

BRAIN TUMOR CLASSIFICATION

(PART 1 - IMPLEMENTATION)

Team Members (Group 3)

1. Elyas Irankhah
2. Jash Gandhi
3. Naga Sai Tejaswi Gandu
4. Rudram Vyas

Link to GitHub Repository: [Brain Tumor Classification](#)

1. ABOUT THE PROJECT:

A brain tumor is an abnormal growth of cells in the brain or near it. Brain tumors can be malignant or benign or in simple words you can say Cancerous or Non Cancerous. They can be of different size, shape and can occur in different parts of the brain. To be specific, They can occur in the brain tissue or near the brain tissue.

We chose this topic because brain tumors are a critical health concern affecting millions worldwide. The stats also say that it affects millions worldwide every year, especially children and people belonging to younger age groups. By leveraging CNNs and deep learning techniques, Our goal of this project is to improve the accuracy of diagnosis, reducing human dependency, potentially helping the patients through earlier detection and more accurate treatments.

2. DATASET AND DATA PREPROCESSING:

A. DATASET:

We have used a brain tumor MRI dataset openly available on Kaggle. Link below:

<https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset> The dataset has 7022 images belonging to 4 different classes. These 4 classes are notumor, glioma, pituitary and meningioma. The dataset is already separated into training and test sets so we did not have to perform that step in preprocessing. The training set has 5712 images and the test set has 1311 images. The dataset is also further separated into different classes which means in both training and test folders we have further 4 subfolders for 4 different classes in the dataset.

B. DATA PREPROCESSING OVERVIEW:

Image preprocessing ensures us that all images are uniform in a way that they all follow a specific standard and that they're nearly of the same size, shape, and have the same light,

brightness conditions. This helps us in decreasing the training time of the model while also increasing the time when it comes to detection/prediction.

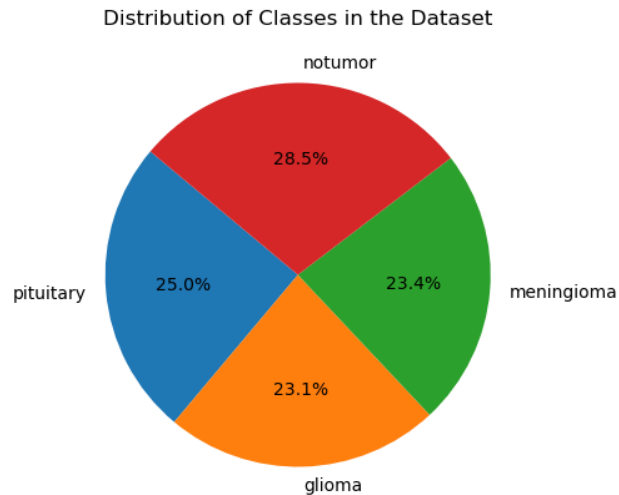
We've performed the image resizing and normalization. By resizing images to a standard size, you ensure that all images fed into the network have the same dimensions. When working with image data, it's important to normalize the pixel values to have a consistent brightness and improve contrast. This makes the images more suitable for analysis and allows machine learning models to learn patterns independent of lighting and different contrast conditions.

These are sort of the main methods we use here. There is also some other functions we use for our convenience and better visualizing the dataset. We have explained them in depth, why we implemented them and for what reason in the Implementation section of this Dataset and Data Preprocessing.

C. DATA PREPROCESSING IMPLEMENTATION:

1. `load_images` function:
 - a. This function is used to load and preprocess the images.
 - b. It takes the path to an image folder as its parameter.
 - c. It goes over each image in the directories and converts them to Grayscale, Transforms them to 224x224 dimensions, and normalizes the pixel values between 0 and 1.
2. We also get the different image class labels present and combine them and extract the unique class labels which are 4 for this database.
3. We then plot a few random images in a 3x3 grid from the training set and each image is displayed with its own class label.
4. We then calculate the count of each class in the combined training + test dataset and create a pie chart to show the distribution of each class across the dataset.

a. Pie Chart:



5. `load_images_and_sizes` function:

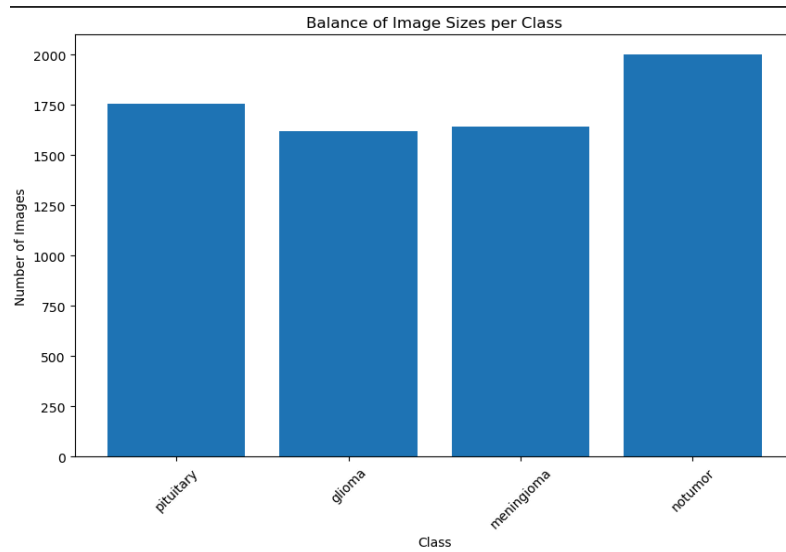
- a. This function is quite similar to `load_images` folder because it loads the images from the folder path. However, this function also keeps track of the original image sizes for each image class.
- b. It stores each original size as per the different classes.
- c. It returns the image data array, class labels and original image sizes which we combine in the next step, loop over and display size of each image as per different image classes.

6. In an intermediary step, we verify that all images have been transformed to 224x224 size.

7. `count_image_sizes` function:

- a. This function takes in image sizes and counts the occurrences of each unique size for each image class.
- b. It returns `size_counts_per_class` which is nothing but image sizes mapped to their class labels
- c. In the subsequent steps, we combine the size counts from both training and test sets and create a Bar graph to visualize the balance of image sizes per class in the combined training + test dataset. The x-axis represents the classes and the y-axis represents the number of images per image class.

d. Bar Graph:



3. DenseNet:

A. INTRODUCTION

DenseNet, or Dense Convolutional Network, is a deep learning architecture that aims to solve the vanishing gradient problem and promote feature reuse in convolutional neural networks (CNNs). Unlike standard CNN designs, DenseNet uses dense connections between all levels inside a dense block. These links provide direct access to gradients throughout the network, improving information flow and minimizing the disappearing gradient issue. Also, DenseNet encourages feature reuse by concatenating feature maps from preceding layers, resulting in more efficient parameter consumption and better performance, particularly in applications with limited training data.

B. RELATED WORK

Before implementing DenseNet ourselves we did some Literature Review of existing work and how it has performed for others. In our case, two papers stood out the most. The first one is Yufan Zhou et al.'s study "Holistic Brain Tumor Screening and Classification Based on DenseNet and Recurrent Neural Network". It combines DenseNet and recurrent neural network (RNN) architectures. DenseNet is used to extract rich hierarchical features from medical imaging data, most likely MRI or CT scans, while the RNN component allows the model to successfully handle sequential information included in medical data.

The other one is Shakil Mahmud Shuvo et al.'s paper "Multi-class Brain Tumor Classification with DenseNet-Based Deep Learning Features and Ensemble of Machine Learning Approaches". This paper captures exact tumor characteristics by using DenseNet-based deep learning features collected from medical imaging data, most likely

MRI or CT scans. Also, the authors use a combination of machine learning algorithms to improve classification performance. These studies provided us with a good base for implementing our DesNet model. We also looked into some available codes that use DesNet in image classifications to understand different ways the DesNet model can be implemented depending on the dataset.

