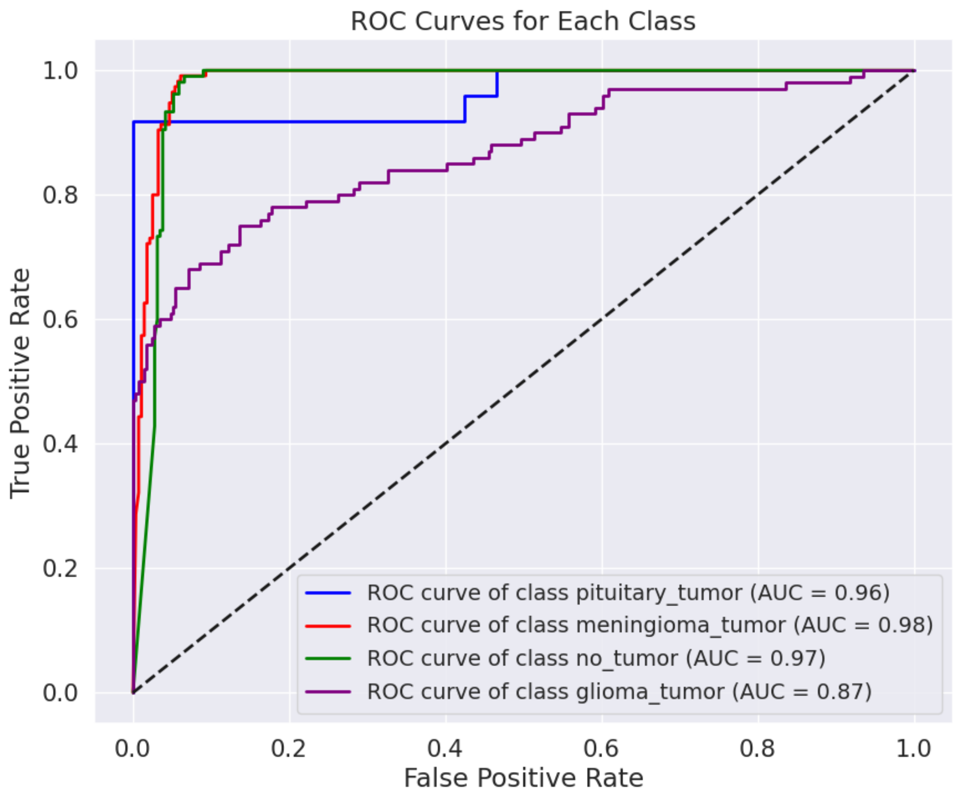
The individual models had significantly improved accuracy on the test set after fine-tuning. The fine-tuned Inception V3 model had the highest test set accuracy out of all of the models, including the ensemble models (the ensemble models most likely performed worse since the errors of the individual models are correlated; the individual models are likely not diverse enough). Thus, the final model that is chosen is the fine-tuned Inception V3 model. Various metrics and visualizations of the results of the model are shown below.

6.3 Results of the Final Model

The ROC curves for each class (using a one-vs-rest approach) are shown in the figure below:

Figure 13

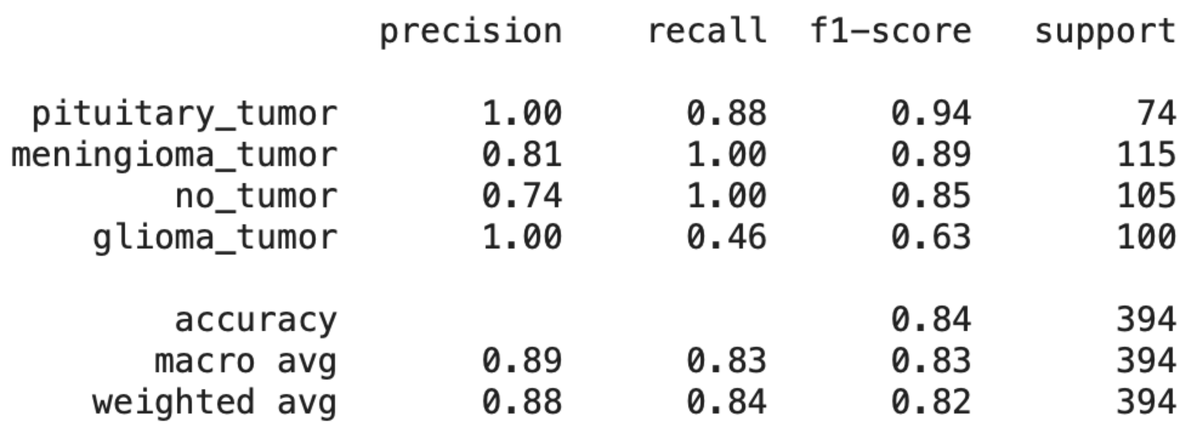
ROC Curves for Each Class, OvR Approach



The ROC curves and corresponding AUC scores show that the model distinguishes exceptionally well between positive cases and non-positive cases for the pituitary tumor, meningioma tumor, and no tumor classes. The glioma tumor class is noticeably worse than the other 3 classes, with a lower true positive rate for the same levels of false positive rate compared to the other classes. The macro average ROC-AUC score is 0.95, which indicates good distinguishing ability of the model when considering all classes. The classification report, which includes the precision, recall, and F1-score for each class, is now shown.

Figure 14

Classification Report

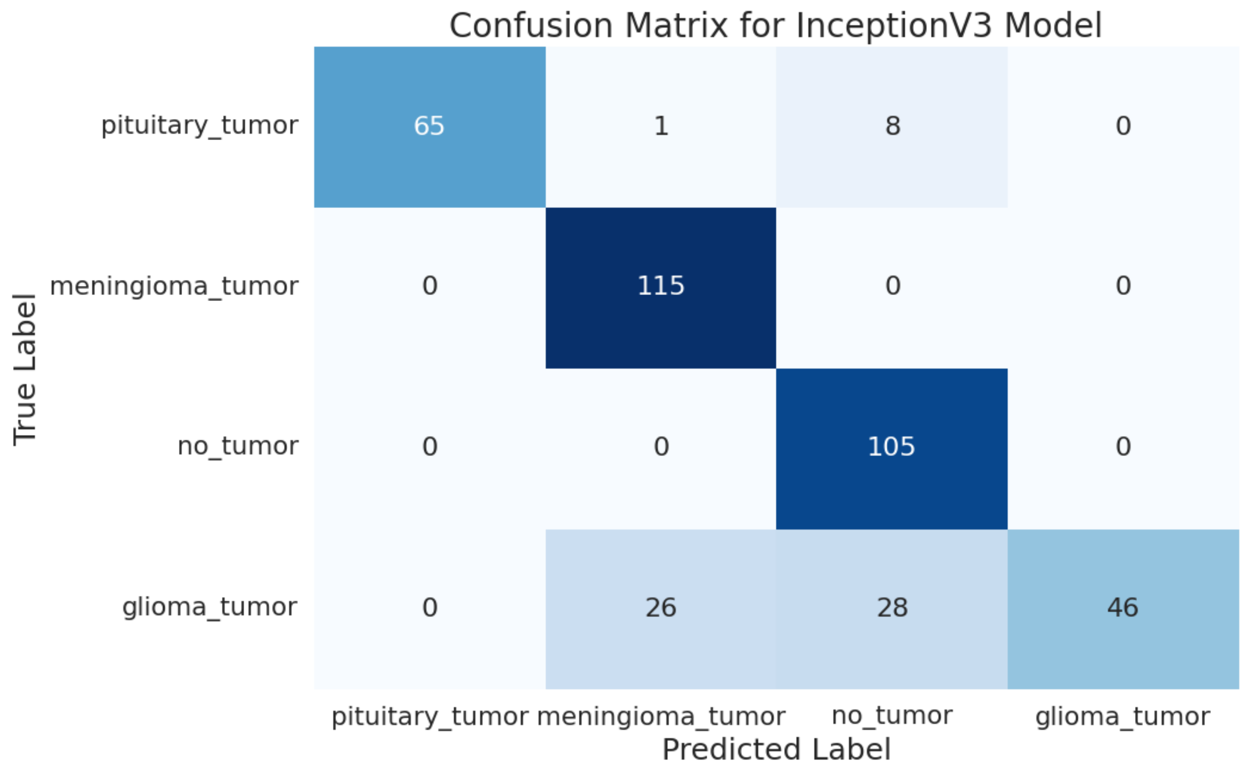


```
                  precision    recall  f1-score   support

 pituitary_tumor       1.00      0.88      0.94        74
meningioma_tumor       0.81      1.00      0.89       115
        no_tumor       0.74      1.00      0.85       105
    glioma_tumor       1.00      0.46      0.63       100

        accuracy                           0.84       394
       macro avg       0.89      0.83      0.83       394
    weighted avg       0.88      0.84      0.82       394
```

