Deep Learning for Predicting Brain Tumor From MRI Images

Juan Carlos && Geun Han

2023-04-04

Table of contents

Abstract

In this study, we investigate the application of two prominent deep learning architectures, ResNet50 and VGG-19, for the detection of brain tumors in MRI images using Python. Our objective is to build and compare the performance of these models for accurate and reliable brain tumor diagnosis. We employ a comprehensive dataset of MRI images, including both healthy and tumor-affected brain scans, to train and evaluate the models. The ResNet50 and VGG-19 models are fine-tuned with custom layers and hyperparameters for optimal performance. In addition to model training, we perform hyperparameter optimization and visualization of evaluation metrics such as the ROC curve, accuracy, loss, and confusion matrix for both models. Our findings demonstrate the potential of utilizing deep learning techniques, specifically ResNet50 and VGG-19 architectures, in aiding clinical decision-making by providing accurate and automated brain tumor detection from MRI images.

1 Introduction to the Project

1.1 Overview

Brain tumor detection from MRI images is a critical and challenging task in medical diagnosis, as early and accurate identification of tumors can significantly improve patient outcomes. Manual analysis of MRI images is time-consuming and subject to inter-observer variability, highlighting the need for automated approaches to assist medical practitioners and enhance the efficiency of their workflow.

1.2 Objective

This project aims to develop a machine learning-based system capable of detecting brain tumors in MRI images with high accuracy, which could potentially enable the general public to upload their MRI scans and receive feedback on the presence of tumors. It is important to develop such a program because as addressed in introduction, it is time-consuming for manual detection so by using such a model, it can increase and maximize efficiency of brain tumor detection for both hospitality infrastructure and patients.

2 Pre-Processing and Exploratory Data Analysis

Since we are using images as the datasets, we do not need to consider dropping images and only things that we need to do is to make sure that we are pre-processing the data into the format that the models will take in.

2.1 Data Pre-processing

Some issues to consider to be included: - Are there missing values in your data? Is there any pattern in whether or not a particular value is missing? How will the missing values be dealt with? We are inputting images as data for training and testing the models which means that there will not be missing values. However, there might be some limitations since we can pass in argument to the models to take in only a certain number of data. The only pre-processing of the data we are doing is to resize the image into the same size so that models will get the images in the same width and height. This is done by using library called tensorflow.

Transforming the data will increase the accuracy of the model because all the input data will have the same size which make the models to classify the brain tumors in the images easier.

2.2 Exploratory Data Analysis and Visualisations

Data visualization is done by matplotlib and seaborn. We are plotting the accuracy, confusion matrix and ROC curve to validate the models. We are also using “evalueate()” function to return the accuracy score of the models for another ascpect of validation.

3 Methodology

3.1 Platform and Machine Configurations Used

We used Google Colab for running our code and not on our machine because Google Colab provided GPU acceleration which means that we can train and test our models better in Google Colab.

3.2 Data Split and Making

We make our data into formats that we can pass into our model for training in two different methods. The first method we are using is "preprocess_image". This function we are taking the image and recode it until a form that we can resize the image and return the resized image. There are libraries that provides the functionality for reading the file and resizing the file which is tensorflow.io and tensorflow.image. We use "read_file", "decode_image", and "resize" functions from these libraries to successfully return the image that we want to pass into the models.

Another function that we created is called "load_data". The main functionality of this function is to successfully load the data into the module that we want, in this case, main.py. This function takes in 3 parameters. The first one is target. We can select the mode through this parameter. If we select "train", then the function will return the train dataset and if we select "test", the function will return the test dataset. The second paramenter is limit. This parameter can pass in an integer parameter which can help the function to control how many data that a model wants to train or test the model with. The third parameter is "randomize" which takes in a boolean parameter and check if the model wants to take in data from dataset randomly or not.

The way this function work is after model selected all the parameters, it will retrieve from the dataset all the images with brain tumors and without brain tumors for training and testing. After retrieving all the data, the function will store the data in X and Y array and return the array in a NDARRAY in numpy library and return it to the model.