C. METHODOLOGY

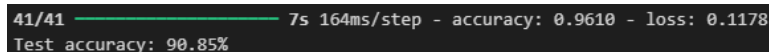
To implement DesNet for our project, we started by creating the required functions that is the `dense_block` and `transition_layer` functions. Within a dense block, a series of layers containing Batch Normalization, ReLU activation, and 3x3 convolutions are stacked. These layers are concatenated with the input tensor, enabling information flow from earlier layers to subsequent ones. Transition layers, on the other hand, reduce the number of feature maps and spatial dimensions using 1x1 convolutions for channel reduction followed by max pooling for downsampling.

The `create_densenet` method handles the DenseNet model's building. We initialize the input layer, add an initial convolutional layer, and then perform batch normalization and activation before repeatedly appending dense blocks and transition layers based on the parameters supplied. Finally, we use a global average pooling layer to minimize spatial dimensions and a dense output layer with softmax activation for classification. After model creation, it is compiled with the Adam optimizer and categorical cross-entropy loss function to prepare for training.

Then we come to the data preparation part. Here we process the images to prepare it for training and testing. We start by splitting the data into testing sets, then convert the labels in a one-hot encoded format and normalize the input images. Then we train the model using the training data with a validation split to check the performance during training. Training parameters such as batch size and number of epochs are specified to control the optimization process. In this project, we ran the DesNet training for 50 epochs with a batch size of 100.

D. RESULTS

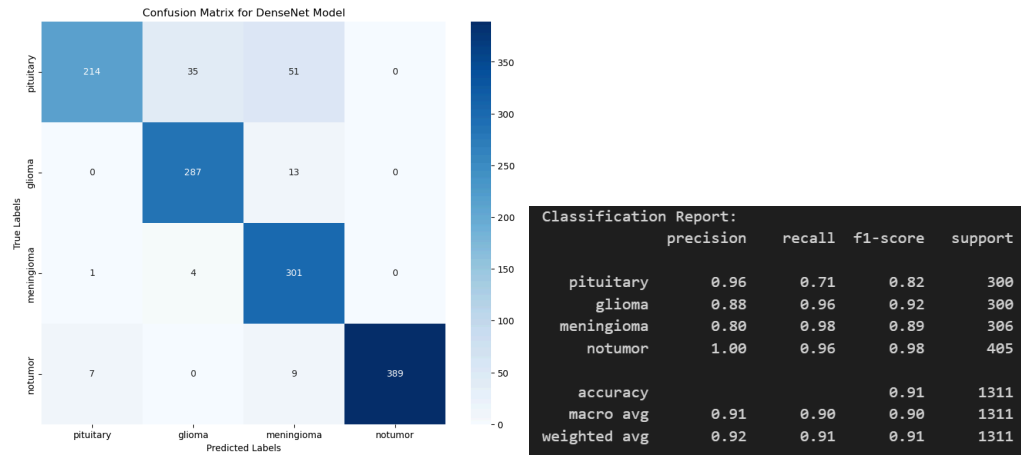
Our implementation was able to achieve 90.85% accuracy.



```
41/41 7s 164ms/step - accuracy: 0.9610 - loss: 0.1178
Test accuracy: 90.85%
```

Our model correctly classified 214 samples as pituitary tumors. However, there were 35 samples of pituitary tumors misclassified as gliomas and 51 misclassified as meningiomas. Gliomas were classified correctly for the most part, with 287 samples correctly identified. There were 13 samples misclassified as meningiomas. The model performed well in identifying meningiomas, with 301 samples correctly classified. There

was 1 sample misclassified as pituitary tumor and 4 samples misclassified as gliomas. The majority of no-tumor cases (389 samples) were correctly classified by the model. There were no misclassifications for this category.