The F1-scores for the pituitary tumor and meningioma tumor classes are the highest. Recall for the pituitary tumor class is 0.88, indicating some false negatives and thus some pituitary tumor cases were predicted to be other classes by the model. The precision of 1 indicates that every time the model made a positive prediction for the pituitary class, it was correct. A precision of 0.81 for the meningioma class indicates false positives for this class. A recall of 1 indicates that every case of meningioma tumor was identified (i.e. no false negatives). The high F1-scores reflect the perfect precision and perfect recall of the pituitary tumor and meningioma tumor classes, respectively. The no tumor class has perfect recall, but a precision of 0.74 indicating some cases with tumors were incorrectly predicted to have no tumors. The glioma tumor class has perfect precision, but a poor recall which is reflected in the F1-score of 0.63. The low recall for the glioma class indicates that many cases of glioma tumor were incorrectly predicted to be other classes. A macro average F1-score of 0.83 indicates good performance when all of the classes are considered together. The confusion matrix is shown in the following figure.

Figure 15

Confusion Matrix



The confusion matrix goes into further detail into the results of the classification report. For instance, 65 cases of pituitary tumors were correctly identified and 9 cases of pituitary tumor were predicted to be other classes, which explains the recall of 0.88. All 115 cases of meningioma tumor were identified as such, and 27 cases of non-meningioma tumors were predicted to be a meningioma tumor case, which explains the recall of 1 and the precision of 0.81 for the meningioma tumor class. Similarly, for the no tumor cases, every instance was identified, while 36 cases of tumors (28 for glioma tumor and 8 for pituitary tumor) were incorrectly predicted to be of the no tumor class, which is reflected in the precision of the no tumor class. The low recall of the glioma tumor class is reflected in the fact that 26 glioma tumor cases were incorrectly predicted to be meningioma tumor cases, and 28 glioma tumor cases were incorrectly predicted to be no tumor cases.

The model demonstrates strong performance for most tumor types and the no-tumor class. However, the model most notably struggles in the recall of glioma tumors, often misclassifying them as having no tumor or a meningioma tumor, which indicates an area that could be targeted for improvement. Overall, when considering all of the metrics such as the macro average AUC score and macro average F1-score, the model is capable of distinguishing between classes with a high level of confidence.

