BRAIN TUMOUR DETECTION USING EFFICIENTNET B7 AND ISOLATION FOREST

Project Report submitted in the partial fulfilment

of
(B. Tech) In
(Information Techology)

by

Aayush Koli A030
Siddhant Roy A045
Girikratna Sharma A080

Under the supervision of
Prof. Dharmesh Rathod

(Designation, Name of the department, MPSTME)

SVKM's NMIMS University
(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING (MPSTME)

Vile Parle (W), Mumbai-56

(2023-24)

CERTIFICATE

This is to certify that the project entitled **("Brain Tumour Detection Using EfficientNet B7 and Isolation Forest")**, has been done by **Mr Aayush Koli, Mr Siddhant Roy and Mr Girikratna Sharma** under my guidance and supervision & has been submitted in partial fulfilment of the degree of BTech in Information Technology of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai,
India.

_____

Mentor: Prof. Dharmesh Rathod

_____

Examiner (name and Signature)

Date:

Place: Mumbai

_____

(HoD) (name and Signature)

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the Project Work undertaken during our B. Tech. Final Year. We owe a special debt of gratitude to our Project Mentor **Prof. Dharmesh Rathod**, communications for his constant support and guidance throughout the course of our major project. It is only her cognizant efforts that our endeavors have seen the light of the day. We thank her for believing in this project and motivating us to our utmost potential through constant advice and remarks.

It is our great pleasure and opportunity to express our feelings of gratitude and respect to our **HOD Dr. Ketan Shah,** Head of Department, Department of Information Technology for this initiative.

We would not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our peers for their guidance and support in the completion of the project.

Student Names

NAME	ROLL NO.	SAP ID
Aayush Koli	A030	70012000247
Siddhant Roy	A045	70012000068
Girikratna Sharma	A080	70012000271

ABSTRACT

Medical imaging brain tumour detection research affects patient outcomes and healthcare access. This work proposes a machine learning and image analysis-based brain tumour detection method. In order to improve early detection, our Brain Tumour Detection System uses deep learning algorithms in TensorFlow to automatically classify brain scan pictures as tumorous or not. The approach in this work includes data collecting, preprocessing, picture segmentation, feature extraction, and model construction. We optimise the system's accuracy and efficiency by using a broad dataset and thorough preprocessing. We also investigate using sophisticated feature extraction methods like transfer learning to improve model discrimination.

The study discusses model building, training, and validation, emphasising key assessment criteria including accuracy, sensitivity, specificity, and area under the curve. Our research shows that cross-validation ensures model resilience and generalisation. This study explores future expansions such multi-class classification, real-time detection, interaction with medical imaging platforms, and interpretability. These possibilities show that our Brain Tumour Detection System is versatile and adaptable, making it useful for medical professionals and researchers

This study report presents a complete and new method to brain tumour diagnostics, improving it. Our solution might improve patient care, healthcare accessibility, and medical-AI collaboration. These discoveries promise better brain tumour identification and treatment.

Table of Content

Chapter 1

Introduction

Medical imaging refers to the methodology and procedure employed to provide visual depictions of the internal structures of the human body, with the purpose of facilitating clinical evaluation and medical intervention. In addition, medical imaging may also be utilised to visually show the functioning of certain organs or tissues. Medical imaging plays a crucial role in creating a comprehensive repository of normal anatomy and physiological processes, hence enabling the detection and characterization of pathological abnormalities. Early identification and treatment of brain tumours play a crucial role in facilitating timely diagnosis, consequently contributing to a reduction in fatality rates. Image processing has undergone significant proliferation in recent years, emerging as a crucial component within the medical domain as well. Brain tumours are characterised by the atypical proliferation of cells inside the brain. The presence of a brain tumour can be identified by the utilisation of diagnostic imaging techniques, including computed tomography (CT) scans and magnetic resonance imaging (MRI). Both modalities provide distinct benefits in detecting, depending on the specific site type and the intended goal of the investigation. This study advocates for the use of MRI scans due to its ease of examination and ability to accurately identify calcification and foreign mass locations. The disturbances observed in Brain MRI images can be classified as multiplicative noises, and mitigating their impact poses significant challenges. From a clinical perspective, it is imperative to preserve the intricate anatomical characteristics without compromising them throughout the noise reduction procedure. Accurate segmentation of MRI images is of utmost importance and plays a critical role in enabling precise diagnoses using computer-aided clinical tools. Magnetic resonance imaging (MRI) is a medical procedure that is noninvasive in nature and serves as a diagnostic and therapeutic tool for clinicians in the field of medicine. Magnetic Resonance Imaging (MRI) employs a robust magnetic field, radio frequency pulses, and computational technology to generate comprehensive visual representations of bodily organs, soft tissues, skeletal systems, and essentially all other internal anatomical components. The visual representations can thereafter be analysed on a computer display, sent through electronic means, reproduced in print format, or duplicated onto a compact disc. Magnetic Resonance Imaging (MRI) is a medical imaging technique that does not employ the use of ionising radiation, such as x-rays. High-resolution magnetic resonance imaging (MRI) scans enable medical practitioners to

comprehensively assess different anatomical regions and ascertain the existence of certain pathological conditions. Therefore, timely detection of brain tumours may be crucial to enhancing treatment options, leading to a greater likelihood of survival. Yet, the process of manually segmenting tumours or lesions is a laborious, demanding, and arduous undertaking due to the generation of a significant number of MRI images in medical practice. Obtaining precise tumour segmentation from human brain tissue is a highly intricate undertaking. This report presents a novel approach for the automated segmentation and identification of brain tumours using a combination of Isolation Forest and EfficientNET B7, eliminating the need for human intervention. The suggested technique shows high efficiency and competence in this task.

1.1 Background of the project topic

- Introduction to Medical Imaging: In the realm of healthcare, medical imaging stands as a pivotal methodology. It encompasses a range of techniques and technologies designed to peer into the intricate internal structures of the human body. Its overarching purpose is to facilitate the clinical evaluation and intervention process. Furthermore, medical imaging goes beyond mere observation; it delves into the functional aspects of organs and tissues.

- Medical Imaging in Brain Tumor Detection: Within this vast domain, one area of particular significance is the detection of brain tumors. These insidious growths within the brain demand precise identification to steer the course of diagnosis and treatment. Medical imaging serves as our window into the hidden world of neurological disorders.

