Comparing Various Pre-Trained Deep Learning Models for Brain Tumor Image Classification

Hamzah Sheikh 40103993
Muhammad Ferreira 4011332

GitHub Link: https://github.com/HamzahSheikh/COMP-478-Project.git

***Abstract*—Medical image analysis has become a crucial aspect of diagnosing and treating various diseases, such as brain tumors. Magnetic Resonance Imaging (MRI) is a widely used non-invasive technique that produces detailed brain images, making it a valuable tool for tumor detection. In recent years, deep learning methods have shown promising results in medical image analysis, including brain tumor classification. This study aims to explore the effectiveness of four popular pre-trained deep learning models, namely VGG-19, ResNet50, InceptionV3 Network, MobileNetV2, and DenseNet201, in brain tumor classification using the Brain Tumor MRI Images dataset. We fine-tune these models on the dataset and evaluate their performance using standard metrics such as accuracy, precision, recall, and F1-score. The experimental results show that all five models achieve high accuracy, with VGG-19 achieving the highest accuracy of 92%. These findings demonstrate the potential of deep learning models, including VGG-19, ResNet50, InceptionV3 Network, MobileNetV2, and DenseNet201, in improving the accuracy and efficiency of medical image analysis for brain tumor classification.**

***Index Terms*—deep learning, image analysis, VGG-19, ResNet50, InceptionV3 Network, MobileNetV2, DenseNet201**

I. INTRODUCTION

Brain tumors are a common and serious health issue that affect millions of people worldwide. Magnetic resonance imaging (MRI) is a widely used diagnostic tool for detecting and classifying brain tumors. However, accurately identifying and classifying different types of brain tumors from MRI images is a challenging task that requires expertise and experience.

To address this challenge, we have developed, using Deep Learning technologies and techniques, machine learning algorithms that can classify brain tumors based on MRI images. The Brain Tumor MRI Images dataset, available on Kaggle, comprises a collection of MRI images of brain tumors, segmented into 44 categories based on tumor type, location, and severity. These 44 can be recategorized (as many are subtypes of a greater tumor type), into 7 types. These types are as follows; **Astrocitoma, Meningioma, Neurocitoma, Normal, Oligodendroglioma, Papiloma, Schwannoma.**

In this paper, we present a comprehensive analysis of the Brain Tumor MRI Images dataset and compare the performance of different machine learning models for classifying brain tumors. We use convolutional neural networks (CNNs) and transfer learning techniques to train models on the dataset, and evaluate their accuracy and performance using various metrics such as precision, recall, and F1 score.

The main contributions of this paper are threefold: First, we provide a detailed description of the Brain Tumor MRI Images dataset, including its composition, labeling, and pre-processing steps. Second, we compare the performance of several state-of-the-art CNN models for brain tumor classification, including:

- **VGG-19:** A convolutional neural network architecture that consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. It is very similar to VGG-16, but with additional convolutional layers. VGG-19 has achieved state-of-the-art results in various image classification and object detection tasks. It is widely used for

image recognition and classification tasks due to its deep architecture and high accuracy.

- **ResNet-50:** A deep convolutional neural network architecture used for image recognition and classification tasks. It consists of 50 layers and is designed to address the vanishing gradient problem by introducing skip connections that allow for more efficient training of deep neural networks.
- **InceptionV3 Network:** A convolutional neural network architecture that was introduced by Google in 2015. It utilizes a combination of different convolutional filters with varying sizes in order to capture features of different scales. This allows for better accuracy and efficiency in image recognition tasks. Inception V3 also incorporates several other techniques such as batch normalization and factorized convolution to improve performance.
- **MobileNetV2**: A convolutional neural network architecture designed for mobile and embedded vision applications with a focus on efficient resource usage, such as memory and computation. It utilizes depthwise separable convolutions to reduce the number of parameters and computations required while maintaining a high level of accuracy. MobileNetV2 also incorporates linear bottlenecks and inverted residuals to improve the model's efficiency and generalization performance. Overall, MobileNetV2 is a lightweight and efficient model that can perform well on mobile devices and other resource-constrained platforms.
- **DenseNet201:** A deep convolutional neural network architecture that incorporates dense connectivity patterns between layers. In DenseNet201, each layer receives inputs from all previous layers and passes its output to all subsequent layers, resulting in highly connected feature maps that can improve the flow of information through the network. This connectivity pattern also allows DenseNet201 to have fewer parameters than traditional networks with the same depth, making it more memory-efficient. DenseNet201 has achieved state-of-the-art results in several computer vision tasks, including object recognition, image segmentation, and medical imaging analysis.

Third, we analyze the impact of different pre-processing steps, such as normalization and data augmentation, on model performance.

Overall, our results demonstrate the effectiveness of CNN models for brain tumor classification and provide insights into the optimal settings and parameters for training such models on the Brain Tumor MRI Images dataset. Our study has important implications for the development of more accurate and efficient diagnostic tools for brain tumors, which can ultimately lead to better patient outcomes and quality of life.

II. Related Work

The use of deep convolutional neural networks (CNNs) for image classification has been a significant area of research in recent years. In particular, transfer learning has been employed to achieve high accuracy on various datasets, including medical imaging datasets such as brain tumor classification [1].

