School of Physics,
Engineering and
Computer Science

MSc Data Science Project

7PAM2002-0509-2023

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Classification of Brain Tumor Using MRI Images and Artificial Intelligence

Student Name and SRN:

Sehar Fatima and 22089093

Supervisor: Severin Bunk

Date Submitted:

Word Count:

DECLARATION STATEMENT

This report is submitted in partial fulfillment of the requirement for the Master of Science in Data Science degree at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct, and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7, and UPR AS/C/5, section 3.6). I have not used chatGPT or any other generative AI tool to write the report or code (other than where declared or referenced).

I did not use human participants or a survey for my MSc Project.

I now give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed:

Student Name Signature:

Student SRN number:

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Acknowledgment

As I finish my post-graduate studies, I emphasize that it has been an incredible learning experience. I want to express my gratitude to all those who have supported me. I want to start by expressing my gratitude to Almighty God for never ceasing to inspire me with His endless blessings and giving me the confidence and bravery to move forward with assurance and self-belief.

I want to convey my appreciation and gratitude to Severin Bunk, my supervisor, for her constant advice and assistance in this project. I appreciate her continuous support and her patience with my curiosity.

I would also like to express my gratitude to all my professors at the University of Hertfordshire, who helped me gain knowledge and understanding of the subjects and helped me throughout my course. I would also like to thank my parents, sister, and friends for their unwavering encouragement and support, without which this would not have been possible.

Abstract

Accurate classification of brain tumors is crucial for effective treatment planning and improved patient outcomes. However, the subtle variations in MRI images can limit traditional diagnostic methods relying on expert radiologists, leading to difficulties differentiating between tumor types. This study aims to develop and evaluate various machine learning models for classifying brain tumors using MRI images, offering a supplementary tool to assist medical professionals in making precise diagnoses. We explored multiple deep learning architectures, including Convolutional Neural Networks (CNN), CNN with an attention layer, ResNet50, MobileNetV2, and DenseNet121, to determine their effectiveness in distinguishing between glioma, meningioma, pituitary tumor, and nontumor cases. The dataset utilized in this research comprised 7,023 MRI images divided into training and testing sets. Each model was trained using a systematic approach involving data preprocessing techniques such as resizing and normalization. The models were evaluated based on key performance metrics, including accuracy, precision, recall, F1-score, and confusion matrices, to determine their ability to classify tumors accurately. Our findings indicate that the CNN model with an attention layer demonstrated superior performance, achieving a smoother increase in accuracy and better generalization than other models, achieving 98 percent accuracy. This study contributes to the growing body of knowledge in medical imaging by validating the application of advanced deep learning techniques for brain tumor classification, ultimately paving the way for more accurate and efficient diagnostic tools in clinical settings.

Keywords: Brain Tumor Classification, MRI Images, Deep Learning, Attention Mechanism

Contents

1 Introduction

1.1 Background of the study

Brain tumors have remained an essential part of medical research for a long time due to the severe consequences that these conditions can have on patients' health and quality of life. Traditionally, the classification of brain tumors has been done through imaging techniques such as Magnetic Resonance Imaging (MRI) in combination with histopathological analysis. However, these traditional procedures possess several shortcomings; for instance, they may be invasive, people interpret them differently, and they take time to analyze manually with the rise of intense machine learning, which has opened new avenues for improving the accuracy and efficiency of tumor types classification. Deep learning models, specifically CNNs, can process large volumes of image data and learn from them, identifying patterns that may not be visible using manual visualizations. Recent studies have illustrated the effectiveness of these models in various medical applications, including differentiating between different types of brain tumors. Nonetheless, their performance varies significantly depending on the model architecture employed and dataset quality, among other things particular to tumors being categorized. This work is premised on existing knowledge by assessing how well several deep learning architectures perform regarding brain tumor classification.

Deep learning, which mimics human brain neural networks, has led to remarkable changes in medical image analysis(Puttagunta & Ravi, 2021). In the context of several medical applications, brain tumor classification is an essential area of concern because of its life-threatening nature and difficulty in diagnosis. There are very significant differences in the heterogeneous appearance and growth patterns, plus effective influence on the brain, making it a complicated matter to classify them accurately. Traditional tumor classification methods mainly depend on radiologists' manual assessment, which leads to time-consuming work processes and variability. Automated classification systems using CNNs are highly promising because they can learn complex patterns and characteristics from large quantities of medical images. This study aimed to investigate the efficacy of several advanced deep learning models in classifying brain tumors while evaluating their performance to determine reliable and efficient means. It proposed a generalized model that doctors and patients can use to predict the type of brain tumor. By addressing these challenges regarding brain tumor classification, this research contributes towards improving accuracy in diagnosis and patient outcomes within neuro-oncology.

Brain tumor categorization is an essential aspect of medical diagnosis as it helps to determine the fate of patients through treatment planning and prognostic evaluation. Recently, the emergence of deep learning has brought about a revolution in the medical field concerning imaging analysis, which seeks to improve brain tumor precision. The research considers different deep learning models such as CNN, ResNet50, MobileNetV2, DenseNet121, and CNN with attention mechanisms applied to classify brain tumors into glioma, meningioma, non-tumor, and pituitary tumor categories. It aims to analyze different model performances and determine the best way to accurately and reliably classify tumors. This study will then add to existing knowledge by highlighting their strengths and weaknesses, particularly in handling complex heterogeneous tumor types.

1.2 Problem Statement

This research aims to tackle the big issue of correctly distinguishing tumors in the brain through the analysis of MRI images, which is exceptionally crucial in treatment planning because these conditions can be fatal. Typical diagnostic techniques that greatly rely on expert radiologists may falter due to subtle individual differences among the MRI images revealing various types of tumors, causing possible misclassifications and less effective treatment. There is a strong justification for this research since it could substantially change medicine when detecting brain tumors more accurately and faster. However, machine learning models have not been fully explored and validated in regard to the classification of brain tumors, even though they have shown promising results in different medical imaging tasks. To develop and assess these models, this research intends to provide an additional element aiding the current diagnostic procedure to enhance patient outcomes through better clinical decision-making.

1.3 Research objective

1.3.1 General Objective

The primary objective of this research is to create and evaluate machine learning models that can classify brain tumors based on MRI images to enhance diagnostic accuracy and aid healthcare decisions.

1.3.2 Specific Objectives

- To collect and preprocess MRI images of brain tumors to make them suitable for training machine learning models.
- To perform feature engineering on the image dataset, extracting relevant features that enhance the model's ability to distinguish between different tumor types.
- Train and compare different deep learning models for brain tumor classification, evaluating them using various metrics.
- Identify the best model among those tested for detecting brain tumors, which might serve as a future guide for clinical practice.

1.4 Research Questions

This study aims to answer a research question: How can machine learning models utilize MRI images to classify brain tumors precisely? This is a crucial question regarding the potential of machine learning models as reliable diagnostic tools for brain tumors, with particular attention to their ability to sort out various brain tumors using MRI data.

In support of this primary research question, here are some sub-questions:

1. Do machine learning models use MRI pictures to identify brain cancers?
2. How well are artificial intelligence (AI) systems at recognizing and identifying different types of brain tumors from magnetic resonance imaging?
3. How do various machine learning models compare accuracy in classifying brain tumors using MRI images?

1.5 Significance of the study

This study has medical imaging and machine learning knowledge, focusing on using MRI images to classify brain tumors through machine learning model applications. It evaluates various models' effectiveness, strengths, and limitations in this medical scenario. These research results could be a basis for further investigations that could pave the way for more sophisticated diagnostic technologies.

From a practical point of view, it is strongly linked to the healthcare sector as it has profound implications. By pointing out how machine learning models support accurate brain tumor classification, it enhances diagnostic precision, which is imperative for effective patient management and outcomes. Radiologists could incorporate these newly developed models into their daily operations as they would help them expand their diagnosis abilities while at the same time decreasing the chances of misclassification issues.

1.6 Scope of the research

The extent of this investigation is only oriented toward how machine learning methods can be used to identify different brain tumors by using MRI images. This involves gathering and preprocessing MRI datasets before developing multiple machine learning models for training purposes and evaluating their accuracies in classifying various types of brain tumors into specific categories. It is confined to such forms of brain tumors as glioma, meningioma, nontumor, and pituitary, which are the critical categories under consideration in this study.