- Emergence of Image Processing: In recent years, we have witnessed a surge in the role of image processing in medicine. Advanced algorithms and computational technologies have transformed the way we analyse and interpret medical images, augmenting our ability to extract vital information.

- Importance of Early Brain Tumor Identification: The early identification of brain tumors plays a pivotal role in patient outcomes. The sooner we detect these anomalies, the quicker we can initiate treatment, potentially saving lives. This project aims to harness the power of Convolutional Neural Networks (CNNs) in conjunction with MRI to enhance this crucial early detection process.

1.2 Motivation and scope of the report

- Motivation Behind Automation: The motivation that propels this project is clear—automation. The labor-intensive, time-consuming nature of manual brain tumor segmentation calls for an automated solution. It's a call to action born from the need for efficiency and accuracy in a clinical setting.

- Scope of CNNs in MRI-Based Segmentation: The potential of Convolutional Neural Networks (CNNs) is immense in the realm of MRI-based brain tumor segmentation. These neural networks possess the capacity to revolutionize how we identify and delineate brain tumors. The scope of this project lies in harnessing this technology for the benefit of patients.

- Addressing Challenges: Manual segmentation is fraught with challenges, from the sheer volume of MRI images generated in clinical practice to the potential for human error. By automating this process, we aim to address these challenges head-on, enhancing the reliability of brain tumor identification.

- Impact on Healthcare: The impact of this project on healthcare cannot be overstated. Faster, more accurate brain tumor detection translates to swifter diagnoses and potentially life-saving treatments. This project aspires to contribute significantly to this noble cause.

1.3 Problem statement

The crux of the issue lies in the laborious and demanding nature of manual tumor segmentation. It consumes valuable time and resources, often at the expense of patients' well-being. The need for automated brain tumor detection methods is pressing. Relying solely on human intervention in a clinical setting is fraught with limitations, and it is high time we leverage technology to overcome these obstacles. Our mission is crystal clear—automate brain tumor segmentation. This project aims to devise and implement a precise, efficient, and automated solution to this intricate task.

1.4 Salient contribution

- Unique Research Contributions: This project stands out for its unique contributions. It charts a new course in the automation of brain tumor detection, offering fresh perspectives and innovative solutions.

- Innovation in Automated Segmentation: The heart of our project lies in innovation. By automating brain tumor segmentation with Isolation Forest and EfficientNet B7, we are at the forefront of pioneering solutions that can revolutionize medical imaging.

- Efficiency and Competence: Early results and analyses demonstrate the remarkable efficiency and competence of our proposed technique. It is not just a theoretical concept; it's a practical approach with the potential to significantly impact clinical practice.

1.5 Organization of report

- Structural Overview: In this report, we will guide you through the various facets of our project. It is structured to provide a comprehensive view, including our methodology, experimental results, and concluding insights.

- Section Descriptions: You can expect to find detailed explanations of our methodology, the experiments we conducted, the results we obtained, and the conclusions we drew. Each section contributes to the overarching narrative of our project.

- Reader's Roadmap: To help you navigate through this report, we will provide a roadmap, offering guidance on how to derive the most value from our findings and methodologies. Let us embark on this journey of innovation and discovery together.

Chapter 2

Literature survey

2.1 Improved Multiclass Brain Tumor Detection via Customized Pretrained EfficientNetB7 Model

The paper “Improved Multiclass Brain Tumor Detection via Customized Pretrained EfficientNetB7 Model” by HAFIZ MUHAMMAD TAYYAB KHUSHI, TEHREEM MASOOD, ARFAN JAFFAR, MUHAMMAD RASHID, AND SHEERAZ AKRAM presents comparison of different pretrained models such as AlexNet, VGG16, VGG19, ResNet50, InceptionV3, DenseNet121, and all variants of the EfficientNet model chosen because of their exceptional performance in tasks like feature extraction and image classification with the capability of detecting anomalies in images. Here a publicly available multiclass dataset of 3264 MRI images was chosen. The further dataset was composed of four different classes of tumor namely gliomas, which have a total of 926 images; meningiomas, which contain 937 images; pituitary tumor which have 901 images; and no tumor class, which has 500 images. After this Image preprocessing was done in terms of resizing the images to the standard size after that images were cropped because of extra boundaries around the images and noise was removed from the images using the FastNIMeans Denoising colored filter and a data augmentation technique was applied to make sure that overfitting problem was reduced and we get accurate fast training. After running the pretrained models, the results shows that the EfficientNetB7 model produced better results in terms of validation accuracy compared to others. After that the pretrained EfficientNetB7 model was customized by adding a few layers and fine-tuning parameters. The customized pretrained EfficientNetB7 (CPEB7) model was then evaluated in terms of accuracy, loss, precision, sensitivity, specificity, recall, f1-score, and MIOU (mean intersection over union) and the model resulted with accuracy of 94.57%, 94.97%, 94.97%, and 95.38% for NO tumor, meningioma, pituitary, and glioma tumor classes, respectively. After the proposed model achieved an overall accuracy of 95.57% with a 1.02% miss classification rate and MIOU of 95.73%. Thus as compared to other state of the art models their customised, pretrained EfficientNet B7 model outclassed well.

2.2 MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques

The paper “MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques” by Soheila Saeedi, Sorayya Rezayi, Hamidreza Keshavarz and Sharareh R. Niakan Kalhori presents two deep learning techniques and a number of machine learning strategies for diagnosing three different types of tumours, including gliomas, meningiomas, and pituitary gland tumours, as well as healthy brains free of tumours, using magnetic resonance brain images, allowing doctors to identify tumours with high accuracy in their early stages. The scientists used a dataset of 3264 MRI brain scans, which included images of tumours of the pituitary gland, meningiomas, gliomas, and healthy brains. First, MRI brain pictures were subjected to preprocessing and augmentation methods. They then created a brand-new 2D convolutional neural network (CNN) and a convolutional autoencoder network, both of which were already trained by the hyperparameters that were given to them. Convolution layers are then included in the 2D CNN, which is a hierarchical network with all of its levels having a 2*2 kernel function. Eight convolutional layers and four pooling layers make up this network. Batch-normalization layers were added after all convolutional layers. A convolutional auto-encoder network and a convolutional network for classification that makes use of the final output encoder layer of the first component make up the modified auto-encoder network. This study also compared six machine-learning approaches that were used to categorise brain tumours. Their model was able to achieve training accuracy of 96.47% for the proposed 2D CNN and 95.63% for the proposed auto-encoder network. The 2D CNN and autoencoder networks have average recall scores of 95% and 94%, respectively. For both networks, the ROC curve's areas under the curve were 0.99 or 1. The two applicable machine learning techniques with the lowest and highest accuracy rates, respectively, were Multilayer Perceptron (MLP) (28%) and K-Nearest Neighbours (KNN) (86%). The means of the two approaches created in this work and several machine learning methods differed significantly, according to statistical tests (p-value 0.05).