3.3 Model Planning

The first algorithm that we are going to implement is VGG-19. The benefit of using this algorithm is that it can train data without data augmentation and further to tune the model using data augmentation. VGG-19 is consists of 19 layers which includes 16 convolutional layers, 3 fully connected layers, and 5 max-pooling layers, and some benefits of using VGG-19 is that it is a pre-trained model containing million

of images and thousand of classes. This means that the model has already learned a wide range of features and it can be fined-tune for some specific tasks and in this case, classifying brain tumor. It is also one of the models that has a strong feature extraction and has a fairly simple architecture. However, VGG-19 is also computational heavy and also takes up large memory which means it is harder for deployment with limited memory devices.

The second algorithm that we are implementing is ResNet50. The main benefit of using ResNet50 is the use of residual connections, which enable the network to learn identity functions. This helps alleviate the vanishing gradient problem and allows the network to train efficiently even with a large number of layers. It can also be scaled up or down depending on the specific problem and available computational resources. It also takes up lesser memory than VGG architecture.

3.4 Model Training:

We are basically fitting all the data that we are loading into our model by “fit” method. This method will allow us to fit all the data that we are loading into the model.

4 Model Evaluation

After training all the data, we are evaluating the models in different aspect:

4.1 Accuracy

We are trying to see the accuracy score that is returned from the `fit` method so we can see if the accuracy score is high enough or not.

4.2 Plotting ROC curve

We are also plotting ROC curve to see the area of the curve is big enough or not. Since the bigger the area gets, more accurate the model is, so we want the models to have a big ROC curve area.

4.3 Model Optimization

In this phase, grid_search is used to improve the accuracy of the model. We passed in different parameters for the model and tried to improve the accuracy to a sweet point where it is not overfitted but still reaches a high accuracy. We also use the epoch parameter to train the model so it can get the highest accuracy out of n times where n is epoch.

4.4 Final Model Building

After the model is successfully trained, we can see the final result by plotting all the different ascpects of the model. We have functions like “plot_accuracy”, “plot_confusion_matrix”, “plot_roc_curve” to visualize the aspects of our models and validate the models.

5 Results

5.1 Description of the Models

Both VGG-19 and ResNet50 are a type of CNN pre-trained model that we are using in this project. In this section, we are going to sepcify the performance and some graphs that we are getting from the result of the training and testing.