4. VGGNet - 19:

A) INTRODUCTION:

VGG Net stands for Visual Geometry Group Network. It is a deep convolutional Neural Network design. It is one of the most often used image recognition architectures. There are two most popular VGGNet architectures, VGGNet 16 and VGGNet 19. In our project, we used VGGNet 19 architecture. 19 here means it has 19 convolutional layers. It has 16 Convolutional layers and 3 fully connected layers. One of which is softmax layer. Also, there are 5 max pooling layers. VGG19 takes 224x224x3 pixel image inputs where 224x224 represents the spatial dimensions (height and width) of the image, and 3 represents the number of color channels, which are usually RGB colors. The architecture consists of a series of convolutional blocks, each containing multiple convolutional layers followed by max-pooling layers, which extract features at different levels of abstraction. The Max pooling layers downsample the feature maps spatially, reducing their dimensions by half while retaining the important information like 224 to 112 and so on like in Figure 1. Each block in the architecture is labeled with its depth (number of filters) and the type of convolutional layers it contains. For example, in Figure 1. the first block has a depth of 64 and contains two 3x3 convolutional layers labeled as Conv1_1 and Conv1_2. And as we move towards right, the depth of the layers increases from 64 to 128, and so on. Each convolutional layer performs 3x3 convolutions with a stride of 1 and a padding of 1 to preserve spatial dimensions. The FCC in the Figure 1 are labelled as FC1 and FC2. Each of them has a hidden ReLU activation function. The sizes of the fully connected layers are indicated (4096 and 1000), which represent the number of neurons in each layer. The final layer of the VGG19 architecture is the softmax layer. It outputs the probability distribution over the output classes.

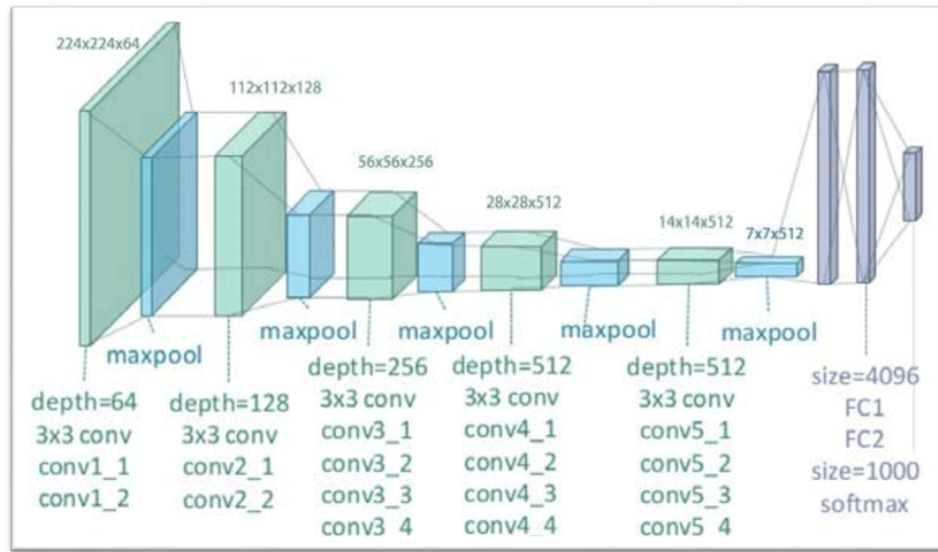


Figure 1

B) RELATED WORKS:

The related works I found interesting and helpful are [1] [2]. In paper [1], the study aimed to enhance brain tumor diagnostic prediction using ensemble deep learning models, combining CNN with other architectures for improved accuracy. Training with VGG19, Inception v3, and ResNet 10, a stacked ensemble achieved 96.6% accuracy for binary classification. Another paper, [2], introduces NeuroNet19, a deep neural network architecture for brain tumor classification, incorporating VGG19 with an Inverted Pyramid Pooling Module (iPPM) for multi-scale feature extraction. This NeuroNet19 model achieved accuracy of 99.3% which is outstanding. Both of them implemented individual models and then used ensemble learning to see the combined accuracy. In [1] they got an accuracy of 92% with 25 epochs with VGGNet-19 model. However, [2] mostly focussed on integrating VGGNet-19 with NeuroNet19 model. It was an interesting paper.