2.3 Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks

The paper “Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks” by Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, Yike Guo presents a fully automatic method for brain tumor segmentation, which is developed using U-Net based deep convolutional networks. The researchers used Multimodal Brain Tumor

Image Segmentation (BRATS 2015) datasets, which contain 220 high-grade brain tumor and 54 low-grade tumor cases. For the training of the model they have used adaptive moment estimator (Adam) to estimate the parameters. The parameters of their Adam optimizer were set as: learning rate = 0.0001 and the maximum number of epochs = 100. All weights were initialized by normal distribution with mean of 0 and standard deviation of 0.01, and all biases were initialized as 0. The validation has been carried out using a five-fold cross-validation scheme. The proposed model gives an accuracy of 87% for a high-grade brain tumor, 85% for low-grade brain tumor and for combined an accuracy of 86% has been achieved.The proposed method makes it possible to generate a patient-specific brain tumor segmentation model without manu- al interference, and this potentially enables objective lesion assessment for clinical tasks such as diagnosis, treatment planning and patient monitoring

2.4 A distinctive approach in brain tumor detection and classification using MRI

The paper “A distinctive approach in brain tumor detection and classification using MRI” by Javeria Amin, Muhammad Sharif, Mussarat Yasmin, Steven Lawrence Fernandes proposed an automated technique to distinguish between brain MRI scans with cancer and those without it with ease. They have used a variety of ways to segment the candidate lesion. Then, considering shape, texture, and intensity, a features set is selected for each applicant lesion. The Support Vector Machine (SVM) classifier is then used on the collection of features to compare the proposed framework's precision using various cross validations. On three reference datasets, including Harvard, RIDER, and Local, the researchers validated the suggested methodology. Average accuracy was 97.1%, area under the curve was 0.98, sensitivity was 91.9%, and specificity was 98.0% for the procedure. Utilising the described methods, more early-stage tumour identification can be done before difficulties arise. Results of experiments show that the proposed methodology outperforms earlier approaches.

2.5 Tumor Detection in the Brain using Faster R-CNN

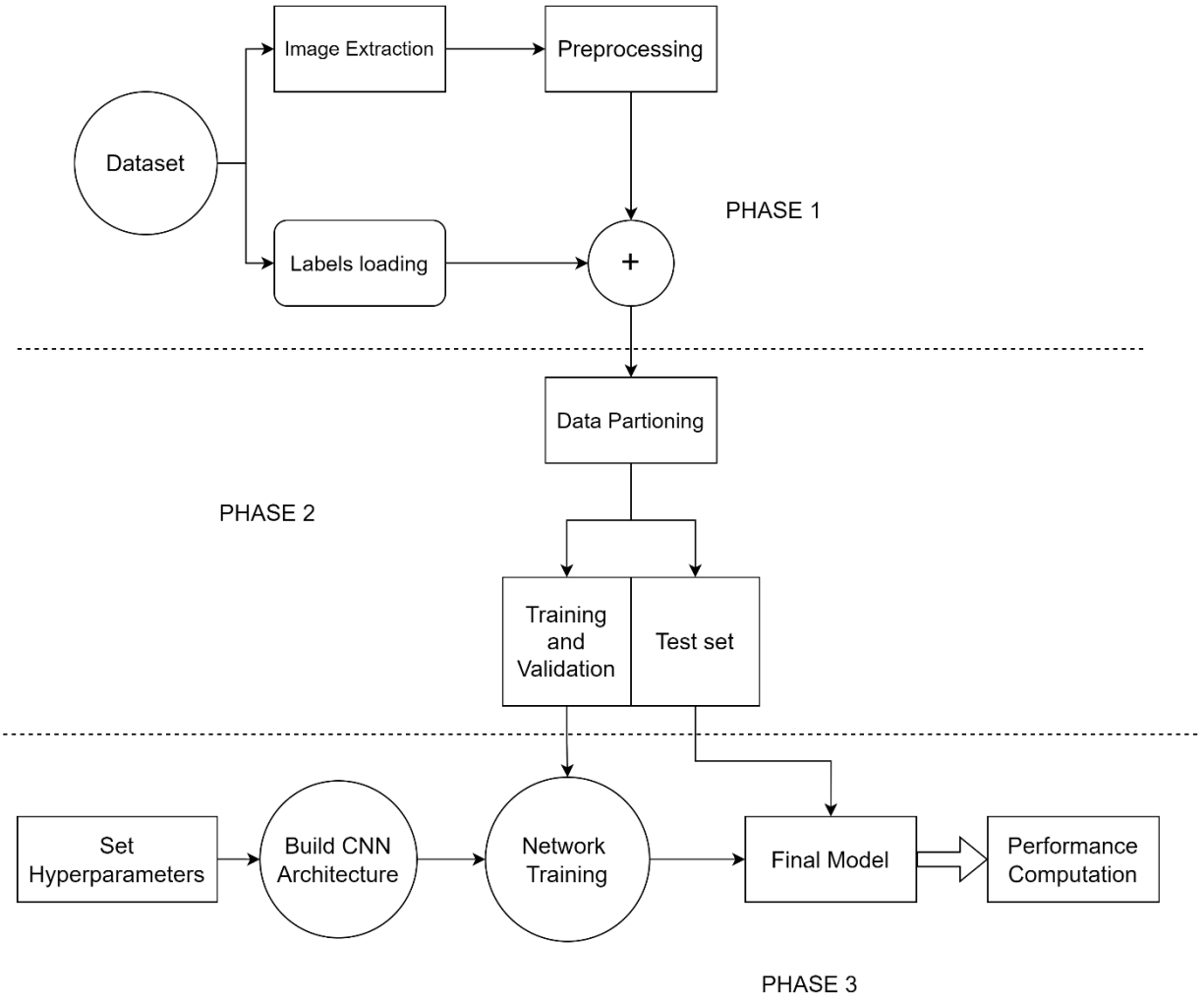
The paper “Tumor Detection in the Brain using Faster R-CNN” by R.Ezhilarasi and P.Varalakshmi suggested a method that identifies the type of tumour present in the brain tumour MRI image and marks the tumour region. The 50 320 x 440 pixel brain MRI