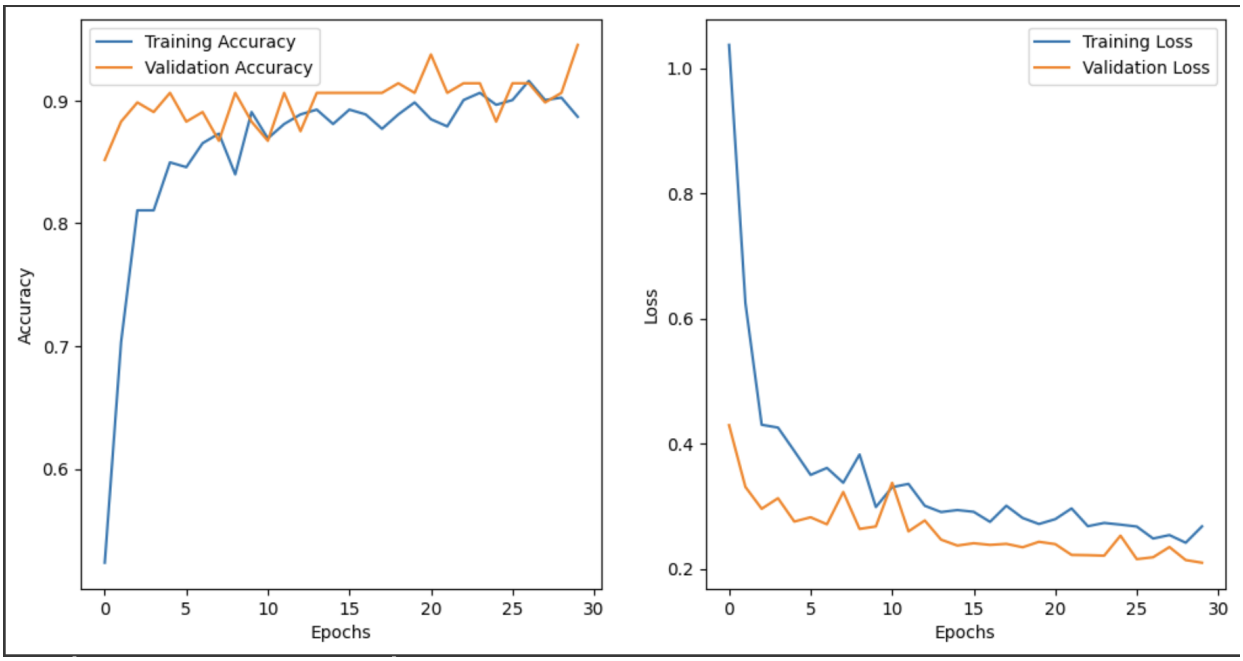
5.2 Performance Metrics

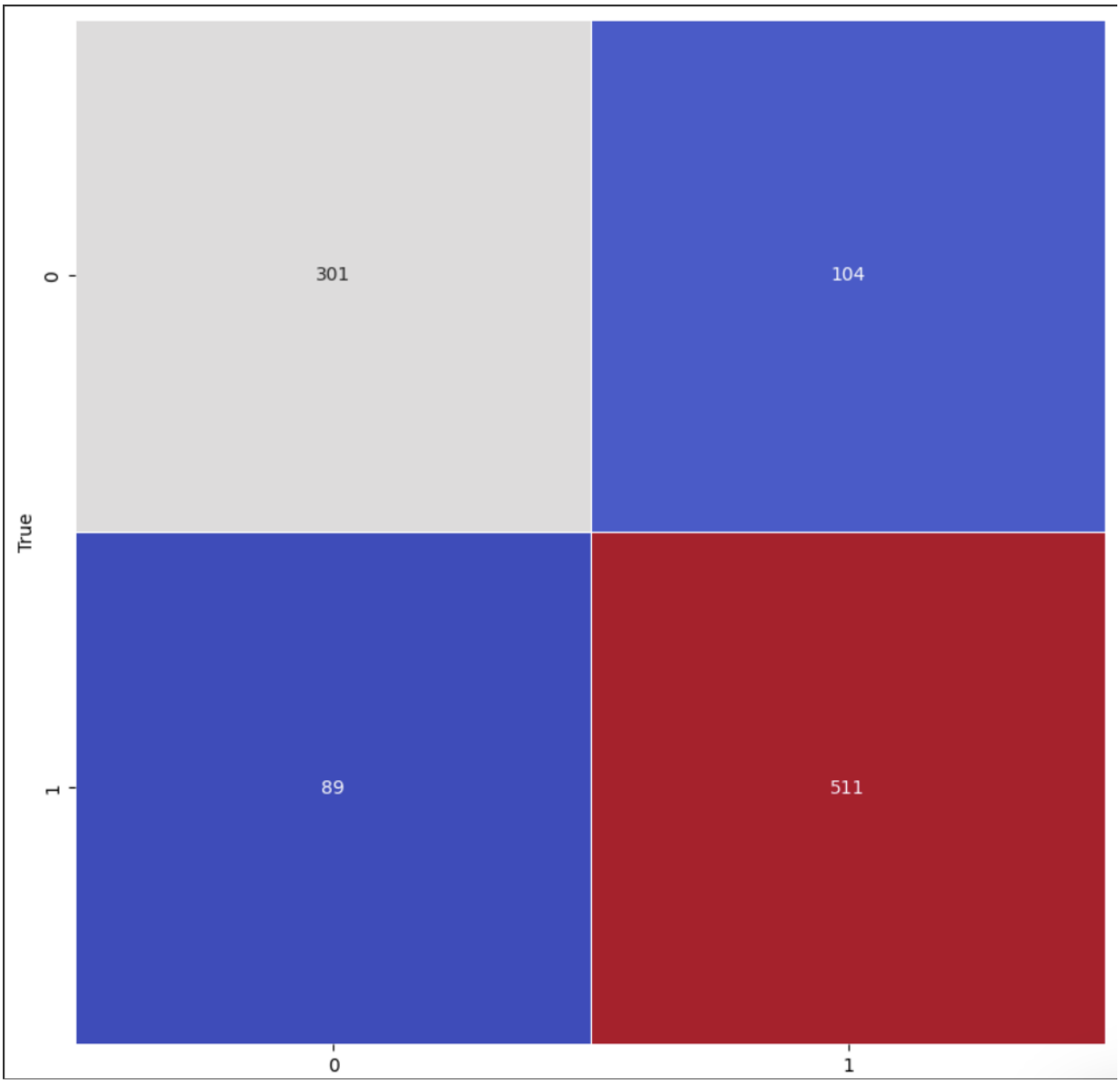
Some of the performance metrics that we are testing on the models are:

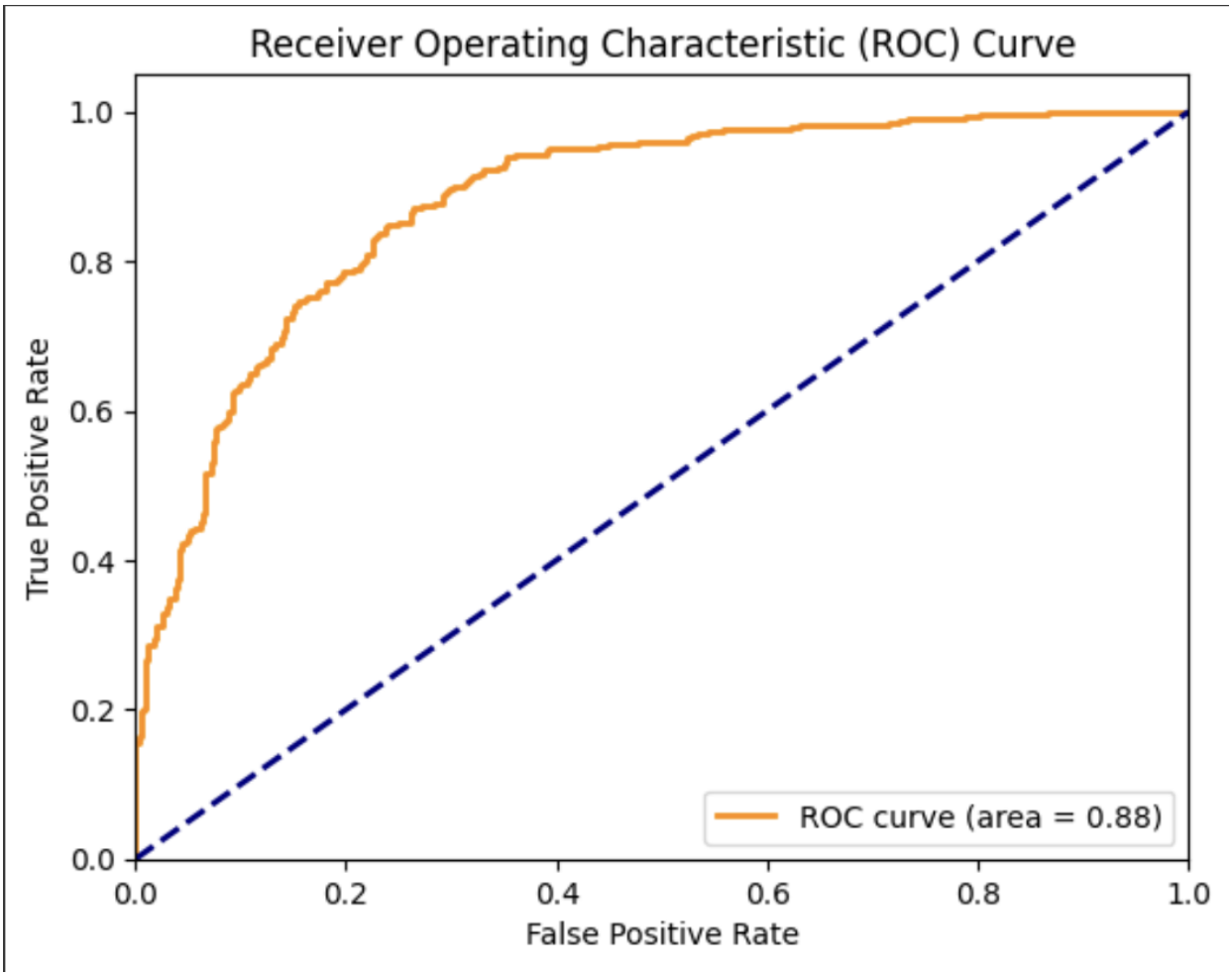
- Accuracy
- Precision
- Recall
- F1 Score
- AUC
- ROC Curve

