Brain Tumor Classification

Atharva Kshirsagar, Manichandra Sripaad Nemmikanti

Introduction

Brain tumors are a serious medical condition that affect a large number of people worldwide. Accurate and timely diagnosis of brain tumors is critical for effective treatment and management of the disease. Magnetic resonance imaging (MRI) is a commonly used imaging technique for the detection and classification of brain tumors. However, the analysis of MRI images can be challenging due to the complexity of the images and the variability of tumor characteristics. In this project, we explore the use of deep learning models for the classification of brain tumors using MRI images. Specifically, we investigate the performance of four different models: LeNet5, VGG16, VGG19, and ResNet50.

The main objective of our project is to compare the performance of these models and identify the most effective and accurate approach for brain tumor classification using MRI images. Through this project, we demonstrate the importance of data exploration, preprocessing in the development of deep learning models for medical imaging analysis. We also highlight the need for careful model selection, performance validation and fine tuning to ensure that the models are effective and reliable.

Data Exploration

The dataset used in this project consists of MRI images of brain tumors collected from Kaggle. The dataset contains a total of 7033 images, which are classified into four different types of brain tumors: glioma, meningioma, pituitary tumor, and no tumor. There are 1321 images of glioma, 1339 images of meningioma, 1605 images of no tumor, and 1457 images of pituitary tumor in the training set. In the test set, there are 300 images of glioma, 306 images of meningioma, 405 images of no tumor, and 300 images of pituitary tumor. The dataset is split into training and test sets, with 5722 images (81.36%) in the training set and 1311 images (18.64%) in the test set. To better visualize the class distribution, we plotted a pie chart showing the proportion of images in each set (Figure 1). The pie chart indicates that the training set contains the majority of the images, while the test set is relatively smaller.

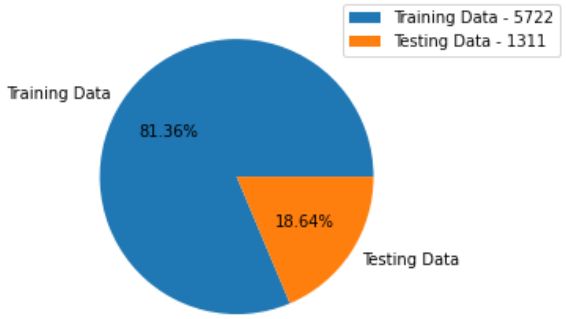

Figure 1 - Distribution of training and test datasets

To better understand the characteristics of the images, we also visualized some sample images from each class (Figure 2). The images appear to have varying levels of brightness and contrast, and some have visible abnormalities such as lesions or masses. The images also differ in size, with some being smaller and others larger. These differences are handled during the preprocessing to avoid the effect on the performance of the models.

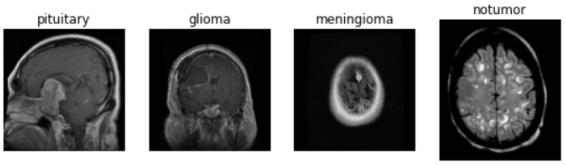

Figure 2 - MRI Data Samples from each class

To further analyze the distribution of classes in the dataset, we plotted histograms showing the number of samples in each class for both the training and test sets (Figure 3). The histograms show that the classes are relatively balanced in both sets, with no significant skewness towards any particular class. The blue bars represent the training set, while the orange bars represent the test set.

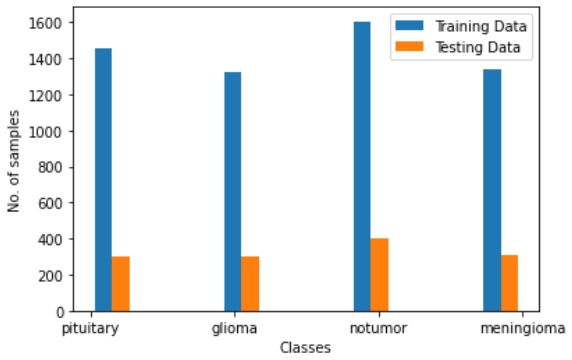

Figure 3 - No. of samples in each class

These insights provide the characteristics of the dataset that guides the preprocessing and model selection to ensure that our models are trained on relevant and representative data.