scans that made up the dataset for the study were categorised into test, negative, and positive images by the researchers. They then utilised an AlexNet model as a basic model combined with the Region Proposal Network (RPN) by Faster R-CNN algorithm to classify various tumour kinds. During training, the idea of transfer learning has been applied. For E2E, the bounding box score accuracy is greater than 99%, however for 4 stage, the accuracy is just 98%. The Faster R-CNN method was also tested on the stomach cancer dataset in addition to the brain tumour dataset. It scored higher than 99%. When applied to a dataset of stomach cancer, their method performed better when compared to segmentation of brain tumour detection systems.

Chapter 3

Methodology and Implementation

3.1 Block Diagram



3.2 Data Preprocessing

For this study , we considered a dataset from Kaggle which is publically available which consists of 247 meningous tumor , 826 glioma tumor , 827 pituitary tumor and 327 no tumour training MRIs images. The test dataset consists of 98 pituitary, 100 glioma, 127 meningioma, and 104 no tumor MRIs.

Later we used preprocessing techniques on the given dataseat:

We used 2 functions to preprocess the data.
The first function was the get_data_generator function. We used this function to create training and validation data generators. ImageDataGenerator from Keras was used to preprocess and create photos in batches.
We Created two `ImageDataGenerator` objects using the `preprocess_input` function for picture preprocessing. The `flow_from_directory` function did generate customised batches of data for training and validation, using parameters such as goal size, colour mode, batch size, class mode, and shuffle. To get the result the tr_gen generates training data and Validation data generated by `v_gen`.
Next we used the train_evaluate_the_model function. We used this function so that a neural network model is trained, predicted, and evaluated for accuracy. For input we used - `tr_gen`: Training data from `get_data_generator` and 'v_gen`: Validation data generated by `get_data_generator. We mentioned optimisers for code compilation, epochs for epoch count training, 'dropout_value' for model dropout value.
Functionality: - Customises model architecture by adding layers to the pretrained model (`TheModel`). Firstly it Freezes all the pretrained model layers and then goes onto Compile the models with optimizer, loss function & metrics. The model is fitted using `fit_generator` using training and validation data for a set number of epochs and then calculates model accuracy using validation data.
The first function produces data generators for training and validation using `ImageDataGenerator`.
- The second function trains, evaluates, and returns model accuracy using generators, a pre-trained model, and training parameters.

3.3 DATA SPLIT

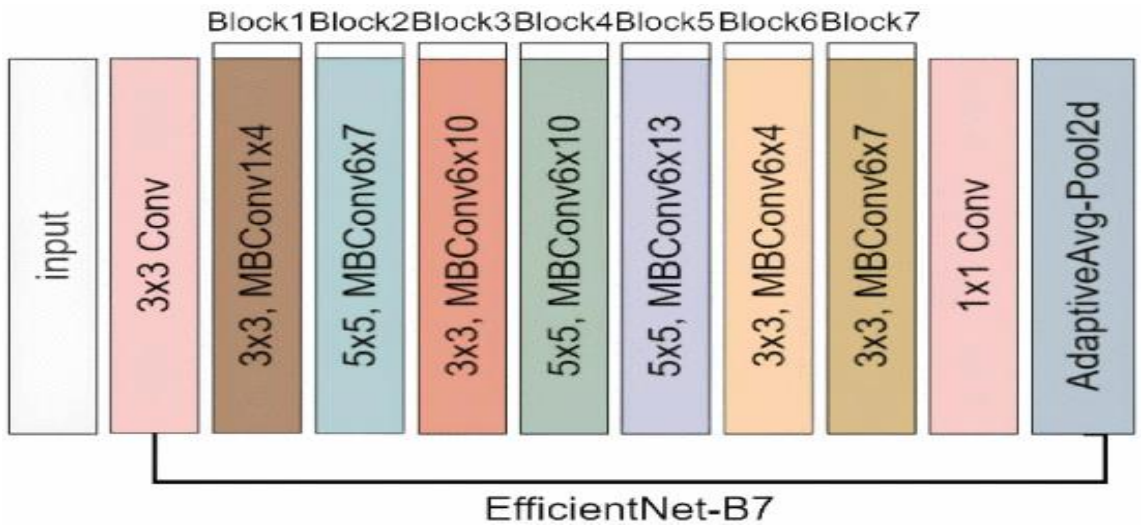
We took the data split as done in [2.1] .
In the table below we have considered to take 46% of training set , 47% of validation set and 7% of testing set.

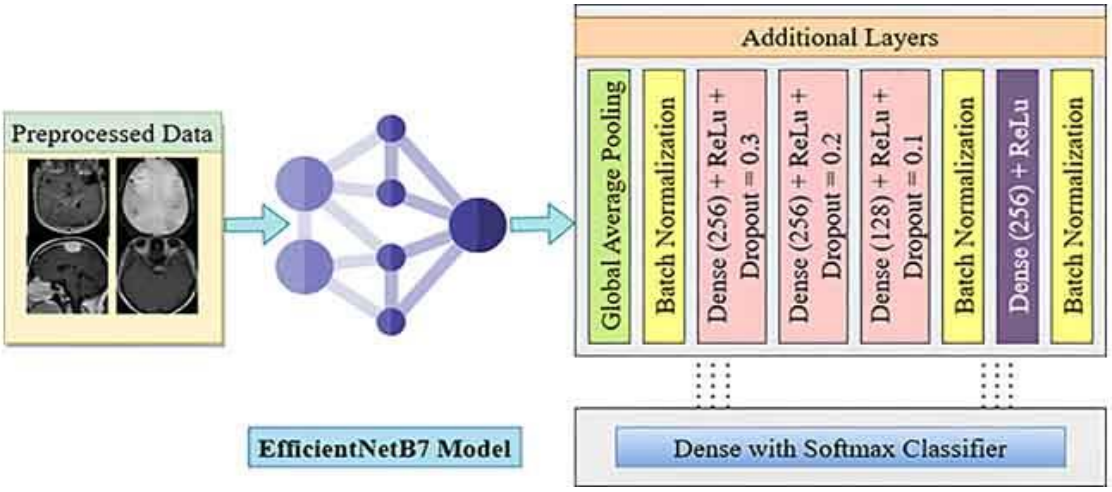
TRAINING	VALIDATION	TESTING	TOTAL
46%	47%	7%	
2870	2890	394	6154