5.3 Visualization

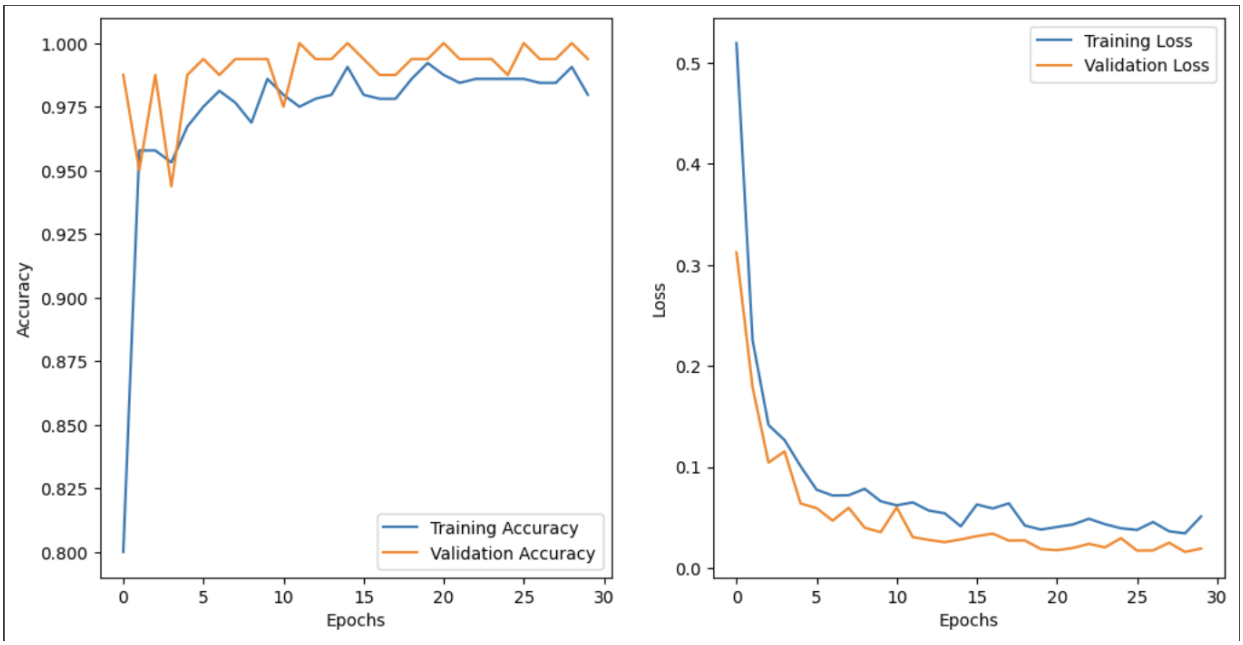
This are some of the images that we are getting from the ResNet50 model.