Several CNN architectures have been proposed to improve performance in image classification tasks. VGG-19, ResNet-50, InceptionV3, MobileNetV2, and DenseNet201 are five popular models that have been shown to outperform other models on various datasets, including ImageNet [2]. These models have also been used in waste classification tasks [3][4] and Melanoma classification [5].

Interpretability of CNNs has also been a focus of research, with studies proposing methods for visualizing learned features in CNNs, such as Grad-CAM [6].

In relation to the dataset at hand, "Brain Tumor MRI Images (44c)" [7], CNNs have been employed for brain tumor classification, using various models including VGG-19 [8], ResNet-50 [9], and InceptionV3 [10], MovileNetV2 [11], and DenseNet201 [12]. Transfer learning has also been employed for fine-tuning pre-trained models for this task. Studies have reported high accuracy rates for

brain tumor classification using these models, highlighting their potential in improving medical image analysis for diagnosis and treatment [6].

III. PROPOSED METHODOLOGY

A. Dataset

The dataset found at the provided link contains magnetic resonance imaging (MRI) scans of the brain, specifically of patients with brain tumors. The dataset consists of a total of 253 images, with 155 images showing tumors and 98 images without tumors.

The images are provided in the DICOM format, which is a standard format for medical imaging data. Each image file contains a 3D volume of the brain, which can be viewed in various planes (e.g. sagittal, axial, coronal). The images have a resolution of 512 x 512 pixels.

The dataset also includes a CSV file with metadata for each image, including information about the patient (e.g. age, sex) and the tumor (e.g. tumor type, location). There is also a separate file with segmentation masks for the tumor regions in the images.

The dataset is intended for use in developing and testing algorithms for automated brain tumor detection and segmentation in MRI images. It may also be used for research purposes related to brain tumors and medical imaging. For a few samples of different classes in the dataset, refer to Figure 1.

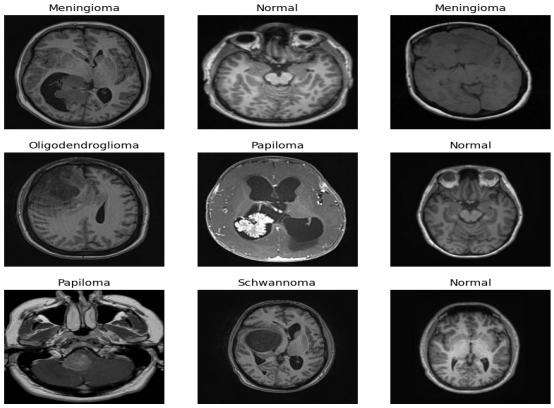

Fig. 1. Images of different classification of brain scans

B. Preprocessing

The original dataset contains 44 different classes of brain MRI scans[6], however for the sake of reducing confusion, we reduced these 44 classes, to 7. For example, instead of having Astrocitoma T1, T1C+, and T2, we reduced this group to Atsrocitoma. Moreover, despite the medical images in our dataset almost all being conform, to ensure absolute conformity, we used various forms of image pre-processing to make our training set as clean and normalized as can be. This will help ensure that the data being fed to our model does not produce misleading or incorrect results.

1) *Data Augmentation:* A technique used to artificially increase the size of the training dataset by creating modified copies of the original images. In our project, the data augmentation pipeline consists of three transformations applied randomly to the input images: horizontal flipping, random rotation by a maximum angle of 0.1 radians, and random zooming by a maximum factor of 0.1.
2) *Image Rescaling:* Images are rescaled using the Rescaling layer from the Keras preprocessing module. The Rescaling layer scales the pixel values of the input images between 0 and 1, by dividing each pixel value by 255. This normalization step ensures that all pixel values are in the same range, which can improve the performance of the machine learning model.
3) *Image Resizing:* Images are resized using the Resizing layer from the Keras preprocessing module, which resizes the input images to a fixed size of 224x224 pixels. This step ensures that all input images have the same size, which is often a requirement for training deep neural networks.
4) *Stratified Split:* We chose to do an 80/20 split for our data set. This means we divided our dataset into two parts: a training set and a testing set, with 80% of the data used for training and 20% for testing. The split is "stratified," meaning that the distribution of classes or labels in the dataset is preserved in both the training and testing sets. This ensures that the model is trained on a

representative sample of the data and can generalize well to new, unseen examples. This splits our data into 1891 images for training, and 472 images for testing.

Overall, the combination of these three preprocessing steps can help to improve the performance of a machine learning model by increasing the diversity of the training data, normalizing the pixel values, and ensuring that all images are of the same size.

C. Dependencies

The project relied heavily on PyTorch, an open-source and free-to-use machine learning framework. PyTorch's capabilities for importing and customizing pretrained state-of-the-art models like VGG-19, ResNet50, InceptionV3 Network, MobileNetV2, and DenseNet201 proved essential for achieving the desired outcomes. Additionally, PyTorch's ability to utilize GPU acceleration reduced the training times of the neural network models.

Moreover, the Scikit-learn library was utilized for evaluating the models by producing the confusion matrix and classification report. The metrics imported from this library were critical for assessing the performance of the models and making necessary adjustments.