1.7 Thesis Organization

The thesis has been divided into four chapters. The first chapter introduces the research topic, discusses its background, states its objectives and problems, setting a base for the study. The second chapter gives an extensive literature review of previous works on brain tumor classification using MRI images, bringing out areas this thesis intends to fill with gaps. The third chapter elaborates on research design, including data collection, preprocessing, and applied machine learning models and evaluation metrics. The result presentation follows this in chapter four, where model performances are outlined with several metrics and visualization techniques employed. Finally, the last section discusses findings within which interpretation of results about research questions are given. The study's weaknesses are addressed while recommendations for future research are made.

2 Literature Review

2.1 Artificial intelligence in healthcare

This research article (Alanazi, 2022) examines the growing role of Machine Learning (ML) in healthcare, emphasizing its potential to enhance medical outcomes through advanced computational power and big data. The article distinguishes between supervised learning, which uses algorithms like linear regression, support vector machines, and decision trees to predict labeled data, and unsupervised learning, which identifies patterns in unlabeled data for applications such as fraud detection. Clinical applications of ML include developing decision support systems and predicting high-risk populations for targeted public health interventions. The article also highlights the importance of integrating ML concepts into medical education to enable health professionals to effectively interpret and guide research in this field.

This research (Callahan and Shah, 2017) explores the application of Machine Learning (ML) to electronic health records (EHRs), highlighting its potential to enhance patient risk scoring, predict disease onset, and improve hospital operations. It reviews current ML uses in clinical settings and contrasts these with traditional analysis methods, noting ML's advantages. The chapter also addresses the methodological and operational challenges of implementing ML in healthcare research and practice. Finally, it provides insights into future areas where ML could profoundly influence health and healthcare delivery.

This research (Nayyar et al., 2021) reviews the application of machine learning (ML) technology in healthcare, emphasizing its role in enhancing, rather than replacing, human physicians. ML is highlighted as a critical tool for developing computational approaches that offer solutions to complex healthcare problems. The chapter examines recent literature on ML's use in diagnosis, prognosis, and treatment planning, noting how it can provide faster and more accurate solutions for medical practitioners. While acknowledging the limitations and challenges of ML in healthcare, such as the evolving nature of medical science and technology, it discusses the opportunities ML presents for improving healthcare solutions. The chapter also stresses the importance of an interdisciplinary approach to fully leverage ML's potential in healthcare and aims to present novel and high-quality research facilitated by ML techniques.

2.2 Machine Learning in Brain Tumor Classification

This study (Sharif et al., 2022) introduces a new automated deep learning method for multiclass brain tumor classification, addressing the challenge of low accuracy often faced in medical imaging. The approach involves fine-tuning the DenseNet201 pre-trained model and using deep transfer learning on imbalanced data. Features are extracted from the average pooling layer. Still, since this alone isn't sufficient for accurate classification, two feature selection techniques are proposed: Entropy–Kurtosis-based High Feature Values (EKbHFV) and a modified genetic algorithm (MGA). These selected features are refined and fused using a non-redundant serial-based approach, followed by a multiclass SVM cubic classifier classification. The method was evaluated on the BRATS2018 and BRATS2019 datasets, achieving over 95% accuracy and demonstrating superiority to other neural networks.

This paper (Mohsen et al., 2018) explores the use of Deep Learning, particularly a Deep Neural Network (DNN) classifier, for classifying brain MRI images into four categories: standard, glioblastoma, sarcoma, and metastatic bronchogenic carcinoma tumors. The study leverages the discrete wavelet transform (DWT) for feature extraction and applies principal component analysis (PCA) for dimensionality reduction. The combined approach was found to perform well across various evaluation metrics, demonstrating the effectiveness of Deep Learning in addressing complex medical imaging problems.

This paper (Alqudah et al., 2020) discusses using Deep Learning, particularly a Convolutional Neural Network (CNN), for grading brain tumors based on T1-weighted contrast-enhanced MRI images. Brain tumor grading is crucial for developing effective treatment plans, as early detection and accurate classification can significantly impact patient outcomes. The study classifies brain tumors into three categories: Glioma, Meningioma, and Pituitary Tumor, using a dataset of 3064 images. The CNN model demonstrated high effectiveness, achieving an accuracy of 98.93% and a sensitivity of 98.18% for cropped lesion images. For uncropped lesions, the accuracy was 99% with a sensitivity of 98.52%, while segmented lesion images yielded an accuracy of 97.62% and a sensitivity of 97.40%. These results highlight the potential of CNNs as a powerful tool for brain tumor classification in medical imaging.

This paper addresses (Deepak and Ameer, 2019) the critical problem of brain tumor classification in computer-aided diagnosis (CAD) systems, specifically focusing on distinguishing between glioma, meningioma, and pituitary tumors. The proposed system utilizes deep transfer learning, leveraging a pre-trained GoogLeNet model to extract features from brain MRI images. These features are then classified using established classifier models. The study employs patient-level five-fold cross-validation on an MRI dataset, achieving a mean classification accuracy of 98%, surpassing other state-of-the-art methods. Additional performance metrics, including AUC, precision, recall, F-score, and specificity, are also reported. The paper highlights the effectiveness of transfer learning, mainly when training samples are limited, and provides a detailed analysis of misclassifications, emphasizing the practical implications of the research.

3 Research Methodology

3.1 Introduction to Methodology

This section provides insight into the methods used in this study for brain tumor classification with machine learning. The methodology explains the stepwise approach followed through data collection, preprocessing, and analysis of MRI images of brain tumors, as shown in the methodology diagram in Figure 3.1. It gives detailed information on the selection of machine learning models, their training and evaluation procedures, and metrics applied in assessing performance. The methodology selected is vital for ensuring that results are reliable and valid because it lays down the framework for this research. A systematic methodology aims to investigate how effective machine learning could be regarding brain tumor classification, thus providing insights into the medical imaging and diagnostic domain.