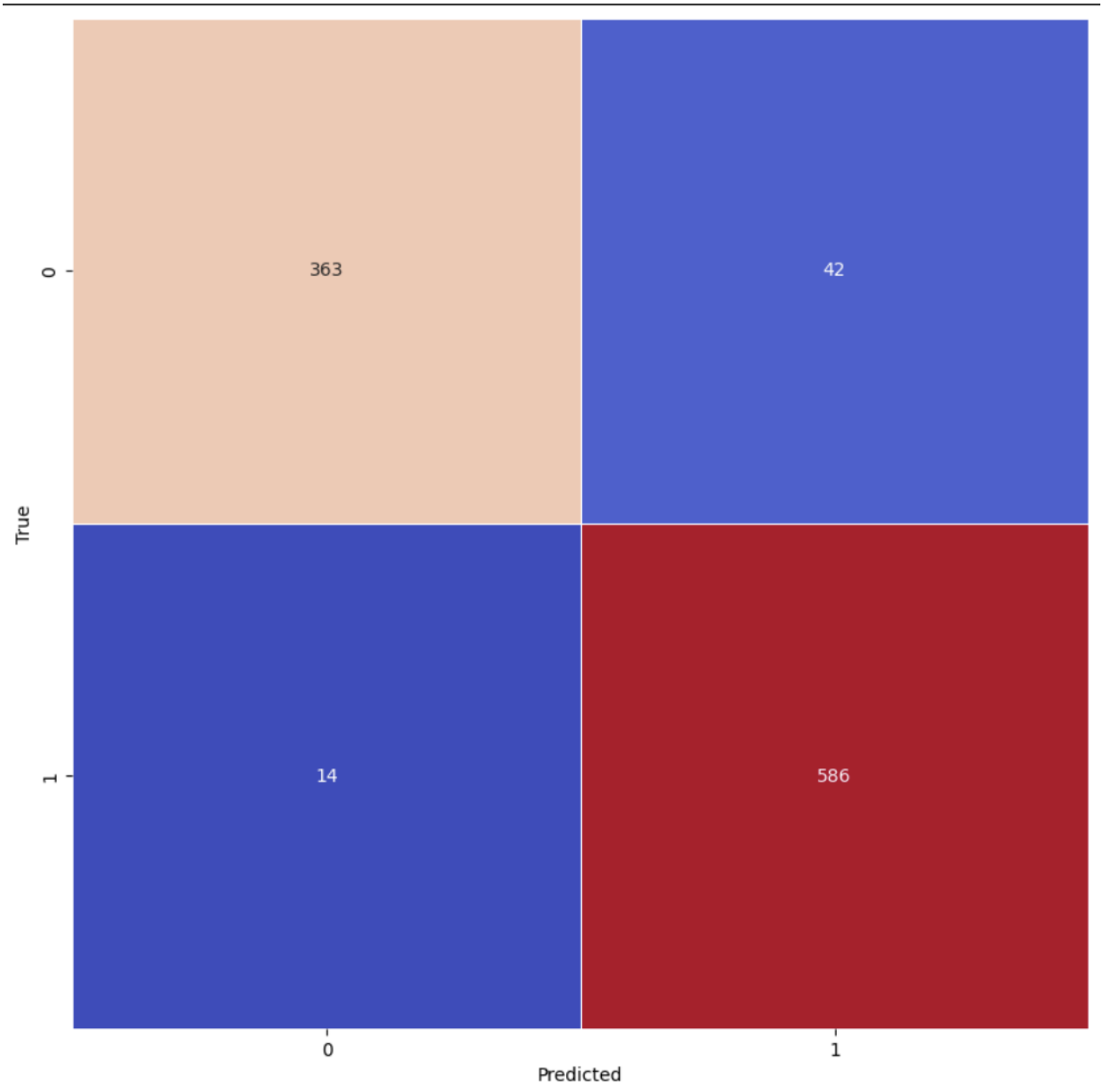

Training Accuracy
Validation Accuracy
Accuracy
Epochs
Training Loss
Validation Loss
Loss
Epochs