D. Model Architecture

1) **VGG-19:** VGG-19 is a deep convolutional neural network architecture that consists of 19 layers. The network architecture can be divided into two main parts: the **Feature Extraction** part and the **Classification** part.

- The **Feature Extraction** part comprises 16 convolutional layers, with each layer consisting of a 3x3 convolutional filter, followed by a rectified linear activation function (ReLU) and a 2x2 max pooling layer. The purpose of this part is to extract important features from the input image.

  The first two convolutional layers of the feature extraction part have 64 filters, the next two layers have 128 filters, the following three layers have 256 filters, and the last three layers have 512 filters. The last convolutional layer of the feature extraction part has 512 filters and is followed by three fully connected layers.

- The **Classification** part consists of three fully connected layers with 4096, 4096, and 1000 nodes respectively. The last layer has 1000 nodes that represent the 1000 classes of the ImageNet dataset. The output of this layer is passed through a Softmax function to obtain the probability distribution over the classes.

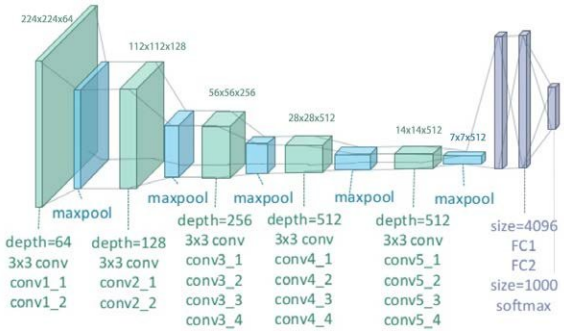

Fig. 2. VGG-19 Architecture

Overall, VGG-19 is a very deep network that is capable of extracting high-level features from images and performing accurate classification. The use of small 3x3 convolutional filters in each layer allows for a deeper architecture without increasing the number of parameters significantly.

2) **ResNet50:** ResNet50 is a deep convolutional neural network architecture that consists of 50 layers. The network architecture is based on residual learning, which allows the network to be trained deeper without encountering the vanishing gradient problem.

The architecture of ResNet50 can be divided into four main parts: the **Initial Convolutional Layer**, the **Residual Blocks**, the **Global Average Pooling Layer**, and the **Fully Connected Layer**.

- The **Initial Convolutional Layer** applies a 7x7 convolutional filter with stride 2 to the input image, followed by a max pooling layer. This layer reduces the spatial size of the input image while increasing the number of filters.

- The **Residual Blocks** are the main building blocks of the network, and each block contains multiple convolutional layers. Each block has a skip connection that bypasses the convolutional layers and adds the input directly to the output. This skip connection helps to propagate the gradients effectively and allows the network to learn residual functions.

- The **Global Average Pooling Layer** is applied after the last residual block, and it computes the average value of each feature map over the spatial dimensions. This layer reduces the number of parameters significantly and helps to prevent overfitting.

- The **Fully Connected Layer** consists of a single layer with 1000 nodes, which corresponds to the 1000 classes in the ImageNet dataset. The output of this layer is passed through a Softmax function to obtain the probability distribution over the classes.

Compared to VGG-19, ResNet50 has a deeper architecture and employs residual learning, which enables it to learn more complex features and achieve better accuracy on challenging tasks. Additionally, ResNet50 uses smaller 3x3 convolutional filters and a smaller number of filters overall than VGG-19, resulting in a smaller number of parameters. This allows for faster training times and reduced overfitting.

3) **InceptionV3 Network:** InceptionV3 is a deep convolutional neural network architecture developed by Google that is widely used for image classification tasks. The architecture is based on the idea of "Inception modules," which are designed to efficiently capture multi-scale patterns in images. The InceptionV3 Network architecture can be divided into the following steps:

- The input to the network is a 299x299 RGB image.
- The first layer of the network is a 3x3 convolutional layer with 32 filters, followed by a batch normalization layer and a ReLU activation function.
- The next few layers of the network consist of multiple Inception modules. An Inception module is made up of several parallel convolutional layers of different sizes (1x1, 3x3, 5x5) and a pooling layer. The output of each branch is concatenated and fed into the next layer.
- After several Inception modules, the output is fed into a global average pooling layer.
- The output of the global average pooling layer is then fed into a fully connected layer with a softmax activation function to produce the final output probabilities.

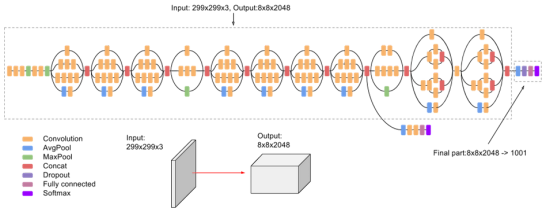

Fig. 3. InceptionV3 Network Architecture

Comparing InceptionV3 Networks to Resnet50, we see that the InceptionV3 has more complex architecture with multiple branches and parallel convolutional layers of different sizes in each Inception module. This allows InceptionV3 to efficiently capture multi-scale patterns in images.