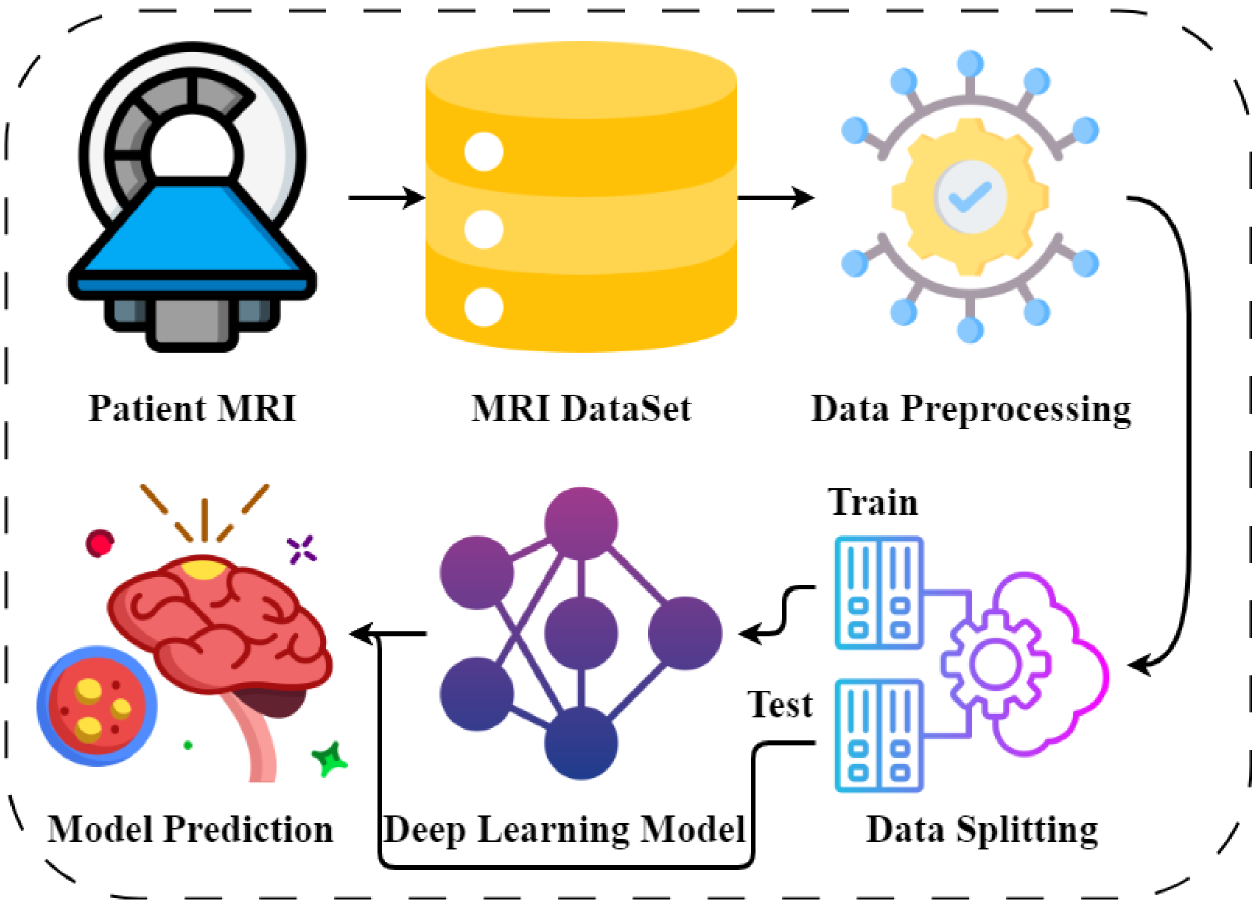


Figure 3.1 Proposed Methodology of Our Study

3.2 Data Collection

The data in this research comprises MRI images displaying brain tumors obtained from Kaggle (https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset). This dataset is made up of a total of 7023 MRI pictures, which cover a variety of types of brain tumors that include glioma, meningioma, and pituitary tumors, among others, as well as cases without actual tumors. The images are in RGB format; each tumor type has a corresponding label.

Data Exploration

The dataset included MRI scan images of brain tumors, classified into four different classes, which include normal brain tissue (nontumor), pituitary tumors, gliomas, and meningiomas. The training set consists of 5712 images, while the test set has 1311. The distribution by class is as follows: 2000 images for nontumor, 1757 images for pituitary tumors, 1621 images for glioma, and 1645 images for meningioma, as shown in Figure 3.2.

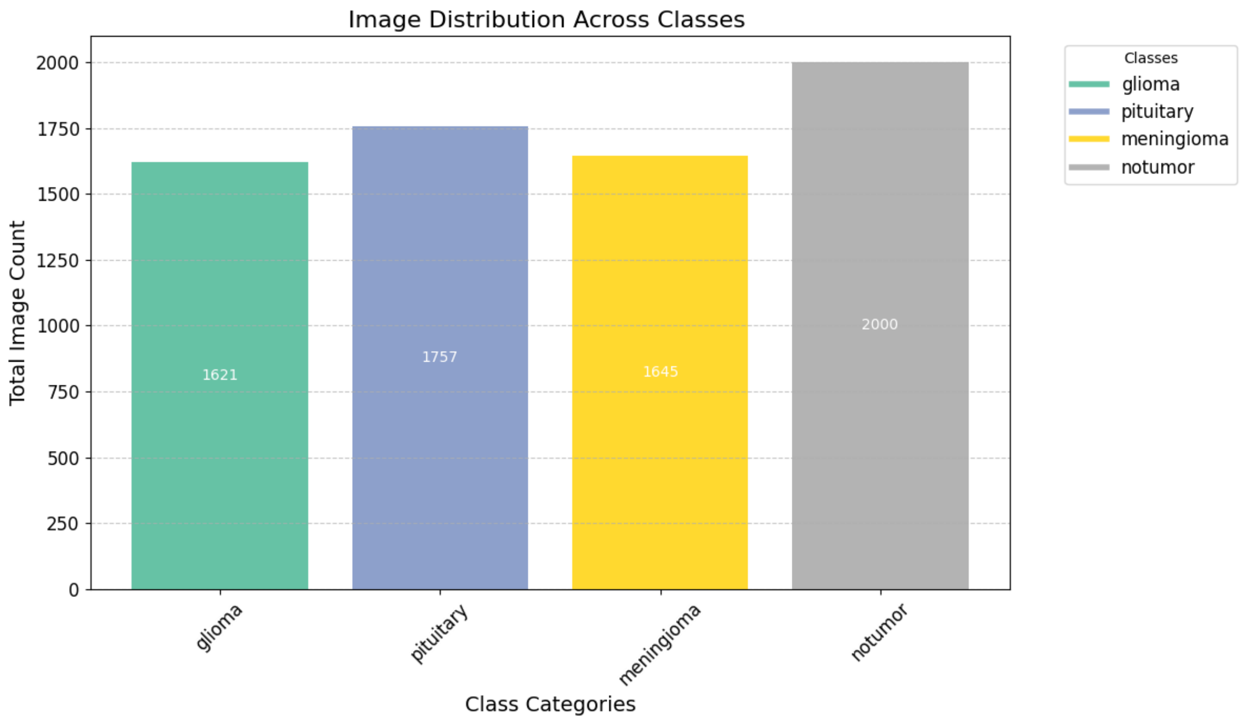


Figure 3.2 Image Distribution along Classes

Large data sets are used in the training stage to train any model successfully. This dataset provides model learning capabilities that allow them to perform well when applied elsewhere. In addition, this testing set contains 1311 images intended to determine how much progress has been made concerning the ability of models used in prediction when new data enters the system without it being expressed before.

3.3 Data Preprocessing

The primary data preprocessing step involved resizing all MRI images to have uniform dimensions across the entire dataset. Such an action was necessary to ensure homogeneity and compatibility for the input data fed into machine learning models. The resizes allowed us to standardize inputs, facilitating proper training of models and their evaluation. Additional processing mechanisms like normalizing or enhancing were not applied to the pictures. The aim was to ensure that all resized photos underwent similar processing procedures, leading to accurate classifications without untrustworthiness.

3.4 Deep Learning Models

This study implements a Multiple deep learning model, and all the models are defined individually.

3.4.1 Convolutional Neural Network (CNN)

This study uses a CNN to extract and learn features from MRI images to train a classifier for brain tumor classification. Its architecture consists of several convolutional layers followed by max-pooling and batch normalization. Specifically, the CNN model contains four convolutional layers with 32, 64, 128, and 128 filters, respectively. These are then followed by a max-pooling layer for downsampling the feature maps. After flattening the output from the final convolutional layer, a dropout layer with a rate of 0.5 is added to avoid overfitting. Then, there are two fully connected or dense layers. The last layer has an activation function with softmax for class probability output. Thus, the network architecture is defined by the following equations for each layer:

Convolution Layer:

$$\text{Output} = \text{Activation}\big(\text{Conv}(\text{Input}, \text{Filters}, \text{Kernel Size})\big)$$

Pooling Layer:

$$\text{Output} = \text{MaxPooling}(\text{Input}, \text{Pool Size})$$

Fully Connected Layer:

$$\text{Output} = \text{Activation}\big(\text{Dense}(\text{Input}, \text{Units})\big)$$

Dropout Layer:

$$\text{Output} = \text{Dropout}(\text{Input}, \text{Rate})$$

3.4.2 CNN with Attention Layer

Attention mechanisms in the CNN with an attention layer improve this model's feature extraction by focusing on the parts relevant to the input image. Thus, it has almost the same architecture as the simple CNN model but includes an additional attention layer after flattening the feature maps. Attention is a mechanism that works with the following equations:

Attention Mechanism:

$$\text{Attention Output} = \text{Attention}(\text{Query}, \text{Key}, \text{Value})$$

Global Average Pooling:

$$\text{Output} = \text{GlobalAveragePooling1D}(\text{Attention Output})$$

Fully Connected Layers:

$$\text{Output} = \text{Activation}\big(\text{Dense}(\text{Attention Output}, \text{Units})\big)$$

3.4.3 ResNet50

The ResNet50 model uses a deep residual network architecture with 50 layers, using skip connections to enable the training of deep networks. The top classification layer is excluded from the pre-trained ImageNet weights and replaced by custom layers for brain tumor classification. The architecture is defined as:

Residual Block:

$$\text{Output} = \text{Activation}\big(\text{Residual}(\text{Input}, \text{Layers})\big)$$

Global Average Pooling:

$$\text{Output} = \text{GlobalAveragePooling2D}(\text{Input})$$

Fully Connected Layers:

$$\text{Output} = \text{Activation}\big(\text{Dense}(\text{Input}, \text{Units})\big)$$

3.4.4 MobileNetV2

MobileNetV2 is a light, computationally optimized model for mobile and edge devices. It uses depth-wise separable convolutions, which help to reduce computational complexity. It has pre-trained layers on ImageNet and is fine-tuned for brain tumor classification. This model contains:

Depth-wise Separable Convolution:

$$\text{Output} = \text{DepthwiseConv}(\text{Input}, \text{Filters})]$$

Global Average Pooling:

$$\text{Output} = \text{GlobalAveragePooling2D}(\text{Input})$$

Fully Connected Layers:

$$\text{Output} = \text{Activation}\big(\text{Dense}(\text{Input}, \text{Units})\big)$$

3.4.5 DenseNet121

DenseNet121 relies on densely connected convolutional layers; every layer is connected directly to all preceding layers. This model also uses pre-trained ImageNet weights and fine-tunes them for brain tumor classification. Architecturally, it features:

Dense Block:

$$\text{Output} = \text{DenseBlock}(\text{Input}, \text{Layers})$$

Global Average Pooling:

$$\text{Output} = \text{GlobalAveragePooling2D}(\text{Input})$$

Fully Connected Layers:

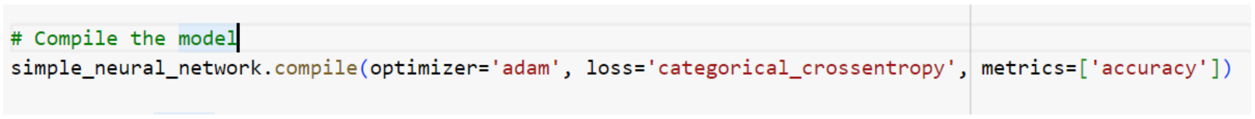
$$\text{Output} = \text{Activation}\big(\text{Dense}(\text{Input}, \text{Units})\big)$$

The models were chosen based on recorded success in image classification projects and the usefulness of transfer learning for better performance. The CNN is a reliable basis with customized layers for specific functions, while the adoption of attention aids in prioritizing essential aspects. ResNet50, MobileNetV2, and DenseNet121 offer advanced architectures that enhance feature extraction and classification precision. These models were selected to analyze their competence and comprehend their advantages and disadvantages regarding brain tumor classification.

3.5 Training of Model

The models were trained using different deep learning architectures, including Convolutional Neural Network (CNN), CNN with attention layer, ResNet50, MobileNetV2, and DenseNet121.

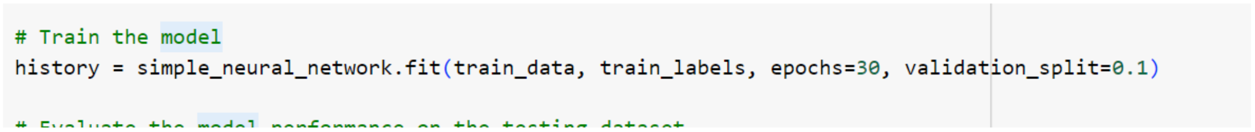
Compilation: The Adam optimizer and categorical cross-entropy loss function were used to compile each model since this combination fits perfectly for multiclass classification problems. Accuracy was the metric used for evaluation during training, as shown in Figure 3.3.



```
# Compile the model
simple_neural_network.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 3.3 Model Compilation

The models were fitted to the training data for various epochs to enhance the training process. Expressly, 30 epochs were set aside for the model's training. During training, a split of 10% of the training data was used to avoid overfitting and check the model performance on unseen datasets, as shown in Figure 3.4.



```
# Train the model
history = simple_neural_network.fit(train_data, train_labels, epochs=30, validation_split=0.1)