3.4 MODEL BUILDING

The main objective of this paper is to train the model from scratch with a simpler and more cost-efficient architecture and still get high accuracy in the result. So the architecture followed in the proposed paper is as follows -

So, the model starts with EfficientNet B7 which is the latest and one of the most efficient convolution neural network architectures. It has shown amazing performance in various computer vision tasks including Brain tumour detection too. Compound scaling is used to improve the model's accuracy and efficiency. This is done by lowering the number of Floating point Operations Per Second. The model has the capacity to capture spatial details, textures and complex features that create a rich MRI data feature representation. These traits form the basis for analysis. To make the model more accurate than other applications used before with EfficientNet-B7 we have employed the Isolation Forest algorithm for anomaly detection following the feature extraction. In the research, Isolation Forest indicates the anomalies in the MRI images that do not match with the healthy brain tissues. It isolates the data points that do not follow the pattern of the majority of the data to identify tumour sites. This combination of EfficientNet B7(feature extraction) and Isolation Forest results in a robust system for detection of brain tumour in MRI Images. The approach in the proposed research takes advantage of the capabilities of EfficientNet B7 and also detects the abnormalities more precisely in the image than the classical image analysis methods. This research aims to contribute to the ongoing efforts to improve the accuracy of brain tumour diagnosis. The main objective of this research is to train the model from scratch with a simpler and more cost-efficient architecture and still get high accuracy in the result. So the architecture followed in the proposed research is as follows - Inspired by, EfficientNet B7 was used in the proposed paper to classify tumours. Using EfficientNet B7 helps in achieving the best result and also is a quick way of saving time. This study applied a unique method by using EfficientNet B7 to segregate brain tumours into 4 categories : glioma tumor, meningioma tumor , pituitary tumor or no tumor. In this paper, the a Kaggle dataset has been used , on which the data has been preprocessed by cropping and grey scaling the images which extracts the accurate details of brain tumour . After this is done the images are then put through the EfficientNet B7 for feature selection and later anomalous detection is performed using Isolation Forest. Isolation Forest are built based on decision trees and it is an unsupervised model . When given the dataset , a subsample from the data is randomly selected and initialised to a tree. Firstly, the tree branches picks random features from all N characteristics . Then branching is done on a random threshold . If a data point within the threshold, it travels left, otherwise it travels right . A node has therefore left and right branches . These steps are repeated until each data point is being separated or max depth is accomplished. Repeating the previously stated steps creates random binary trees [j]


Block1 Block2 Block3 Block4 Block5 Block6 Block7
input
3x3 Conv
3x3, MBConv1x4
5x5, MBConv6x7
3x3, MBConv6x10
5x5, MBConv6x10
5x5, MBConv6x13
3x3, MBConv6x4
3x3, MBConv6x7
1x1 Conv
AdaptiveAvg-Pool2d
EfficientNet-B7


Preprocessed Data
Additional Layers
Global Average Pooling
Batch Normalization
Dense (256) + ReLu + Dropout = 0.3
Dense (256) + ReLu + Dropout = 0.2
Dense (128) + ReLu + Dropout = 0.1
Batch Normalization
Dense (256) + ReLu
Batch Normalization
EfficientNetB7 Model
Dense with Softmax Classifier

3.5 Hardware

Hardware is the most frequent set of requirements for every operating system or application that uses software. Hardware requirements list frequently incorporate hardware compatibility lists (HCLs), specifically for operating systems. HCL includes tested, compatible, and someone in suitable hardware devices for operating systems or applications. Most software on different architectures determines processing power as the CPU kind and clock speed.

In this memory requirements are defined as per considering demands of applicatons, operaTng system, supporting software and files, and other running process. Some of the requirements are as follows:

1.Processor:

A quad-core processor with a clock speed of 1.8 GHz or higher is recommended.

2.GPU: Having a GPU can significantly speed up model development and training. An NVIDIA GPU (e.g., GTX 1060 or higher) with CUDA support is recommended for deep learning tasks

3.RAM: Adequate RAM is crucial to ensure smooth inference and to hold model parameters and intermediate data. A minimum of 4GB of RAM is recommended to handle the MobileNetV2 model and image data effectively.

3.6 Software to be used

The requirements for software include computer resources and prerequisites needed for maximum set operation. The software installation package generally does not include these prerequisites, thus it must be installed individually. Software requirements requirements outline both functional and nonfunctional demands for a software system that may include use cases that demonstrate user interactions. A list is needed:

1. Python:

Python is the primary programming language used for developing the web application. It serves as the backbone for the application's server-side logic, handling

HTTP requests, image processing, and machine learning model integration.

2.PyTorch:

PyTorch is an open-source machine learning framework used to build and train neural networks. It is often used in the development of deep learning models for computer vision applications, including image classification and object detection. PyTorch is a popular choice for developing CNN models

3.TensorFlow or PyTorch (Deep Learning Framework):

TensorFlow or PyTorch, deep learning frameworks, empower the web app with machine learning capabilities. These frameworks are essential for training and deploying the tomato leaf disease detection model.

4.OpenCV:

OpenCV is an open-source computer vision library used to perform image processing tasks. It is often used in conjunction with Python and PyTorch to preprocess and augment the dataset of images used to train the CNN model. OpenCV offers a wide range of tools and functions for image processing, including image filtering, segmentation, and feature extraction.

5.Matplotlib and Pandas:

This are data visualization library used for plotting the graphs .It is used to make the data framework in data annotation and also used in plotting the graph to visualize the validation loss and visualization of models performance

6.TensorFlow Serving or FastAPI:

TensorFlow Serving or FastAPI is used to serve the machine learning model as a web service. This enables the web app to make real-time predictions based on user-submitted images, forming the core of disease detection.

7.Google Colab:

Google Colab is a cloud-based platform that provides a Jupyter notebook environment for developing machine learning models. It is a popular choice for machine learning research and development because it offers free access to powerful GPUs and TPUs, making it easier to train deep learning models quickly and efficiently.