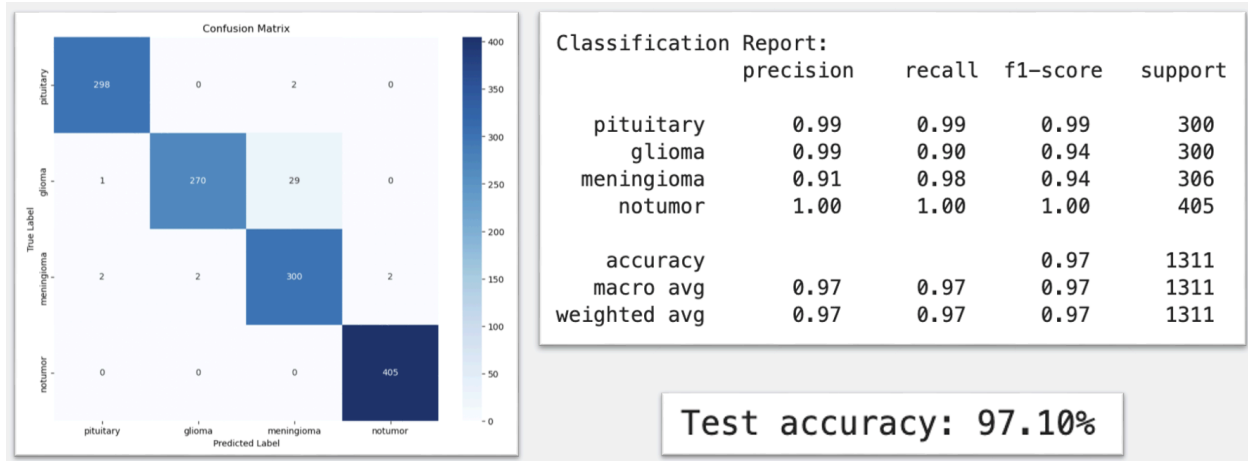
C) METHODOLOGY:

In this project, I focused on VGGNet-19 model implementation. Firstly, I started by loading the images for both training and testing datasets using a custom function called 'load_images'. This function reads the images from the specified image folder, converts them to grayscale, resizes them to a common size of 224x224 pixels, and stacks them to create a 3-channel representation suitable for input to the VGG19 model. Additionally, I performed normalization on the pixel values by scaling them to the range [0, 1]. Next, I created the VGG19 model architecture using the Keras library. The VGG19 model was initialized with pre-trained weights from the ImageNet dataset and configured to exclude the fully connected layers at the top of the network. I then added custom fully connected

layers on top of the VGG19 base model, including two dense layers with ReLU activation functions and a final dense layer with softmax activation to output class probabilities. After constructing the model, I compiled it using the Adam optimizer, which is known for its efficiency and effectiveness in training neural networks. As this is a multi-class classification task, I chose categorical cross-entropy as the loss function, which is well-suited for optimizing models for multi-class classification problems. Additionally, I used accuracy as the evaluation metric to monitor the performance of the model during training. With the model architecture defined and compiled, I proceeded to train the model on the training dataset using the 'fit' method. The training process involved feeding batches of preprocessed images through the model, computing the loss, and updating the model parameters using backpropagation and the Adam optimizer. I trained the model for a specified number of epochs, allowing it to learn and optimize its parameters on the training data. Once the training process was complete, I evaluated the performance of the trained model on the testing dataset using the 'evaluate' method. This computed the test loss and accuracy of the model on unseen data, providing an objective measure of its generalization performance. Finally, I generated predictions for the testing dataset using the trained model to classify the brain tumor images. I computed classification metrics such as the confusion matrix and classification report using functions from the scikit-learn library to assess the model's performance and visualize the results using the seaborn library.

D) RESULTS & DISCUSSION:

VGGNet-19 model was able to achieve an accuracy of 97.10%. Our model perfectly classified the 'notumor' class. It achieved higher accuracy than in study [1]. However, meningioma classification was not as accurate as the classification of the other classes but 91% is also decent enough. As far as VGG-19 implementation alone is taken into account, our goal is to get higher accuracy than the previous study. Main challenge I faced is doing the grayscaling part, i.e. converting to 3 channels. Also, going over more than 10 epochs were not showing satisfying results probably due to overfitting. Another challenge was with the computation time. It took about 2-3 hrs to run on GPU. Overcoming this can be added to future work.