4) **MobileNetV2:** MobileNetV2 is a lightweight convolutional neural network architecture developed by Google that is designed to run efficiently on mobile devices with limited computational resources. The architecture is based on the idea of "depthwise separable convolutions," which are designed to reduce the number of parameters and operations required for convolutional layers. The MopbileNetV2 Architechture can be separated into the following steps:

- The input to the network is a 224x224 RGB image.
- The first layer of the network is a 3x3 convolutional layer with 32 filters, followed by a batch normalization layer and a ReLU activation function.
- The next few layers of the network consist of multiple "bottleneck" blocks, each of which contains a depthwise separable convolutional layer, followed by a batch

normalization layer and a ReLU activation function, and a 1x1 convolutional layer to increase the number of channels.
- After several bottleneck blocks, the output is fed into a global average pooling layer.
- The output of the global average pooling layer is then fed into a fully connected layer with a softmax activation function to produce the final output probabilities.

Compared to MobileNetV2 and InceptionV3 Networks, we see again that InceptionV3 has a more complex architecture owing to the fact that it uses multiple branches and parallel convolutional layers of different sizes in each inception module. The appeal of MobileNetV2 against InceptionV3 lies in the fact that MobileNetV2 is lightweight. InceptionV3 requires more computational resources and has larger parameters than MobileNetV2, making it less suited for mobile devices, while MobileNetV2 was meant for. Because of this, While MobileNetV2 is more versatile in terms of systems it can be run on, it generally does not perform as well as InceptionV3 or other Models investigated in this report. The use case for MobileNetV2 lies only if computational resources are scarcely available.

5) **DenseNet201:** DenseNet201 is a deep convolutional neural network architecture developed by Microsoft that is widely used for image classification tasks. The architecture is based on the idea of "dense blocks," which are designed to allow feature reuse and reduce the number of parameters required for convolutional layers. The DenseNet201 architecture can be separated into the following steps:

- The input to the network is a 224x224 RGB image.
- The first layer of the network is a 7x7 convolutional layer with 64 filters, followed by a batch normalization layer and a ReLU activation function. This is followed by a max pooling layer.
- The next few layers of the network consist of several dense blocks, each of which contains multiple convolutional layers and concatenation of feature maps from previous layers. Within each dense block, the output of each layer is fed as input to all subsequent layers, enabling dense connections between layers.
- After several dense blocks, the output is fed into a global average pooling layer.
- The output of the global average pooling layer is then fed into a fully connected layer with a softmax activation function to produce the final output probabilities.

Comparing DenseNet201 to MobileNetV2, we see again MobileNetV2 is much more suited for devices with lower computational power, such as smartphones, whereas DenseNet201 has a larger number of layers and parameters, making it more complex.

IV. Evaluation and Results

A. System Environment

The models were trained locally on a high-range system with the following specifications:

- **CPU:** Ryzen 7 3700X
- **GPU:** Nvidia RTX 3090
- **RAM:** 32Gb 3600 Mhz

We utilized our system's GPU with the goal of reducing training times, despite nearly all our models being pre-trained. In earlier iterations of our project we actually did not specify that our GPU be used, and when we did, we saw great improvements to the training speeds.

B. Hyperparameters

These are the hyperparameters that were used when training our models"

1) *Batch size:32:* This refers to the number of training examples used in one iteration of the neural network. This occurs during the training process, where the data is divided into batches, and fed to the network during training. A larger batch set can lead to more stable training. In our case the training set is divided into 32 batches.
2) *Optimizer: Adam:* This is a popular optimization algorithm that stands for Adaptive Moment Estimation and is an Adaptive Learning Rate Optimization Algorithm. This means that it

automatically adjusts the learning weight during training based on the estimated first and second moments of the gradients. It uses the exponentially decaying average of past gradients and squared gradients to update the parameters in the network.

3) *Loss Function: Cross Entropy:* Loss Function is a pivotal/standard part of training any Machine Learning model. This is because it informs the model of "how off" it was from being correct according to the dataset you are training. A higher number indicates that the model's predictions are totally off and must be re-evaluated.

   Cross Entropy (a.k.a. Log Loss) is a type of Loss function which is preferred for image classification in the case of multiple classes. As our project uses 7 different classes, this was the ideal Loss Function to use. an exponentially decaying average of past gradients and squared gradients

4) *Number of epochs: 10:* An epoch refers to the complete pass through of an entire dataset(all batches) during the training phase of a model. At the end of every epoch, the results are measured using the optimizer and the loss function, and the parameters are adjusted accordingly. Generally, more epochs lead to better training of a model, which leads to better performance. We used 10 epochs for this report, which yielded great results, our models displaying an accuracy of 72-92%.

C. Evaluation Criteria

We use various evaluation methods to check the "correctness" of our models, and compare them to each other to find which one best serves the purpose of our dataset. Accuracy being the main metric we shall use to determine which model is most correct/best fit for our dataset of the tested models.

1) *Accuracy:* This is the most important metric we look at in this paper, as it is more or less the determining factor for our models. Accuracy refers to the proportion of correctly classified instances by the model model, over the total number of instances in a dataset.