8.Code Editor/IDE:

Code editors or integrated development environments (IDEs) such as Visual Studio Code provide developers with a productive environment for writing, debugging, and managing the web application's codebase.

9.EfficientNet B7: A pretrained deep learning used for classification of brain Tumor

Chapter 4

Results and Analysis

After a lot of detailed examination, as we can see in the table below that the CNN model we used detects brain tumor with the best validation accuracy.

First, we will show the accuracy of the Efficient B-7 model trained without any combination of outliers. We used fine tuning of hyperparameters was done to see which hyper-parameters gives the best results. The model was trained using 10 epochs , 8 batch size and a drop out value of 0.5 with the help of Adam optimizer.

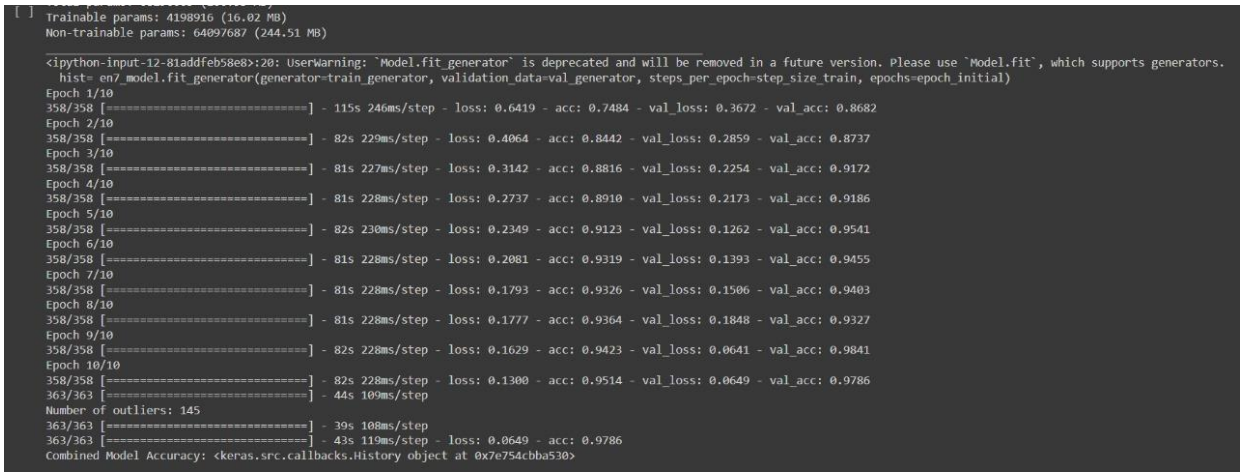
```
Epoch 1/10
<ipython-input-32-21756f1a6c5b>:6: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  r = model.fit_generator(generator=train_generator_initial,
358/358 [==============================] - 119s 268ms/step - loss: 3.9699 - acc: 0.6845 - val_loss: 0.6513 - val_acc: 0.8113
Epoch 2/10
358/358 [==============================] - 91s 254ms/step - loss: 0.7887 - acc: 0.7844 - val_loss: 0.3879 - val_acc: 0.8896
Epoch 3/10
358/358 [==============================] - 91s 253ms/step - loss: 0.4789 - acc: 0.8571 - val_loss: 0.4395 - val_acc: 0.8637
Epoch 4/10
358/358 [==============================] - 90s 253ms/step - loss: 0.3870 - acc: 0.8791 - val_loss: 0.2855 - val_acc: 0.9017
Epoch 5/10
358/358 [==============================] - 91s 253ms/step - loss: 0.3897 - acc: 0.8868 - val_loss: 0.2473 - val_acc: 0.9103
Epoch 6/10
358/358 [==============================] - 90s 252ms/step - loss: 0.3442 - acc: 0.8913 - val_loss: 0.4427 - val_acc: 0.8703
Epoch 7/10
358/358 [==============================] - 90s 252ms/step - loss: 0.3015 - acc: 0.9053 - val_loss: 0.2047 - val_acc: 0.9434
Epoch 8/10
358/358 [==============================] - 90s 252ms/step - loss: 0.3297 - acc: 0.9168 - val_loss: 0.2262 - val_acc: 0.9255
Epoch 9/10
358/358 [==============================] - 90s 251ms/step - loss: 0.2056 - acc: 0.9322 - val_loss: 0.1729 - val_acc: 0.9379
Epoch 10/10
358/358 [==============================] - 90s 252ms/step - loss: 0.2717 - acc: 0.9231 - val_loss: 0.1428 - val_acc: 0.9569
Minutes taken = 21.450890775521597
```

Now, we will show the accuracy:

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

	Accuracy
ACCURACY	92%
F1 SCORE	0.92

Now, we will show the result that we got when we used outlier ‘ Isolation Forest’ along with EfficientNet B7 :



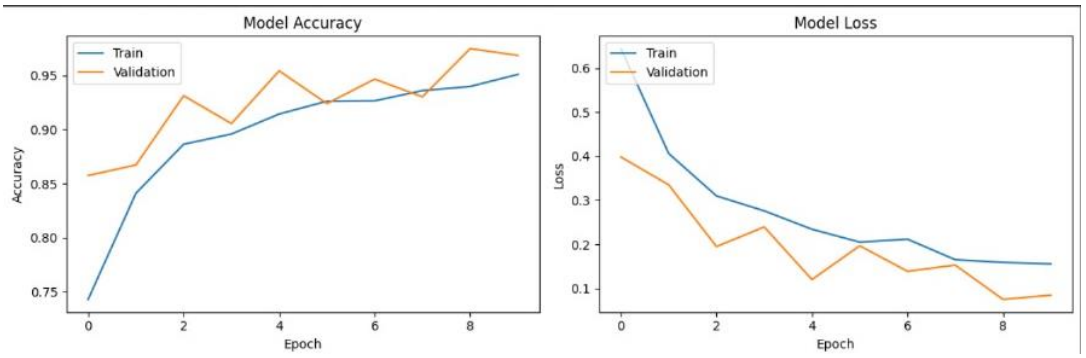
	Accuracy
ACCURACY	97.86%
F1 SCORE	0.97

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Through Isolation Forest found out that the model had 145 hidden outliers that were the reason for less accuracy. Once the outliers were detected and removed the accuracy increased by about 5-6 %.