References

AlTahhan, F. E., Khouqeer, G. A., Saadi, S., Elgarayhi, A., & Sallah, M. (2023). Refined automatic brain tumor classification using hybrid convolutional neural networks for MRI scans. *Diagnostics*, *13*(5), 864. https://doi.org/10.3390/diagnostics13050864

Anand, V., Gupta, S., Gupta, D., Gulzar, Y., Xin, Q., Juneja, S., Shah, A., & Shaikh, A. (2023). Weighted average ensemble deep learning model for stratification of brain tumor in MRI images. *Diagnostics*, *13*(7), 1320. https://doi.org/10.3390/diagnostics13071320

Asiri, A. A., Shaf, A., Ali, T., Pasha, M. A., Aamir, M., Irfan, M., Alqahtani, S., Alghamdi, A. J., Alghamdi, A. H., Alshamrani, A. F., Alelyani, M., & Alamri, S. (2023). Advancing brain tumor classification through fine-tuned vision transformers: A comparative study of pre-trained models. *Sensors*, *23*(18), 7913. https://doi.org/10.3390/s23187913

Asiri, A. A., Shaf, A., Ali, T., Shakeel, U., Irfan, M., Mehdar, K. M., Halawani, H. T., Alghamdi, A. H., Alshamrani, A. F., & Alqhtani, S. M. (2023). Exploring the power of Deep Learning: Fine-tuned vision transformer for accurate and efficient brain tumor detection in MRI scans. *Diagnostics*, *13*(12), 2094. https://doi.org/10.3390/diagnostics13122094

Brain tumor detection using deep convolutional neural network. (2023). *The International Conference on Scientific Innovations in Science, Technology, and Management*. https://doi.org/10.59544/poda4062/ngcesi23p130

Cahall, D. E., Rasool, G., Bouaynaya, N. C., & Fathallah-Shaykh, H. M. (2019). Inception modules enhance brain tumor segmentation. *Frontiers in Computational Neuroscience*, *13*. https://doi.org/10.3389/fncom.2019.00044

Cinar, N., Ozcan, A., & Kaya, M. (2022). A hybrid densenet121-UNET model for Brain tumor segmentation from Mri Images. *Biomedical Signal Processing and Control*, *76*, 103647. https://doi.org/10.1016/j.bspc.2022.103647

Duong-Trung, N., Da Quach, L., & Nguyen, C.-N. (2019). Learning Deep Transferability for Several Agricultural Classification Problems. https://doi.org/10.14569/IJACSA.2019.0100107

Krishnaswamy, S., Almas, B., & Rajasekaran, S. (2019). A deep analysis of Google Net and AlexNet for lung cancer detection. https://doi.org/10.35940/ijeat.B3226.129219

Nguyen T-H, Nguyen T-N, Ngo B-V. A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease. AgriEngineering. 2022; 4(4):871-887. https://doi.org/10.3390/agriengineering4040056

Siddaiah, N., Vignesh, C., Narendra, T., Pujitha, K., &amp; Madhu, D. (2022). Brain tumor detection from MRI scans using VGG- 19 networks. 2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22). https://doi.org/10.1109/icetet-sip-2254415.2022.9791683

Rathore, S., Rana, T., Mittal, U., Gupta, T., Malik, N., & A S, M. S. (2023). Brain tumor detection by using resnet-50 and image processing tools. *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. https://doi.org/10.1109/icacite57410.2023.10183081

Ruiz, P. (2018). Understanding and visualizing DenseNets. Medium. https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

Saeedi, S., Rezayi, S., Keshavarz, H., &amp; R. Niakan Kalhori, S. (2023). MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques. BMC Medical Informatics and Decision Making, 23(1). https://doi.org/10.1186/s12911-023-02114-6

Stephen, A., Punitha, A. & Chandrasekar, A. Designing self-attention-based ResNet architecture for rice leaf disease classification. Neural Comput & Applic 35, 6737–6751 (2023). https://doi.org/10.1007/s00521-022-07793-2

Vankdothu, R., Hameed, M. A., & Fatima, H. (2022). A brain tumor identification and classification using Deep Learning based on CNN-LSTM method. *Computers and Electrical Engineering*, *101*, 107960. https://doi.org/10.1016/j.compeleceng.2022.107960