# Evaluate the model performance on the testing dataset
```

Figure 3.4 training of the machine learning model

3.6 Evaluation of Models

In evaluating model performance, several metrics and methods were employed:

3.6.1 Performance Metrics

The primary metrics to evaluate the models include precision, recall, and F1 score. In this way, they give a holistic view of how well each model is performing in different classes by looking at its accuracy, precision (the ratio of true positives to all the predicted positives), recall (the ratio of true positives to actual positives) and harmonic mean of precision and recall (F1-score).

3.6.2 Confusion Matrices

The performance visualizations of all models and their misclassifications were represented individually using confusion matrices for each model. In addition, it is essential to understand where models are making mistakes; hence, the confusion matrix provides detailed information about true positives, false positives, true negatives, and false negatives per class.

3.7 Experimental Setup

The experiments were conducted in a cloud-based environment using the GOOGL Colab, which provides a flexible and powerful platform to run deep learning models. The critical aspects of the experimental setup are given below:

3.7.1 Hardware Specifications

GPU: is used to accelerate computation and reduce training time. Models were trained on GPU using Google Colab. Depending on availability, it provides access to high-performance GPUs like NVIDIA Tesla K80, T4, or P100.

RAM: Up to 12GB RAM available by Google Colab for handling large datasets and model parameters during training.

3.7.2 Software Specifications

Programming Language: Python relegates itself as a primary programming language for implementing deep learning models.

Deep Learning Framework: TensorFlow and Keras were used as frameworks, which include essential tools in building, training, and evaluating neural networks.

Google Colab: The whole code was written in GOOGLE Colab notebooks, which makes it easier to code online.

3.8 Summary

In conclusion, the methodology for this study entailed a systematic approach to modeling and evaluation. The deep learning models like CNN, CNN with attention layer, ResNet50, MobileNetV2, and DenseNet121 were crafted and trained to identify brain tumors from MRI images. Adam optimizer and categorical cross-entropy loss function were used in compiling each model, then trained for 30 epochs with a 10% validation split, ensuring a solid performance. Precision, recall, F1-score, and confusion matrices were used to evaluate each model's effectiveness and accuracy. The execution took place on Google Colab, utilizing its computational resources to handle large-scale image data alongside complex models. This methodological approach presented a holistic framework for analyzing the performance of various models on the classification of brain tumors, hence achieving the study's objectives plus providing meaningful contributions to the field.

4 Results and Discussion

The study results highlight the comparative performance of various deep learning models, including CNN, CNN with an attention layer, ResNet50, MobileNetV2, and DenseNet121, in classifying brain tumors into four distinct classes: glioma, meningioma, nontumor, and pituitary. Each model was evaluated using precision, recall, and F1-score to assess their effectiveness in identifying and differentiating between these tumor types. The baseline CNN model demonstrated strong performance overall, particularly in classifying nontumor and pituitary tumors, with F1 scores reaching 0.98. However, it exhibited slightly reduced accuracy in identifying glioma and meningioma tumors. Introducing an attention mechanism to the CNN architecture improved performance, particularly in the meningioma class. This indicates that the attention layer helped the model focus on more relevant features, thus enhancing classification accuracy. This section provides a detailed analysis of each model's performance, offering insights into their strengths and limitations in handling this critical medical imaging task.

4.1 Class wise Result of Deep Learning Model

First of all, the class-wise results are analyzed. Each class's precision, recall, and f1 score on each applied model are evaluated individually. The baseline CNN performed well in nontumor and pituitary classifications (F1-scores of 0.98) but was slightly less accurate for glioma and meningioma, as shown in Table 4.1. Adding an attention layer improved CNN's performance, particularly for meningioma, increasing the F1-score to 0.96. ResNet50 excelled in nontumor classification but struggled with glioma and meningioma, while MobileNetV2. Despite being lightweight, it showed lower accuracy for these classes. DenseNet121 had the lowest overall performance, particularly for glioma and meningioma, indicating a need for further model refinement or larger datasets. The consistently high performance in nontumor classification suggests this class is more easily distinguished, while glioma and meningioma remain challenging. The attention-enhanced CNN and ResNet50 were the most balanced models, but further improvements, such as ensemble learning or data augmentation, may enhance performance in complex cases. This analysis highlights the need for careful model selection and tuning based on task-specific challenges.

Table 4.1 Class wise Result of Deep Learning Model

Model	Target Class	Precision	Recall	F1-score	Support
CNN	glioma	0.99	0.94	0.96	300
	meningioma	0.93	0.84	0.94	306
	nontumor	0.97	1.00	0.98	405
	pituitary	0.99	0.98	0.98	300
CNN with attention layer	glioma	0.97	0.98	0.98	300
	meningioma	0.98	0.94	0.96	306
	nontumor	0.99	1.00	0.99	405
	pituitary	0.98	0.99	0.98	300

ResNet50	glioma	0.96	0.87	0.91	300
	meningioma	0.86	0.95	0.91	306
	nontumor	1.00	1.00	1.00	405
	pituitary	0.96	0.95	0.95	300
MobileNetV2	glioma	0.93	0.83	0.87	300
	meningioma	0.82	0.88	0.86	306
	nontumor	0.99	0.99	0.99	405
	pituitary	0.94	0.97	0.96	300
DenseNet121	glioma	0.91	0.76	0.83	300
	meningioma	0.76	0.80	0.78	306
	nontumor	0.89	0.99	0.94	405
	pituitary	0.97	0.93	0.95	300

4.2 Overall results of the Deep learning model

The overall results of these models show a clear hierarchy in performance, with CNN with the attention layer leading the group, achieving the highest Precision, Recall, and F1-score of 0.98, indicating its superior ability to identify and classify brain tumor images correctly. The baseline CNN follows closely with uniformly strong performance across all metrics and has 0.97 scores, making it a robust choice, though slightly less effective than Inception CNN. ResNet50, while still competent, shows a slight drop in performance. The RestNet has a 0.94 score on Recall and F1 Score, which may indicate some challenges in capturing all relevant cases accurately. MobileNet20, known for its lightweight architecture, exhibits slightly lower scores, with 0.92 across all metrics, suggesting a trade-off between efficiency and accuracy. DenseNet12 shows the lowest performance, with an F1-score of 0.87, reflecting potential difficulties in complex feature extraction and indicating it may not be as reliable for this specific classification task. Overall, while all models perform reasonably well, Inception CNN is the most effective, with DenseNet12 requiring further refinement or possibly using additional data to improve its classification capabilities.

Table 4.2 Overall Result of the Deep Learning Model

Model	Accuracy	Precision	Recall	F1-score
CNN	0.97	0.97	0.97	0.97
CNN with attention layer	0.98	0.98	0.98	0.98
ResNet50	0.95	0.95	0.94	0.94
MobileNet20	0.92	0.92	0.92	0.92
DenseNet12	0.88	0.88	0.87	0.87

4.3 Confusion Matrix

The performance of all applied deep learning models was further evaluated by plotting the confusion matrices for each model. These confusion matrices, visualized in Figure 4.1, provide a detailed view of each model's classification accuracy and errors. The confusion matrices offered a thorough insight into the performance of applied deep learning models in classifying brain tumors into four distinct classes: glioma, meningioma, nontumor, and pituitary. The true positive (TP) and true negative (TN) values are crucial indicators of each model's effectiveness in accurately identifying the correct tumor class and avoiding misclassifications.

The CNN model exhibited strong performance, particularly in classifying nontumor and pituitary tumors, with a high true positive rate of 404 for nontumor, 295 for pituitary, and true negative rates of 973 for nontumor and 996 for pituitary. However, it showed slightly reduced accuracy in identifying glioma where TP is 282 and meningioma TP is 289, indicating challenges in differentiating these classes due to their potentially more complex and less distinguishable features.

Introducing an attention mechanism into the CNN architecture significantly enhanced its performance, particularly for the meningioma class, where the true positive value increased to 297. The model achieved the highest overall performance with true positive rates of 294 for glioma, 297 for meningioma, 405 for nontumor, and 297 for pituitary, along with high true negative values across the board, 1003 for glioma, 992 for meningioma, 999 for nontumor, 997 for pituitary. This improvement underscores the efficacy of attention layers in focusing on the most relevant features of an image, thereby increasing the model's classification accuracy.

ResNet50 also performed well, especially in the classification of nontumor cases. The TP rate is 405, and the TN rate is 993, but it struggled with glioma, where the TP is 260, and the meningioma class TP is 292. While it demonstrates high performance in certain areas, the slightly lower true positive rates for glioma and meningioma suggest that this model may require further refinement or additional data better to capture the nuances of these more complex tumor classes

MobileNetV2 showed slightly lower accuracy, particularly for the glioma class, where the TP is 248 and the TN is 980. For the meningioma class, the TP is 272, and TN is 976. This trade-off between model efficiency and classification accuracy suggests that while MobileNetV2 is suitable for environments with limited computational resources, it may not be the best choice for tasks requiring high accuracy.