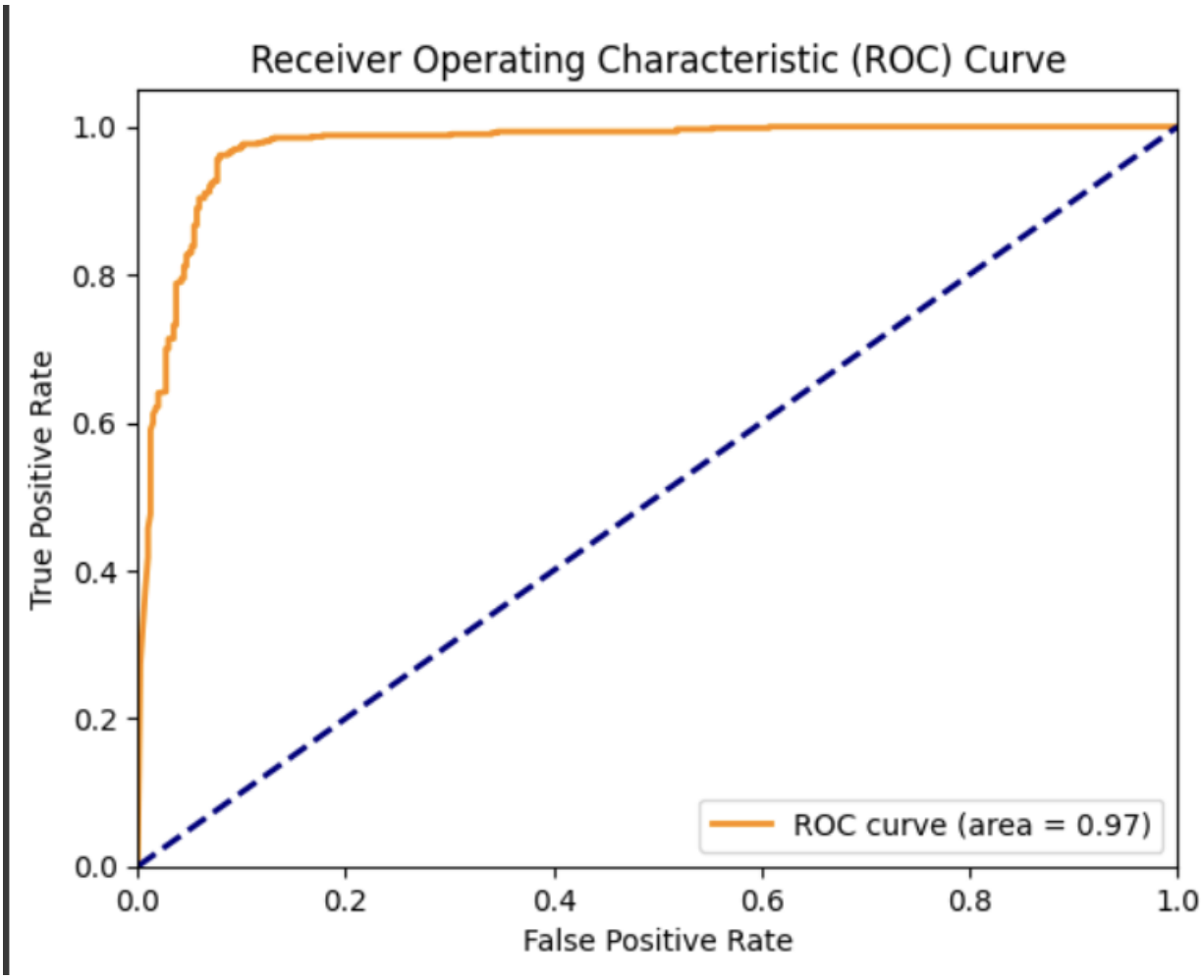

301
104
89
511
True
0
1
0
1



This are some of the images that we are getting from the VGG-19 model.


Accuracy
Epochs
Training Accuracy
Validation Accuracy
Loss
Epochs
Training Loss
Validation Loss


363
42
14
586
0
1
True
0
1
Predicted



5.4 Interpretation of the Results

Through all the visualization that we are providing, we can clearly see that VGG-19 model has a better result in ROC curve and in the confusion matrix. However, the accuracy and loss of the ResNet50 model similiar to the VGG-19 model. Due to these reasons, we are going to choose the VGG-19 model as our final model.

6 Conclusion

In conclusion, this project has successfully demonstrated the application of deep learning techniques, specifically ResNet50 and VGG-19 models, for brain tumor detection using MRI images. By leveraging the power of transfer learning and fine-tuning the models for this specific task, we have achieved promising results in terms of accuracy and generalization capabilities.

The project has also showcased the importance of data preprocessing and visualization in model evaluation and performance improvement. Through the implementation of various visualizations such as ROC curves, accuracy and loss plots, and confusion matrices, we have been able to effectively assess the models' performance, identifying potential areas of improvement.

It is important to note that the performance of the models depends on the quality and size of the dataset used for training and validation. Future work in this area could explore additional data augmentation techniques, as well as experimenting with other deep learning architectures and hyperparameter tuning to further improve the models' performance. Additionally, the integration of these models into clinical workflows could have a significant impact on the speed and accuracy of brain tumor diagnoses, ultimately improving patient outcomes and reducing the burden on healthcare professionals.

In summary, this project has demonstrated the potential of deep learning models, particularly ResNet50 and VGG-19, for brain tumor detection in MRI images. The results serve as a solid foundation for further research and development, as well as a stepping stone towards more efficient and accurate diagnostic tools in the field of medical imaging.