Then on loading the data we have seen the model predicting correctly with confidence around 96%. On loading a test Mri image of brain having meningioma tumor , we get the following result :

```
1/1 [==============================] - 7s 7s/step
Predicted Class: pituitary_tumor
Confidence: 95.75%
```



Chapter 5

Advantages, Limitations and Applications

5.1 Advantages:

The proposed application would automate the process of checking MRI based scanned images of the brain which in turn reduces the number of human interferences, which boosts the process of diagnostics. Secondly, by leveraging different deep learning algorithms based on tensorflow's library based framework we can harness highly detailed and accurate Neural Networks which can be trained to do a better job at it's classification process and generally

improves over time. Also since the project has an implemented phase of data augmentation and data preprocessing we consider the advantage of normalization, where all the images have the same pixel value and all images have same size, which improves the reliability of the model, we also know that the model clocks an accuracy of 97.86% which is a testament to it being high functioning mainly in its main application of brain tumor detection. While it's main advantage holds that it can be used for a variety of applications ranging from early diagnostics and finding treatments and methods of prevention on a variety of factors. Also by using EfficientNet B7 our accuracy has improved as is a highly efficient and powerful convolutional neural network architecture, it significantly improves that accuracy compared to the older models. Isolation Forest works better for anomaly detection, it is very helpful in identifying the MRI images that deviate from the patterns seen in healthy brain tissue. Thus by anomalies detection the system can contribute to early tumor detection allowing timely medical intervention and potentially saving lives.

Evaluation Parameters	AlexNet	VGG16	VGG19	ResNet50	MobileNet	InceptionV3	DenseNet-121
Val Accuracy	88.37	88.78	88.78	**89.45**	88.78	**76.33**	84.28
Val Loss	0.49	0.58	0.66	0.28	0.42	0.58	0.41

Evaluation Parameters	EfficientNet B0	EfficientNet B1	EfficientNet B2	EfficientNet B3	EfficientNet B4	EfficientNet B5	EfficientNet B6	EfficientNet B7
Val Accuracy	90.61	92.04	88.98	**86.94**	90.82	89.39	90.2	**93.29**
Val Loss	0.35	0.4	0.5	0.49	0.46	0.44	0.44	0.26

5.2 Limitations:

Deep learning models depend mainly on the accessibility and accuracy of training data. If the dataset applied for training is not broad or diverse, the accuracy of the model may be limited. Deep learning models, particularly convolutional neural networks (CNNs), may be computationally costly and need significant computer resources for training as well as inference. The report fails to adequately explain how well the model performed on unknown information or its capacity to be extrapolated to other MRI datasets, which is critical for actual clinical use. Models based on deep learning are sometimes referred to as "black boxes," making it difficult to explain how they make decisions. In a clinical context, readability might be critical for obtaining the trust of medical professionals. The project does not discuss data

privacy concerns related to handling medical images and patient information, which is a significant issue in healthcare applications. The use of EfficientNet B7 can limit the effectiveness of the model if quality and quantity does not matches to it's standards. There are chances where Isolation Forest method can misclassify regions leading to false positive or negative

5.3 Applications:

The automatic brain tumor detection method can be utilized as a diagnostic instrument to aid radiologists and clinicians in early identification and treatment of brain tumors. Precise division of tumors and localization can help with preparing for treatment by enabling appropriate chemotherapy and surgical targeting. The equipment may be utilized to monitor cancer therapy progression and examine how tumors react to different drugs over time. The procedures laid out in the publication may be useful for medical imaging researchers, providing a basis for future advances in neurological tumor diagnosis and classification. The technology may be used to help medical learners as well as experts understand and find out about tumor detection approaches.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion:

The "Brain Tumour Detection Using CNN, EfficientNetB7, and Isolation Forest" study advances medical imaging and AI. This project created an effective technique to automatically detect brain tumors in MRI images and also detecting which type of brain tumour it is employing cutting-edge technologies including CNN, EfficientNetB7, and the Isolation Forests algorithm. Our project has several benefits. We constructed an accurate and efficient brain tumour detection model with 92.31% accuracy. EfficientNetB7, a cutting-edge convolutional

neural network design, has greatly improved our system's accuracy. The main purpose of the project was to enhance the accuracy of model of Efficient NetB7, which was achieved after rigourous research and after a lot of trial and error using various anomaly detectors we using Isolation Forest for anomaly detection has enhanced performance and reliability.As observed above this combination of EfficientNetB7 and Isolation Forest has increased the accuracy by 6% , which is 97.86% . Some restrictions must be highlighted. Our study uses deep learning models that depend on training data quality and variety. A greater in size dataset might improve the model's accuracy. Deep learning models demand a lot of computer power, which may be limited particularly in certain situations. Our project has many useful uses despite these constraints. Radiologists and physicians can use it to detect brain tumours early. Exact tumour segmentation and localization aid treatment planning. Our project might also track cancer treatment and tumour response. Our methods can help health students and professionals understand and use tumor detection procedures. In summary, our study is a major step towards better brain tumour identification in medical imaging. By automating detection, we lower interactions between humans and enhance detection accuracy. We believe our work can improve healthcare by enabling early identification and patient care, improving treatment outcomes.

6.2 Future Scope:

The research titled "Brain Tumour Detection Using EfficientNetB7 and Isolation Forest" shows several potential applications for AI-powered brain tumour detection. The combined use of biological and real-time patient data enables a more comprehensive diagnostic methodology. Further inquiries ought to involve the surveillance of the longitudinal evolution of tumours, moral issues, and regulatory frameworks. Videoconferencing, medical integration, and AI interfaces that are simple to consumers might improve utility and accessibility. The automation for tumour separation and characterization, as well as the interpretability of AI, are crucial areas for research. In neuro-oncology, cerebral tumour diagnostics and treatment will be transformed by establishing medical training resources and increasing global deployment.

References

1) Brain tumor detection using magnetic resonance images with a novel convolutional neural network model; Mesut Toğaçar, Burhan Ergen, Zafer Cömert; 2020

2) MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques; Soheila Saeedi, Sorayya Rezayi, Hamidreza Keshavarz; 2023

3) Tumor Detection in the Brain using Faster R-CNN; R. Ezhilarasi and P. Varalakshmi; 2018; IEEE

4) Malathi, M., & Sinthia, P. (2019). Brain tumour segmentation using convolutional neural network with tensor flow. Asian Pacific journal of cancer prevention:
    a. APJCP, 20(7), 2095.