Lastly, DenseNet121 had the lowest overall performance, especially when struggling with glioma, where the TP was 228 and the TN was 952. For meningioma, the TP is 244, and TN is 963. Its lower true positive and true negative rates indicate low precision, recall and F1 score of the model. These results suggest that DenseNet121's current form may not be as reliable for brain tumor classification as the other models evaluated.

Overall, the consistently high performance of all models in nontumor classification suggests that this class is more easily distinguishable, possibly due to the distinct differences between normal and tumor affected brain tissue. The CNN model enhanced by an attention layer is the most balanced and practical approach among all the applied models. Additionally, the training history graphs for all models were plotted to assess their generalization capabilities. This comprehensive evaluation through confusion matrices and history graphs offers valuable insights into the strengths and weaknesses of each model.

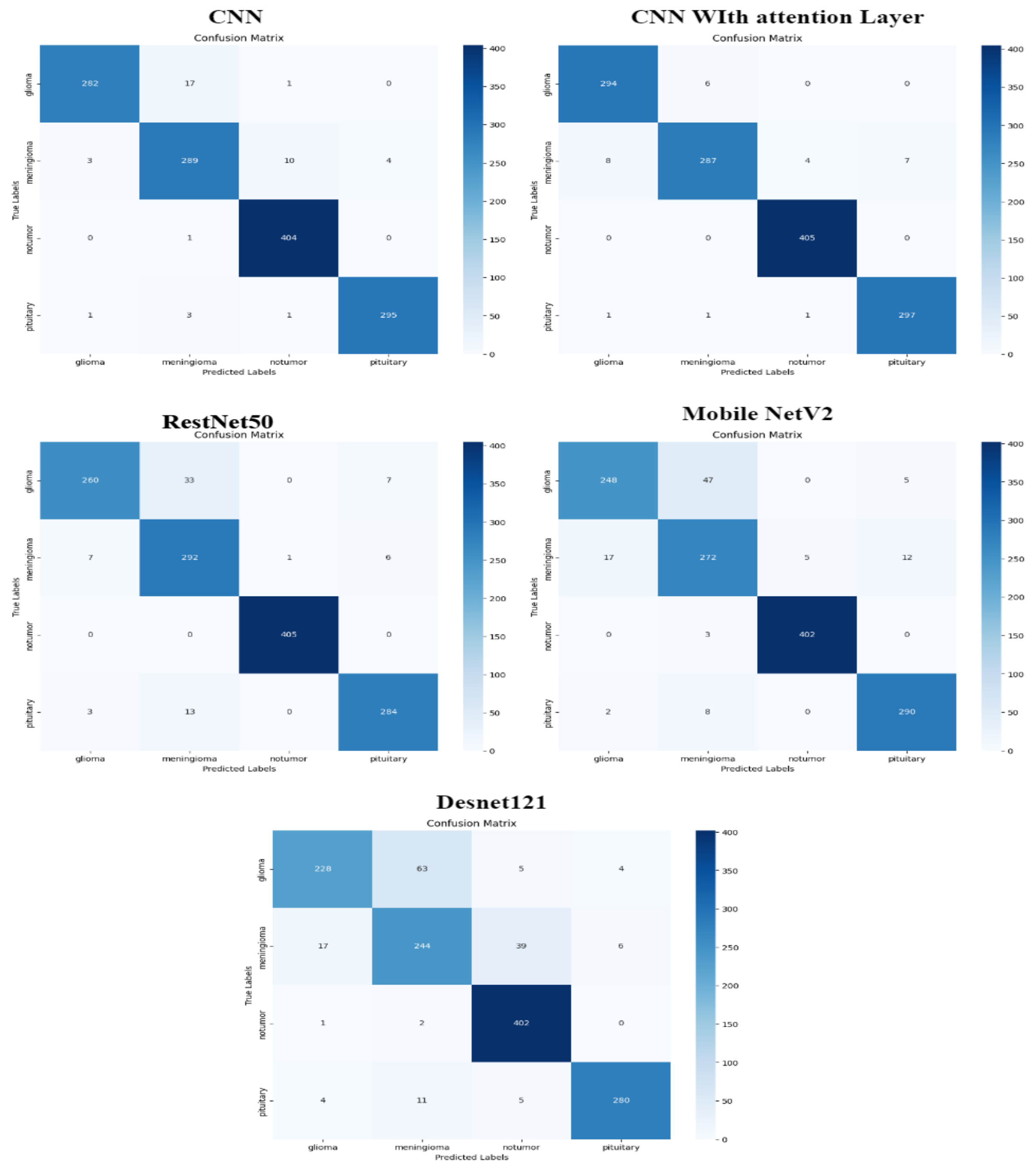


Figure 4.1 Confusion Matrix of Applied Machine Learning Model

4.4 History Graph

The history graph of each model, as shown in Figure 4.2, reveals that the CNN model with an attention layer exhibits a more consistent and smooth increase in accuracy compared to the other models applied in this study. This smoothness in the accuracy curve indicates better generalization, suggesting that the CNN with attention is less prone to overfitting and maintains stable performance across different data distributions.

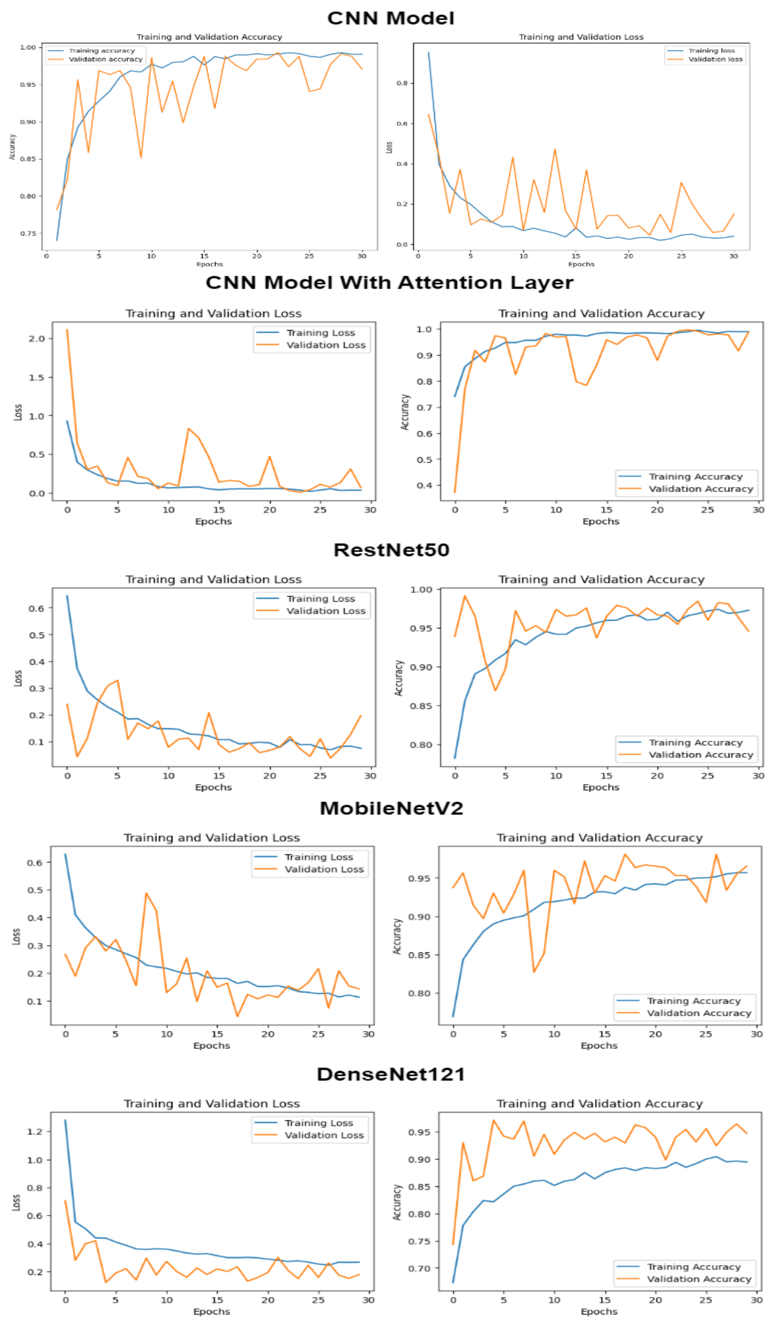


Figure 4.2 History graph of the Machine Learning model

5 Analysis and Discussion

Our study provides useful insights into the efficacy of various deep-learning models in classifying brain tumors using MRI images and proposes a model, namely, a CNN with an attention layer, which could classify brain tumor MRI images accurately. our study has tested five models inclusing CNN, CNN with an attention layer, ResNet50, MobileNetV2, and DenseNet121 for their ability to classify a brain MRI into these four classes: glioma, meningioma, nontumor, and pituitary. These results shows that these models are excellent at specific classification. however, Destnet21 exhibit poor performance with more complex tumor types.

5.1 Model performance and analysis

Among all the models compared in the present research study, the most robust is the single attention layer CNN for the classes of meningioma and pituitary. This likely also significantly contributes to this model's success by focusing the model on only the most relevant parts of the MRI images, which can help make distinctions between the many subtle differences among the tumor types. While ResNet50 performed well, especially for non-tumors, where it scored a perfect true positive rate, the results show that it can perform well in cases with more straightforward tasks.

5.2 Comparison to Existing Literature

Results are thus compliant with the literature pointing out that CNN-based architectures work well on medical image classification tasks. Also, the added attention layer improvement is compliant with studies focusing on advantages provided by attention mechanisms for model focus and accuracy. Nevertheless, the slightly lower performance of MobileNetV2 and DenseNet121, compared with the rest of the models, may deviate from some studies that praise their efficiency, meaning these models require further tuning or perhaps more enormous datasets to realize their full potential in this application.

5.3 Limitations of the Study

Performance across different classes of tumors varied, particularly for glioma and meningioma, indicating that while the models work well on more distinct classes, they have problems with nuanced ones. That is to say, more sophisticated models or more training data will be required. Moreover, the generalizability of the findings might also be limited because MRI images were sourced from one dataset only; the study's findings might not translate very well into other datasets or imaging modalities.

5.4 Relevance to Project Objectives

The project's objective was to benchmark the performance of many deep learning models against each other in classifying a brain tumor and hence be able to tell the most accurate model. The results successfully meet these objectives by identifying the CNN with an attention layer as the top performer and giving insights into each model's strengths and weaknesses. The findings contribute to the excellent vision of enhancing the diagnosis of brain tumors using state-of-the-art machine learning techniques.

5.5 Practical Implications

Clinical application would be especially significant. The attention-layer CNN and ResNet50 models have a high potential for application in the clinic, particularly for classifying nontumor and pituitary tumors. High accuracy and F1 scores make these methods reliable for supporting the diagnosis of brain tumors, especially when integrated into decision-support systems aimed at supporting radiologists in the interpretation of MRI scans.

5.6 Discussion and Future Work

On analyzing MRI images of brain tumors, this study provides an in-depth comparison of different deep learning models, namely CNN, CNN with an attention layer, ResNet50, MobileNetV2, and DenseNet121, whose performances for the tumor classes (glioma, meningioma, nontumor and pituitary) differed significantly. The baseline CNN model achieved remarkable performance in classifying nontumor with F1-scores above 0.98, whereas gliomas had slightly lower performance due to their inherent complexity. TIntroducingan attention mechanism into CNN resulted in notable improvement in classification problems, especially for meningiomas, culminating in enhanced focus on relevant aspects of images these layers serve. ResNet50 started very well by achieving an F1 score of 1 in nontumor classification but failed in both gliomas and meningiomas, suggesting the need for further modifications. MobileNetV2 had lower accuracies than other network types, although it worked efficiently, pointing out the trade-off between efficiency and accuracy. The results obtained from this model were the lowest overall, indicating that detecting differences among images belonging to different classes helps extract complex features requiring more data or architecture changes, particularly in the case of gliomas or meningiomas.

6 Conclusion

In this study, five deep learning models concerning the classification of brain tumors against MRI images are considered. The names of the models are CNN, CNN with an additional attention layer, ResNet50, MobileNetV2, and DenseNet121. The main result is that the model CNN with an additional attention layer performed the best among others, specifically in differentiating the cases of meningioma and pituitary tumors by focusing only on relevant features of the images. ResNet50 was also very accurate, especially in nontumor classification. It showed potential for clinical applications in differentiating normal from tumor-affected brain tissue. On the other hand, MobileNetV2 and DenseNet121 were efficient in computational terms. Still, they turned out to be less accurate, more so in classifying glioma and meningioma tumors, hence requiring further fine-tuning or larger datasets to improve their performance.

The incorporation of attention mechanisms into the CNN model has brought about tremendous improvement in performance by the model in complex classification tasks, making these systems very suitable for clinical applications in diagnosis involving brain tumors. One real-world application of this work would be to integrate this work into decision-support systems that assist radiologists in interpreting MRI scans for accurate diagnosis and increased speed in diagnosing brain tumors.

Future work should be directed toward expanding the dataset to have diverse MRI images, increasing models' generalizability across populations and imaging conditions. Further, investigating more advanced architectures in deep learning, such as transformers or hybrid models combining CNNs with other techniques, may increase the classification accuracy of those challenging tumor types, like glioma and meningioma. Future studies can also test the integration of such models into real-time workflows in clinical operations and provide proof of their usability and reliability in a medical setting.

7 References

Alanazi, A., 2022. Using machine learning for healthcare challenges and opportunities. Inform Med Unlocked 30, 100924. https://doi.org/10.1016/J.IMU.2022.100924

Alqudah, A.M., Alquraan, H., Qasmieh, I.A., Alqudah, A., Al-Sharu, W., 2020. Brain Tumor Classification Using Deep Learning Technique -- A Comparison between Cropped, Uncropped, and Segmented Lesion Images with Different Sizes. International Journal of Advanced Trends in Computer Science and Engineering 8, 3684–3691. https://doi.org/10.30534/ijatcse/2019/155862019

Callahan, A., Shah, N.H., 2017. Machine Learning in Healthcare. Key Advances in Clinical Informatics: Transforming Health Care through Health Information Technology 279–291. https://doi.org/10.1016/B978-0-12-809523-2.00019-4

Deepak, S., Ameer, P.M., 2019. Brain tumor classification using deep CNN features via transfer learning. Comput Biol Med 111, 103345. https://doi.org/10.1016/J.COMPBIOMED.2019.103345

Mohsen, H., El-Dahshan, E.-S.A., El-Horbaty, E.-S.M., Salem, A.-B.M., 2018. Classification using deep learning neural networks for brain tumors. Future Computing and Informatics Journal 3, 68–71. https://doi.org/10.1016/J.FCIJ.2017.12.001

Nayyar, A., Gadhavi, L., Zaman, N., 2021. Machine learning in healthcare: review, opportunities and challenges. Machine Learning and the Internet of Medical Things in Healthcare 23–45. https://doi.org/10.1016/B978-0-12-821229-5.00011-2

Puttagunta, M., & Ravi, S. (2021). Medical image analysis based on deep learning approach. Multimedia Tools and Applications 2021 80:16, 80(16), 24365–24398. https://doi.org/10.1007/S11042-021-10707-4

Sharif, M.I., Khan, M.A., Alhussein, M., Aurangzeb, K., Raza, M., 2022. A decision support system for multimodal brain tumor classification using deep learning. Complex and Intelligent Systems 8, 3007–3020. https://doi.org/10.1007/S40747-021-00321-0/TABLES/6

8 Appendices

Install Kaggle library to download Kaggle dataset to collab.