5. ENSEMBLE LEARNING (DesNet + VGGNet - 19)

A) INTRODUCTION

In machine learning, ensemble methods improve accuracy and reliability by combining different models. This project takes advantage of this approach by bringing together two sophisticated neural networks, DenseNet and VGG19. Each network has its own strengths in analyzing and extracting details from images, which makes them great for ensemble strategies. By merging the capabilities of DenseNet and VGG19, the goal is to build a model that excels at identifying patterns in images and performs well on new, unseen data.

B) RELATED WORKS

In the literature, there is a notable study [3] that presents a blend of deep learning and traditional statistical methods, utilizing architectures like VGG and SVM alongside AlexNet, to improve breast cancer detection. Another significant work introduces a hybrid neural network that merges various deep learning structures, specifically highlighting the combination of VGG19 and DenseNet, which has achieved up to 97% accuracy in identifying brain tumors. These scholarly contributions point out the growing efficacy of machine learning techniques in refining the accuracy of cancer diagnostics [4][5].

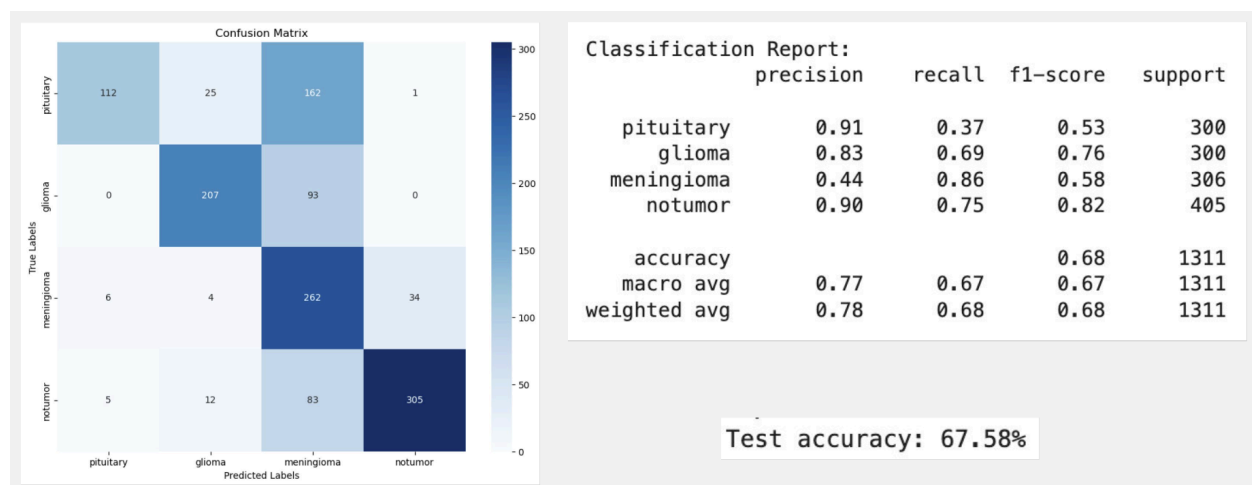
C) METHODOLOGY

I focused on an ensemble learning strategy by creating a combined model that integrates DenseNet and VGG19 neural networks. Both of these networks are well-regarded for their image recognition capabilities. DenseNet is known for its efficiency in connecting layers to maximize information flow, while VGG19 is celebrated for its depth and use of very small convolutional filters, which can capture fine details in images. The combined model essentially takes the best features from both networks, feeding image data through each one separately and then merging the results. This merging is thought to capture a richer set of features than either network could manage alone, potentially leading to more accurate image classification. I then trained and evaluated this model using my image

dataset, seeking to understand how well the ensemble could perform in identifying specific categories compared to the individual models.