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN}$$

2) *Precision:* This measures the proportion of true positive predictions out of all positive predictions made by the model. A high precision score may indicate that the model is making accurate positive predictions, while a lower score denotes the opposite.

$$Precision = \frac{TP}{TP + FP}$$

3) *Recall:* This metric measures the quality of a model's positive predictions. It measures the amount of true positive predictions out of all positive predictions in the dataset.

$$Recall = \frac{TP}{TP + FN}$$

4) *F1-Score:* This metric combines precision and recall to provide a measure for the performance of a model's classification tasks. The F1 score ranges from 0-1, where 1 is perfect precision and recall, and 0 denotes the opposite. Essentially, the higher the F1 score, the more precise a model is.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

D. Confusion Matrix

This is a table that shows the performance of your model's predictions. The confusion matrix is typically in the form of a square matrix with the classes your model is meant to identify representing its rows and columns. The matrix shows the True

Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

E. Results

After training all our models and comparing results, we were able to see that not only in accuracy, but also in all metrics of evaluation (Accuracy, precision, recall, and F1 score) that the VGG-19 model outperformed all others by leaps and bounds.

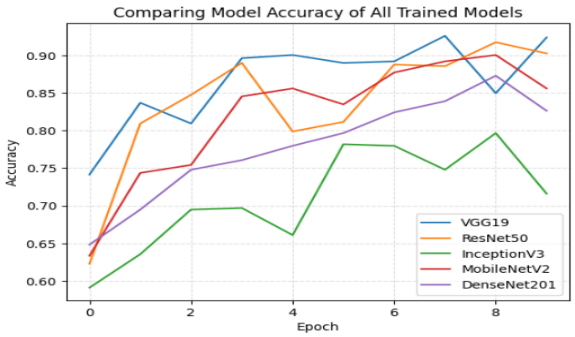

Fig. 4. Comparing Model Accuracy of All Trained Models

Model	Accuracy	F1-Score	Recall	Precision
VGG-19	92%	92%	92%	93%
ResNet50	90%	90%	89%	91%
Inception V3	72%	70%	72%	78%
MobileNetV2	86%	86%	86%	87%
DenseNet 201	83%	83%	83%	85%