```
!pip install kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Downloading Kaggle Dataset

```
!kaggle datasets download -d masoudnickparvar/brain-tumor-mri-dataset
```

Unzip the data set

```
!unzip /content/brain-tumor-mri-dataset.zip
```

Getting statistical analysis from the data set

```
import os  # for dataupload
import matplotlib.pyplot as plt # for   visualization
import matplotlib.cm as cm

# Define directories for training and testing datasets
train_directory = '/content/Training'
test_directory = '/content/Testing'

# Function to calculate the number of images in each class directory
def get_image_counts(directory):
    class_folders = os.listdir(directory)
    image_counts = {}
    total_images = 0
    for folder in class folders:
        folderpath = os.path.join(directory, folder)
        if os.path.isdir(folder_path):
            num_images = len(os.listdir( folder_path) )
            image_counts[folder] = num_images
            total_images += num_images
    return image_counts, total_images

# Get counts for training images
training_counts, training_total = get_image_counts(train_directory)
```

```
# Get counts for testing images
testing_counts, testing_total = get_image_counts(test_directory)

# Aggregate counts from both training and testing sets
combined_counts = {}
for class_name in training_counts:
    combined_counts[class_name] = training_counts[class_name] +
testing_counts.get(class_name, 0)

# Generate a color map
color_map = cm.get_cmap('Set2', len(combined_counts))  # Changed color map to 'Set2'

# Plotting the results
plt.figure(figsize=(12, 7))
bars = plt.bar(combined_counts.keys(), combined_counts.values(), color=[color_map(i) for i in
range(len(combined_counts))])

# Customize the plot appearance
plt.xlabel('Class Categories', fontsize=14)
plt.ylabel('Total Image Count', fontsize=14)
plt.title('Image Distribution Across Classes ', fontsize=16)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add value labels inside the bars
for bar in bars:
    height = bar.get_height() # getting the height
    plt.text(bar.get_x() + bar.get_width() / 2,  height  /  2, int( height) , ha=' center ', va =' center',
fontsize=10, color='white')

# Create a custom legend with updated title
legend_labels = combined_counts.keys()
legend_colors = [color_map(i) for i in  range( len (combined_counts))]
legend_elements = [plt.Line2D([0], [0], color=legend_colors[i], lw=4) for i in
range(len(legend_labels))]
plt.legend(legend_elements, legend_labels, title='Classes', bbox_to_anchor=(1.05, 1),
loc='upper left', fontsize=12)

plt.tight_layout()
plt.show()
```

```
# Output total image counts for each class
print("Combined Image Count per Class (Training + Testing):")
for classname, count in combined_count.items():
    print(f"{classname}: {count}")

# Output total image counts for training and testing sets
print(f"Total Images in Training Set: {training_total}")
print(f"Total Images in Testing Set: {testing_total}")
```

plotting single image from the Training data set

```
import os
import numpy as np
import cv2
from tensorflow.keras.utils import to_categorical

# path of  the dataset folder
dataset_path = "/content/Training"


# Define the list of label folders in the dataset folder
label_folders = ['glioma','meningioma','no tumor','pituitary']

# Define the size of images
img_height = 128
img_width = 128

# Define an empty list to store images and  labels
data = []
labels = []

# dictionary to map label folders for  numerical labels
label_mapping = {label: idx for idx, label in enumerate(label_folders)}

for label_folder in label_folders:
    # Define the path to the label folder
    label_path = os.path.join(dataset_path, label_folder)

    # Loop for images in  label folder
    for img_name in os.listdir(label_path):
        # Define the path to the image
        img_path = os.path.join(label_path, img_name)
        # Load image and resize
        img = cv2.imread(img_path)
        img = cv2.resize(img, (img_height, img_width))
        data.append(img)
        labels.append(label_mapping[label_folder])

data = np.array(data)
labels = np.array(labels)

# Print the values
print("Data shape:", data.shape)
print("Labels shape:", labels.shape)
label_mapping
```

Simple CNN model

Load the dataset from the folder, split it into training, and test

```
import cv2
from tensorflow.keras.utils import to_categorical
import tensorflow as tf
from tensorflow.keras import layers, models

# path for the dataset folders
traindatasetpath = "/content/Training"
test_datasetpath = "/content/Testing"

# Define the list of label folders in the dataset folder
label_folders = ['glioma', 'meningioma', 'notumor', 'pituitary']

# size of input images
img_height = 128
img_width = 128

# Function to load and preprocess images from a given path
def load_data(dataset_path, label_folders):
    data = []
    labels = []
    label_mapping = {label: idx for idx, label in enumerate(label_folders)}

    for label_folder in label_folders:
        label_path = os.path.join(dataset_path, label_folder)
        for img_name in os.listdir(label_path):
            img_path = os.path.join(label_path, img_name)
            img = cv2.imread(img_path)
            img = cv2.resize(img, (img_height, img_width))
            data.append(img)
            labels.append(label_mapping[label_folder])

    data = np.array(data)
    labels = np.array(labels)
    labels = to_categorical(labels, num_classes=len(label_folders))

    return data, labels

# Load training data  and preprocess
train_data, train_labels = load_data(train_dataset_path, label_folders)
# Load and preprocess testing data
test_data, test_labels = load_data(test_dataset_path, label_folders)
```

Model Creation

```
# Define the model creation function
def create_simple_neural_network(input_shape, num_classes):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(layers.BatchNormalization())
    # Add the max pooling layer
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    # batch normalization layer
    model.add(layers.BatchNormalization())
# Add the max pooling layer
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# batch normalization layer
    model.add(layers.BatchNormalization())
# Add the max pooling layer
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# batch normalization layer
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model
input_shape = (128, 128, 3)

# Example number of classes (adjust based on your dataset)
num_classes = 4
```