D) RESULT & DISCUSSION

In this section of our study, we encountered challenges with the classification of the Meningioma class. The ensemble model exhibited a high sensitivity to this category, as evidenced by the classification report and the confusion matrix. It appears that the model may be prone to misclassifying instances of Meningioma, which warrants further investigation into the causes of this issue. We reflected on the performance disparities between the individual models and the ensemble approach. DenseNet achieved a 91% accuracy, while VGG-19 reached 97%. However, the ensemble model combining these architectures resulted in a lower accuracy of 67%. This reduction in performance could be attributed to the increased complexity of the model, potential overfitting, or the dynamics of how the training data was processed. Notably, class imbalance might also play a significant role. Additional measures such as Grad-CAM, data augmentation techniques, and the YOLO-7 algorithm were implemented as supplementary efforts to improve the model's accuracy and robustness. Despite the high accuracies achieved by the individual models, the ensemble model's sensitivity to the Meningioma class suggests that a more nuanced approach may be necessary to handle such complexities effectively.



E) LIMITATION & FUTURE WORK

Our report acknowledges the constraints of our current study and proposes directions for subsequent research. A notable limitation is the substantial computational time required for model training approximately 6 hours on a GPU and up to 12 hours on a CPU. This emphasizes the need for more efficient computing resources or optimization of the code to reduce runtime. Looking ahead, we aim to enhance the performance of our ensemble model. We will explore advanced ensemble techniques to leverage the full potential of the model's accuracy. Additionally, refining our data augmentation strategies and

implementing early stopping could further improve the model's generalization capabilities and prevent overfitting. These improvements are crucial for advancing our model's utility in clinical settings, where accurate and timely diagnosis is paramount. Our report acknowledges the constraints of our current study and proposes directions for subsequent research. A notable limitation is the substantial computational time required for model training approximately 6 hours on a GPU and up to 12 hours on a CPU. This emphasizes the need for more efficient computing resources or optimization of the code to reduce runtime. Looking ahead, we aim to enhance the performance of our ensemble model. We will explore advanced ensemble techniques to elevate the full potential of the model's accuracy. Additionally, refining our data augmentation strategies and implementing early stopping could further improve the model's generalization capabilities and prevent overfitting.

REFERENCES:

- [1] Al-Azzwi, Z., & Nazarov, A. N. (2023). Brain tumor classification based on improved Stacked Ensemble Deep Learning Methods. *Asian Pacific Journal of Cancer Prevention*, 24(6), 2141–2148. doi:10.31557/apjcp.2023.24.6.2141
- [2] Haque, R., Hassan, Md. M., Bairagi, A. K., & Shariful Islam, S. M. (2024). Neuronet19: An explainable deep neural network model for the classification of brain tumors using magnetic resonance imaging data. *Scientific Reports*, 14(1). doi:10.1038/s41598-024-51867-1
- [3] Devi, M. R., Sainath, S., & Pappula, P. (2022, January). Brain tumor detection using hybrid neural network based on VGGNet-19 and DenseNet. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 1775-1780). IEEE. doi: 10.1109/ICSSIT53264.2022.9716422
- [4] VigneshKumar, M. K., & Sumathi, N. (2021). A Hybrid Deep Learning Approach for Breast Cancer Detection Using VGG-19 and Support Vector Machine. *International Journal of Mechanical Engineering*.
- [5] Belaid, O. N., & Loudini, M. (2020). Classification of brain tumor by combination of pre-trained vgg16 cnn. *Journal of Information Technology Management*, 12(2), 13-25. https://jitm.ut.ac.ir/article_75788.html
- [6] Zhou, Y. et al. (2019). Holistic Brain Tumor Screening and Classification Based on DenseNet and Recurrent Neural Network. In: Crimi, A., Bakas, S., Kuijf, H., Keyvan, F., Reyes, M., van Walsum, T. (eds) *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. BrainLes 2018. Lecture Notes in Computer Science()*, vol 11383. Springer, Cham. https://doi.org/10.1007/978-3-030-11723-8_21

[7] Özkaraca, O., Bağrıaçık, O. İ., Gürüler, H., Khan, F., Hussain, J., Khan, J., & Laila, U. E. (2023). Multiple Brain Tumor Classification with Dense CNN Architecture Using Brain MRI Images. Life (Basel, Switzerland), 13(2), 349. <https://doi.org/10.3390/life13020349>

CODE REFERENCES:

[1] <https://www.kaggle.com/code/kveesh/brain-tumor-type-classification-vgg-19>

[2] <https://www.kaggle.com/code/suryaprakashkalyanam/densnet-121>