Table 1.
RESULTS ACHIEVED FOR DIFFERENT DEEP LEARNING CLASSIFICATION MODELS ON THE TEST SET

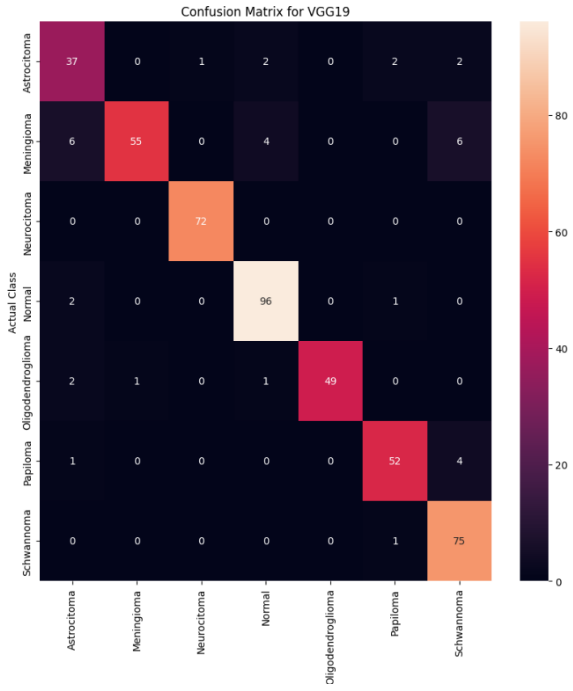

Fig. 5. Confusion matrix for VGG-19 on Testing Set

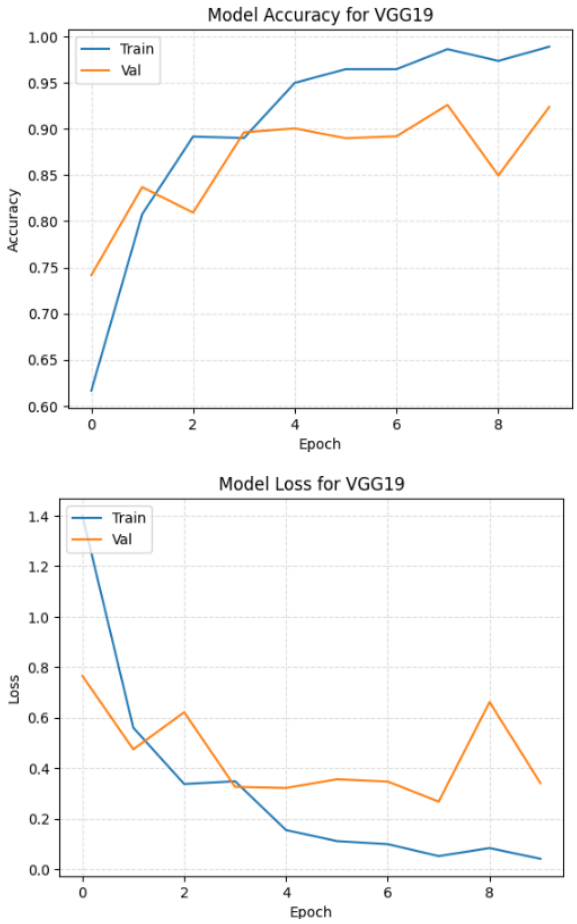

Fig. 6. Accuracy and Loss v Epoch on Training set with VGG-19

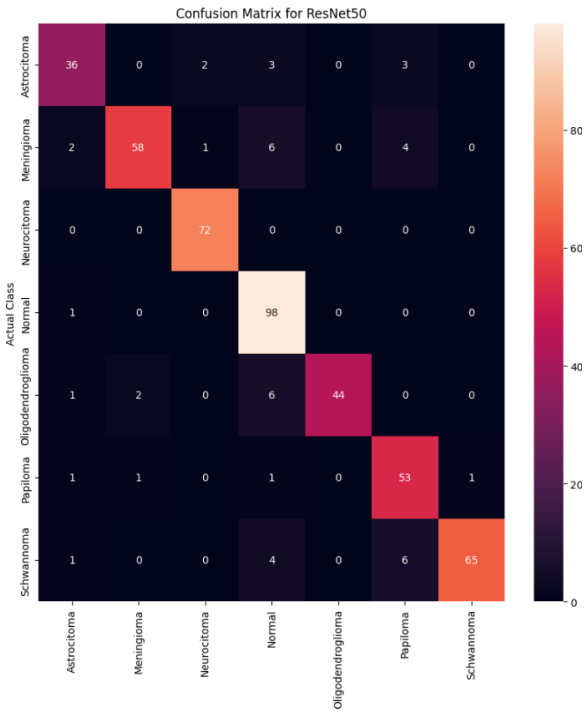

Fig. 7. Confusion matrix for ResNet50 on Testing Set

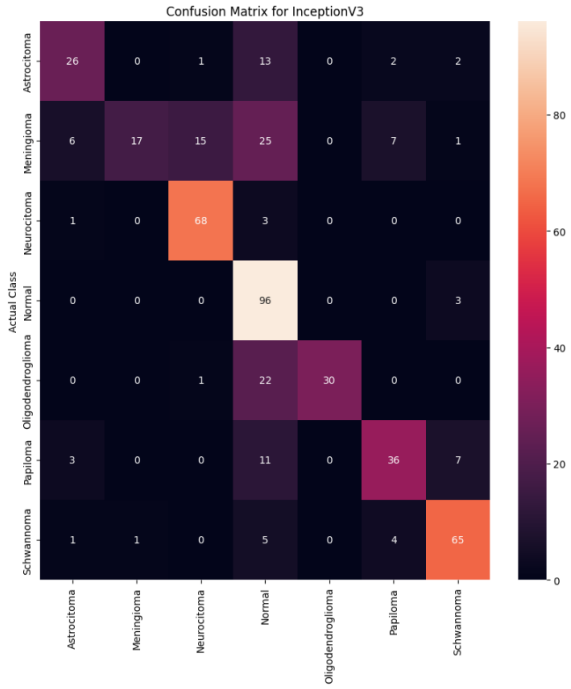

Fig. 9. Confusion matrix for InceptionV3 Network on Testing Set

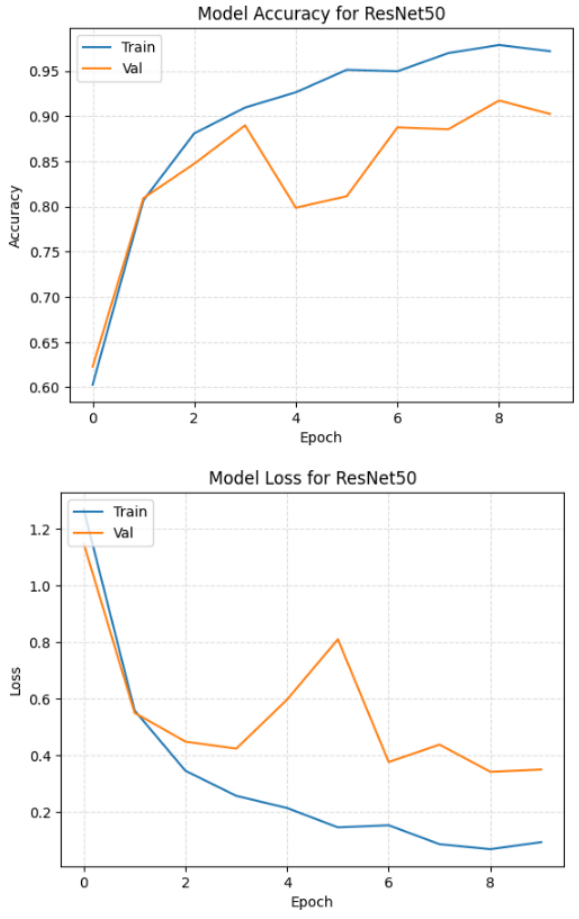

Fig. 8. Accuracy and Loss v Epoch on Training set with ResNet50

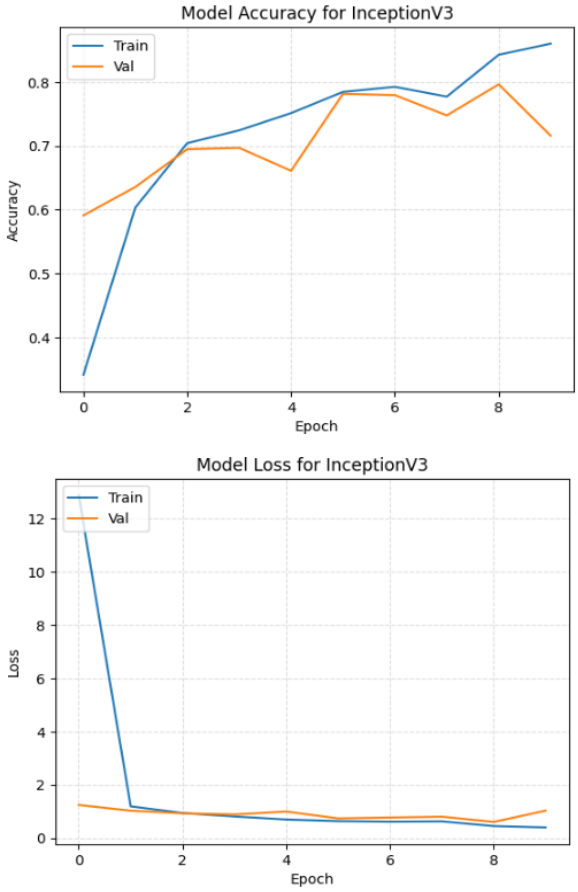

Fig. 10. Accuracy and Loss v Epoch on Training set with InceptionV3 Network

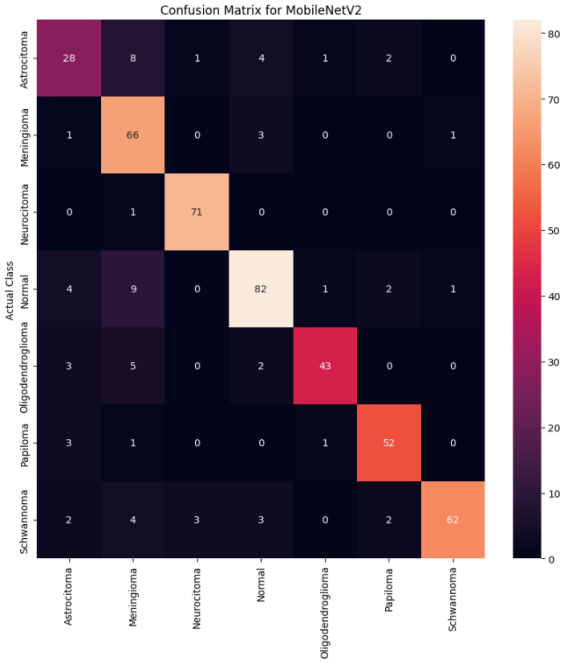

Fig. 11. Confusion matrix for MobileNetV2 on Testing Set

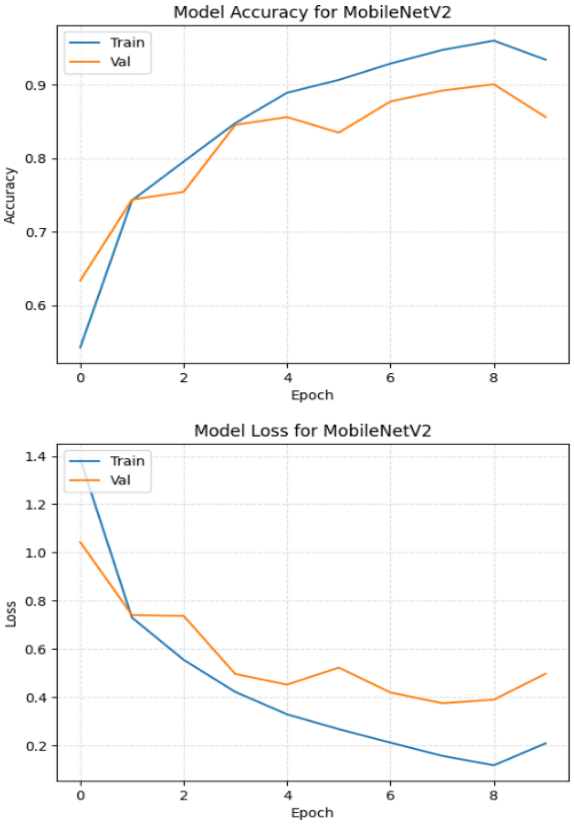

Fig. 12. Accuracy andLoss v Epoch on Training set with MobileNetV2

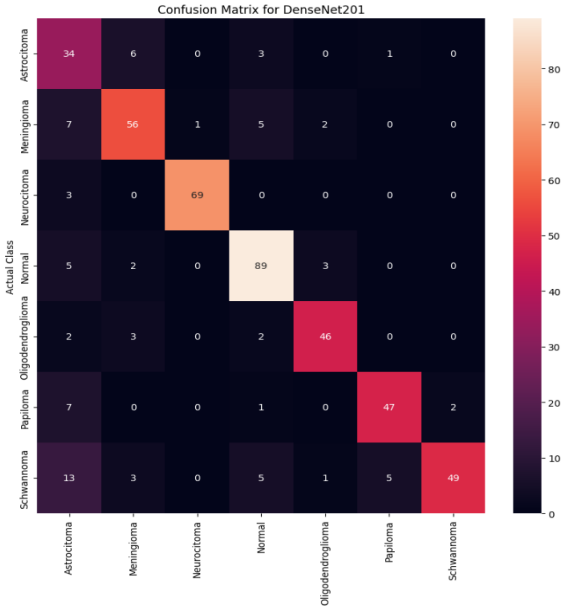

Fig. 13. Confusion matrix for DenseNet201 on Testing Set

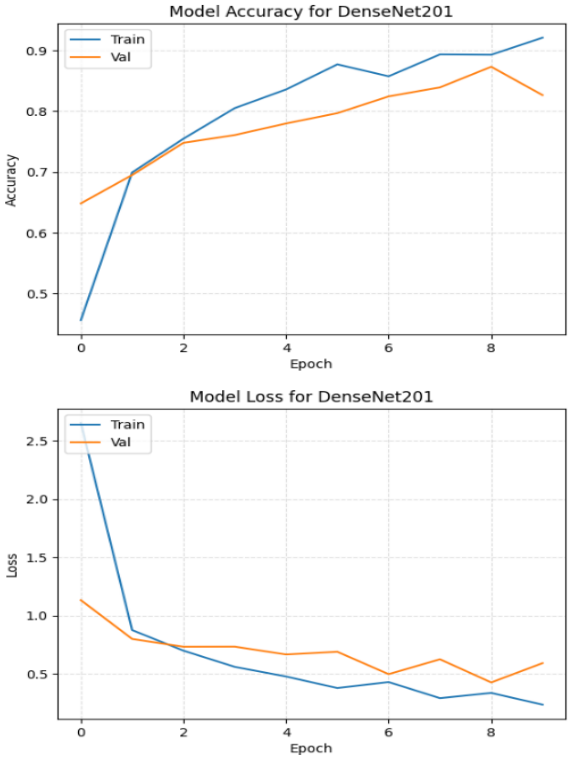

Fig. 14. Accuracy andLoss v Epoch on Training set with DenseNet201

V. Conclusion & Future Work

All the above explored models show great promise for classifying images in our Tumor brain scan dataset, however of them all the one that shows the greatest results is the VGG-19 model with 92% accuracy. This is most likely due to its architecture being much deeper and more complex than the other models compared to it in