Data Preprocessing

Data preprocessing is one of the crucial steps as it can greatly impact the performance of the model. In our project, we applied a number of preprocessing techniques to our image dataset before feeding it to our model. The first step in our preprocessing pipeline is to convert the JPEG image to PIL image using functions in Keras library. This PIL format will then be used to increase the brightness and quality of the images. We accomplished this using the image enhance functions, which takes the PIL format image as input and returns the processed image in PIL format. We increased the brightness of the image by a factor of 1.5 using the ImageEnhance.Brightness from the PIL library. Then further, the contrast of the image is increased by a factor of 1.5 with the help of ImageEnhance.Contrast.

After applying these enhancements, we converted the images from PIL format to NumPy arrays containing the preprocessed images. These images can be seen in Figure 4, which displays the enhanced versions of the original images shown in Figure 2. The enhanced brightness and contrast can help the model better distinguish between different types of tumors, while the conversion to NumPy arrays makes the data easier to work with in the model. Additionally, we ensured that all images are of the same size (128 x 128) to maintain consistency in the input size of images for our models.

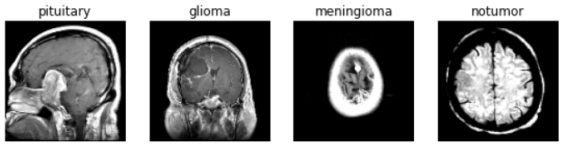

Figure 4 - Enhanced Data Samples from each class

As we are using pre-trained models like LeNet5, VGG16, VGG19, and ResNet50 for the classification, there is no need for feature engineering techniques. These models are already trained on a large dataset and have learned features that can be transferred to our task with little to no modification. Therefore, we focused on data preprocessing and fine tuning to standardize the input data and make it more consistent across different images, which can help to improve the performance of our models.

Model Exploration

As a preliminary step, we explored the data using the LeNet5 architecture. It is the classic CNN architecture that has been widely used for image classification. We experimented with LeNet5 as a simple baseline model to compare the results with the more complex models. Apart from LeNet5, we used the advanced pre-trained models like VGG16, VGG19, and ResNet50 for our final classification task. These models were chosen due to their popularity in image classification tasks and their availability through the Keras library. Each model was initialized with pre-trained weights on the ImageNet dataset and the final classification layer was replaced with a dense layer with a softmax activation function. We froze all layers of the models except for the last few layers to prevent overfitting and to speed up the training process.

We trained each model on the pre-processed image data and compared their performance using metrics such as sparse categorical accuracy and validation accuracy. Additionally, we used techniques such as cross-validation to prevent overfitting and to optimize

the models. We also played around with different hyperparameters like learning rate and weight-decay for regularization. The results of our model exploration showed that VGG16 and VGG19 had similar performance without regularization, but VGG19 was overfitting more on the dataset. ResNet50 had the maximum overfitting.

Overall, our model exploration process allowed us to select the best-performing model for our image classification task and to gain insights into the strengths and weaknesses of different pre-trained models.

Performance Validation

We tested the different models with different choices of hyperparameters and different architectures while fine-tuning the model. We also adapted the model to accommodate for under-fitting and overfitting. The following results depicted using the validation accuracy and loss against the no. of epochs show the performance of models after regularization.