Compile fit and evaluation of the CNN model.

```

# Create the simplified neural network model
simple_neural_network = create_simple_neural_network(input_shape, num_classes)

# Display the model summary
simple_neural_network.summary()

# Compile model
simple_neural_network.compile(, loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train model
history = simple_neural_network.fit(train_data, train_labels, epochs=30, validation_split=0.1)

# Evaluate the model performance on the testing dataset
test_loss, test_accuracy = simple_neural_network.evaluate(test_data, test_labels)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")
```

plot classification report confusion matrix and history graph

```
from sklearn.metrics import, confusion_matrix, classification_report
import seaborn as sns
# Predict the labels
test_predictions = simple_neural_network.predict(test_data)
test_predictions_classes = np.argmax(test_predictions, axis=1)
test_true_classes = np.argmax(test_labels, axis=1)

# print the classification_report
report = classification_report(test_true_classes, test_predictions_classes,
target_names=label_folders)
print(report)
```

```
# Compute the confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_predictions_classes)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_folders,
yticklabels=label_folders)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
import matplotlib.pyplot as plt

# Extract accuracy and loss values from the training history
accuracy = cnnhistory.history['accuracy']
val_accuracy = cnnhistory.history['val_accuracy']
loss = cnnhistory.history['loss']
val_loss = cnnhistory.history['val_loss']
epochs = range(1, len(accuracy) + 1)

# Plot training and validation accuracy
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs, accuracy, '-', label='Training accuracy')
plt.plot(epochs, val_accuracy, '-', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
# xlabels
plt.xlabel('Epochs')
# y labels
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, '-', label='Training loss')
plt.plot(epochs, val_loss, '-', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

CNN Model with attention layer

Model Creation

```
# Define the model creation function
def create_model_with_attention(input_shape, num_classes):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(32, (3, 3), activation='relu')(inputs)
# batch normalization layer
    x = layers.BatchNormalization()(x)
# Add the max pooling layer
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(64, (3, 3), activation='relu')(x)
# batch normalization layer
    x = layers.BatchNormalization()(x)
# add the max pooling layer
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(128, (3, 3), activation='relu')(x)
# batch normalization layer
    x = layers.BatchNormalization()(x)
# add the max pooling layer
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(128, (3, 3), activation='relu')(x)
# batch normalization layer
    x = layers.BatchNormalization()(x)
# add the max pooling layer
    x = layers.MaxPooling2D((2, 2))(x)

    # Flatten the feature maps and add an attention layer
    x = layers.Flatten()(x)
    x = layers.Reshape((-1, x.shape[-1]))(x)  # Reshape for attention layer
    attention_output = layers.Attention()([x, x])
    x = layers.GlobalAveragePooling1D()(attention_output)

    # Fully connected layers
    x = layers.Dropout(0.5)(x)
    x = layers.Dense(512, activation='relu')(x)
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = models.Model(inputs, outputs)
    return model
```

Compile fit and evaluation of the CNN model.

```
# Display the model summary
model_with_attention.summary()

# Compile the model
model_with_attention.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
Attention_history = model_with_attention.fit(train_data, train_labels, epochs=30,
validation_split=0.1)

# Evaluate the model performance on the testing dataset
test_loss, test_accuracy = model_with_attention.evaluate(test_data, test_labels)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")

# Plot the training history

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(Attention_history.history['loss'], label='Training Loss')
plt.plot(Attention_history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(Attention_history.history['accuracy'], label='Training Accuracy')
plt.plot(Attention_history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.show()
```

plot classification report and confusion matrix

```
# Predict the labels for the test data
testpredictions = model_with_attention.predict(test_data)
testpredictions_classes = np.argmax(test_predictions, axis=1)
test_true_classes = np.argmax(test_labels, axis=1)

# Generate the classification report
report = classification_report(test_true_classes, test_predictions_classes,
target_names=label_folders)
print(report)

# Compute the confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_predictions_classes)

# Plot the confusionmatrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_folders,
yticklabels=label_folders)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

ResNet50

```
def create_resnet50_model(input_shape, num_classes):
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=input_shape)
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=outputs)
    for layer in base_model.layers:
        layer.trainable = False

    return model

resnet50_model = create_resnet50_model(input_shape, num_classes)
resnet50_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
resnet50_model.summary()
```

Fit and Evaluate model

```
# Train the model (replace `model_with_attention` with your selected model)
resnet50history = resnet50_model.fit(train_data, train_labels, epochs=30, validation_split=0.1,
batch_size=32)

# Evaluate the model performance on the testing dataset
test_loss, test_accuracy = resnet50_model.evaluate(test_data, test_labels)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")
```

Plot the classification report and confusion matrix

```
1        # Predict the labels for the test data

test_predictions = resnet50_model.predict(test_data)
test_predictions_classes = np.argmax(test_predictions, axis=1)
test_true_classes = np.argmax(test_labels, axis=1)

# Generate the classification report
report = classification_report(test_true_classes, test_predictions_classes,
target_names=label_folders)
print(report)

# Compute the confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_predictions_classes)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_folders,
yticklabels=label_folders)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

Plot the History graph

```
        # Plot the training history
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(resnet50history.history['loss'], label='Train Loss')
plt.plot(resnet50history.history['val_loss'], label='Validation_Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(resnet50histor.history['accuracy'], label='Train Accuracy')
plt.plot(resnet50history.history['val_accuracy'], label='Validation_Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.show()
```

MobileNetV2

```
from tensorflow.keras.applications import MobileNetV2

def create_mobilenetv2_model(input_shape, num_classes):
    base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=input_shape)
    y = base_model.output
    y = GlobalAveragePooling2D()(x)
    y = Dense(512, activation='relu')(x)
    y = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(y)

    model = Model(inputs=base_model.input, outputs=outputs)

    for layer in base_model.layers:
        layer.trainable = False

    return model

mobilenetv2_model = create_mobilenetv2_model(input_shape, num_classes)
mobilenetv2_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
mobilenetv2_model.summary()
```

Fit and Evaluate the model

```
mobilenetv2history = mobilenetv2_model.fit(train_data, train_labels, epochs=30,
validation_split=0.1, batch_size=32)

# Evaluate the model performance on the testing dataset
test_loss, test_accuracy = mobilenetv2_model.evaluate(test_data, test_labels)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")
```

```
# Predict the labels for the test data
test_predictions = mobilenetv2_model.predict(test_data)
test_predictions_classes = np.argmax(test_predictions, axis=1)
test_true_classes = np.argmax(test_labels, axis=1)

# Generate the classification report
report = classification_report(test_true_classes, test_predictions_classes,
target_names=label_folders)
print(report)

# Compute the confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_predictions_classes)
```

```
# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_folders,
yticklabels=label_folders)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

```
# Plot the training history
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(mobilenetv2history.history['loss'], label='Train Loss')
plt.plot(mobilenetv2history.history['val_loss'], label='Validation_Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(mobilenetv2history.history['accuracy'], label='Train Accuracy')
plt.plot(mobilenetv2history.history['val_accuracy'], label='Validation_Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
```

DenseNet121

```
from tensorflow.keras.applications import DenseNet121

def create_densenet121_model(input_shape, num_classes):
    base_model = DenseNet121(weights='imagenet', include_top=False,
input_shape=input_shape)
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=outputs)

    for layer in base_model.layers:
        layer.trainable = False

    return model

densenet121_model = create_densenet121_model(input_shape, num_classes)
densenet121_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
densenet121_model.summary()
```

Fit and Evalute the model

```
# Train the model (replace `model_with_attention` with your selected model)
densenet121history = densenet121_model.fit(train_data, train_labels, epochs=30,
validation_split=0.1, batch_size=32)

# Evaluate the model performance on the testing dataset
test_loss, test_accuracy = densenet121_model.evaluate(test_data, test_labels)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")
```

Plot confusion matrix and classification Report

```
# Predict the labels for the test data
test_predictions = densenet121_model.predict(test_data)
test_predictions_classes = np.argmax(test_predictions, axis=1)
test_true_classes = np.argmax(test_labels, axis=1)

# Generate the classification report
report = classification_report(test_true_classes, test_predictions_classes,
target_names=label_folders)
print(report)

# Compute the confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_predictions_classes)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_folders,
yticklabels=label_folders)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

Plot the History graph

```
# Plot the training history
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(densenet121history.history['loss'], label='Train Loss')
plt.plot(densenet121history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(densenet121history.history['accuracy'], label='Trai Accuracy')
plt.plot(densenet121history.history['val_accuracy'], label='Validat Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.show()
```