this paper. Typically, along with a greater complexity comes performance, however it also comes with more hardware demands. If hardware was a resource not readily available to the user, then VGG-19 would yield the greatest results as it is performance demanding. Luckily for us, in this study we used a PC with ample resources, allowing us to test each model to their fullest capacity. Perhaps for future works, increasing the epochs to 100 epochs would increase the performance of our models, as no signs of overfitting occurred during training and testing of the models, however we say fit to do 10 epochs due to time constraints.

Another improvement which could be made is the equal distribution of classes in our dataset. We already reduced/simplified the classes from 44 to 7, which in theory should have helped greatly, however even with this reduction the number of images per dataset was greatly disproportionate (522 images in the Normal set, while only 61 in the Ganglioma set). By better distributing the files in the classes, we may reduce Bias, and thus improve the performance of our models. This may be obtainable by referencing multiple datasets online, or collecting real world data and incorporating it into our model's training.

References

[1] R. Z. Nogueira and J. A. dos Santos, "Transfer Learning for Brain Tumor Segmentation," arXiv preprint arXiv:1811.02629, 2018.

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.

[3] L. Kaur and P. K. Jha, "A Survey of Waste Classification using Deep Learning Techniques," in 2019 5th International Conference on Computing Sciences (ICCS), 2019, pp. 181-185.

[4] Y. Liang, Y. Lu, and Y. Zhao, "Garbage Classification Based on Deep Convolutional Neural Network," in 2019 6th International Conference on Systems and Informatics (ICSAI), 2019, pp. 2116-2120.

[5] A. G. Roy, S. Conjeti, S. Navab, and N. Wiest-Daesslé, "Efficient assembly of 3D CNNs for brain tumor segmentation using random skip connections and lattice sampling," in Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, 2018, pp. 710-718.

[6] R. R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618-626.

[7] F. Rad, "Brain Tumor MRI Images (44c)," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/fernando2rad/brain-tumor-mri-images-44c?resource=download. [Accessed: 18-Apr-2023].

[8] A. Mahajan et al., "Brain Tumor Classification Using VGG-19 and ResNet-50 CNN Models," in 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), 2021, pp. 377-382.

[9] N. Bera et al., "A Comparative Study on Brain Tumor Classification using VGG-16 and ResNet-50," in 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 315-319.

[10] S. G. Ajitha and S. S. Prabhu, "Performance Comparison of Inception-V3 and Xception CNN Architectures for Brain Tumor Classification," in 2021 IEEE 7th International Conference on Distributed Computing and Internet Technology (ICDCIT), 2021, pp. 91-97.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700-4708.