1. LeNet5

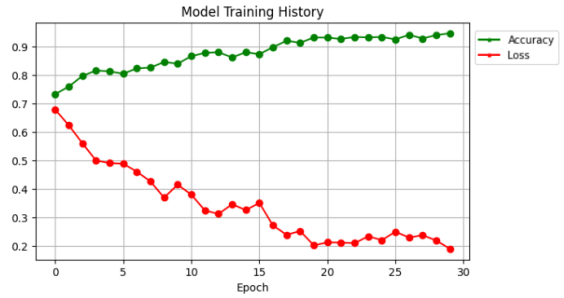

Figure 5: Validation Accuracy Vs Loss for LeNet

2. VGG16

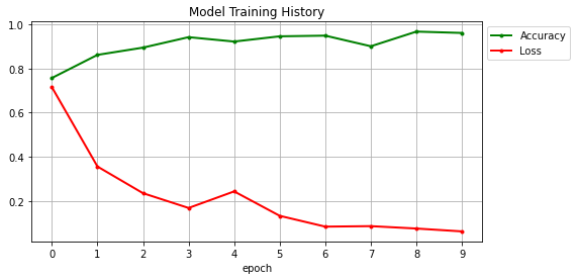

Figure 6: Validation Accuracy Vs Loss for VGG16

3. VGG19

Figure 7: Validation Accuracy Vs Loss for VGG19

4. ResNet50

Figure 8: Validation Accuracy Vs Loss for ResNet50

Dealing with under- and over-fitting

Initially, we ran the models without any form of regularization. We found during model exploration that models VGG16 and VGG19 gave similar training accuracies, but there was a lot of overfitting in these models. Furthermore, the LeNet5 model was overfitting the most without regularization.

Model Name	Training Accuracy	Validation Accuracy
LeNet5	100.00	94.66
VGG16	97.50	93.52
VGG19	96.50	86.80
ResNet50	92.16	90.60

*Table 1: The results **before** Regularization*

So, in order to deal with overfitting, we introduced Dropout regularization to deal with this issue. The results after regularization overcame overfitting such that the training and validation accuracies were almost similar.

Model Name	Training Accuracy	Validation Accuracy
LeNet5	99.11	94.89
VGG16	98.12	96.03
VGG19	91.81	86.80
ResNet50	89.23	87.72

*Table 2: The results **after** Regularization*

Inference

After exhaustive model exploration and hyperparameter-tuning, we can infer that the model based on **VGG16** performs better than all the different models tested with. Although VGG 19 performs well in the training dataset, the validation accuracy even after regularization was not satisfactory. LeNet5 model overfits on the training dataset leaving a huge percentage gap between training and validation accuracies but still overfits when regularization is introduced in our model. Hence, LeNet5 might not be the best choice of model for our dataset. The ResNet50 model copes well with overfitting but underfits on the data as compared to other model accuracies. The maximum ResNet50 achieves is **92.60%** but after regularization drops to **89.23%**.

To sum up, VGG16 along with hyperparameter-tuning and regularization is the model best-suited for our dataset. The VGG16 model does fairly well without regularization by achieving a high training accuracy but suffers from overfitting as seen in the validation dataset. But after regularization, the VGG16 model perfectly adapts to the training data as well as validation dataset, decreasing overfitting and increasing validation accuracy to **96.03%** which is the highest achieved so far.

Hence, we can experimentally conclude that the VGG16 model along with hyperparameter tuning and regularization achieves the highest accuracy of **96.03%** for our Brain Tumor dataset.

Contributions

Atharva and Manichandra Sripaad together formulated the main idea of the project and started researching ways to go about it by reading documentations and exploring datasets. After the dataset was selected, both of them started with the data exploration part of the project. Atharva worked on preprocessing the dataset such that it works well with our model and Manichandra Sripaad visualized the preprocessed output and made sure that the input images were ready to go through our models. Atharva and Manichandra Sripaad both worked on different models and focused towards fine tuning the model for achieving maximum accuracy. The documentation part was also done in collaboration by both the authors and successfully displayed the results and findings of our project and the use of CNNs in the medical industry, in this specific case, identifying brain tumors from patient MRI images.

References

- https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset?select=Training
- https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection
- https://keras.io/api/applications/vgg/
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9468505/
- https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-023-02114-6
- https://pytorch.org/
- https://matplotlib.org/
- https://www.kaggle.com/code/blurredmachine/lenet-architecture-a-complete-guide