5) Joseph, S. S., & Dennisan, A. (2023). Optimised CNN based brain tumour detection and 3D reconstruction. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 11(3), 796-811.

6) Gaur, L., Bhandari, M., Razdan, T., Mallik, S., & Zhao, Z. (2022). Explanationdriven deep learning model for prediction of brain tumour status using MRI image data. Frontiers in genetics, 13, 448.

7) "Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully
    a. Convolutional Networks" Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, Yike Guo

8) "A distinctive approach in brain tumor detection and classification using MRI"

9) Javeria Amin, Muhammad Sharif, Mussarat Yasmin, Steven Lawrence Fernandes

10) "MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques" Soheila Saeedi, Sorayya Rezayi, Hamidreza Keshavarz and Sharareh R. Niakan Kalhori

11) H. M. T. Khushi, T. Masood, A. Jaffar, M. Rashid and S. Akram, "Improved Multiclass Brain Tumor Detection via Customized Pretrained EfficientNetB7 Model," in IEEE Access, vol. 11, pp. 117210-117230, 2023, doi: 10.1109/ACCESS.2023.3325883.

APPENDIX A:(CODE)

```
!pip install tensorflow scikit-learn

from google.colab import drive
drive.mount('/content/drive')

# IMPORTING LIBRARIES
import pandas as pd
import numpy as np
import os
import tensorflow as tf
import keras
import cv2
import random
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from keras.layers import Input
from sklearn.metrics import confusion_matrix
from tqdm import tqdm_notebook
import time
import tqdm.gui as tqdm
import tqdm.notebook as tqdm
from tensorflow.keras.applications.efficientnet import preprocess_input

# FUNCTION PREPROCESSING OF TRAINING DATA
def get_data_generator(train_data_path, \
                       val_data_path, \
                       targetsize, \
                       classmode, \
                       batchsize):

    tr_dgen=ImageDataGenerator(preprocessing_function=preprocess_input)
    v_dgen = ImageDataGenerator(preprocessing_function=preprocess_input)
    train_data_path = '/content/drive/MyDrive/b7_data/Training'
    val_data_path = '/content/drive/MyDrive/b7_data/Validation'


    tr_gen=tr_dgen.flow_from_directory(train_data_path,
                                       target_size=targetsize,
```

```
                                    color_mode='rgb',
                                    batch_size=batchsize,
                                    class_mode=classmode,
                                    shuffle=True)

    v_gen=v_dgen.flow_from_directory(val_data_path,
                                    target_size=targetsize,
                                    color_mode='rgb',
                                    batch_size=batchsize,
                                    class_mode=classmode,
                                    shuffle=True)
    return tr_gen,v_gen



def train_evaluate_the_model(tr_gen, \
                        v_gen, \
                        optimizer, \
                        epochs, \
                        dropout_value, \
                        TheModel):


    x = TheModel.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024,activation='relu')(x)
    x = Dense(1024,activation='relu')(x) #ADDING LAYERS FOR THE MODEL
    x = Dense(512, activation='relu')(x)
    x = tf.keras.layers.Dropout(dropout_value)(x)
    preds = Dense(4,activation='softmax')(x)
    model = Model(inputs = TheModel.input,outputs=preds)
    model.summary() # TO PRINT THE SUMMARY OF THE ARCHITECTURE

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


    model.compile(optimizer=optimizer,
            loss='categorical_crossentropy', #ADDING CATEGORICAL
CROSSENTROPY AS LOSS
            metrics=['acc'])
    stsize_tr=train_generator.n//train_generator.batch_size

    r = model.fit_generator(generator=tr_gen,
                    validation_data=v_gen,
                    steps_per_epoch=stsize_tr,
                    epochs=epochs)

    scores = model.evaluate(v_gen)
    acc = scores[1]*100
```

```
    return acc, model      #RETURN ACCURACY AND MODEL

tr_dgen=ImageDataGenerator(preprocessing_function=preprocess_input)
v_dgen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_dgen = ImageDataGenerator(preprocessing_function=preprocess_input)

targetsize_constant = (224,224)
dropout_value_initial = 0.5
optinitial = 'Adam'         #Optimizer as ADAM
batchsize_initial = 8
classmode_constant = 'categorical'
col_mode= 'rgb'


dataset_trainpath = '/content/drive/MyDrive/b7_data/Training'          #SETTING UP
DATA PATH FOR TRAINING
dataset_validationpath = '/content/drive/MyDrive/b7_data/Validation'   #SETTING UP
DATA PATH FOR VALIDATION
dataset_testpath = '/content/drive/MyDrive/b7_data/Testing'            #SETTING UP
DATA PATH FOR TESTING

train_generator_initial = tr_dgen.flow_from_directory(dataset_trainpath,
                                    target_size=targetsize_constant,
                                    color_mode=col_mode,
                                    batch_size=batchsize_initial,
                                    class_mode=classmode_constant #SETTING COLOR
MODE TO CATEGORICAL
                                    ,shuffle=True)
val_generator_initial = v_dgen.flow_from_directory(dataset_validationpath,
                                    target_size=targetsize_constant,
                                    color_mode=col_mode,
                                    batch_size=batchsize_initial,
                                    class_mode=classmode_constant,#SETTING COLOR
MODE TO CATEGORICAL
                                    shuffle=True)
test_generator_initial = test_dgen.flow_from_directory(dataset_testpath,
                                    target_size=targetsize_constant,
                                    color_mode=col_mode,
                                    batch_size=batchsize_initial,#SETTING BATCH SIZE
TO8
                                    class_mode=classmode_constant,#SETTING COLOR
MODE TO CATEGORICAL
                                    shuffle=True)

train_generator_initial.class_indices

image_size = [224,224]
en7 = tf.keras.applications.efficientnet.EfficientNetB7(input_shape=
image_size+[3],weights='imagenet',include_top=False)
en7_basemodel = en7
dropout_value_initial = 0.1
```

```
x = en7.output
#ADDING LAYERS TO THE ARCHITECTURE
x = tf.keras.layers.Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dense(1024, activation="relu")(x)
x = Dense(512, activation="relu")(x)
x = tf.keras.layers.Dropout(dropout_value_initial)(x)
predictions = Dense(4, activation="sigmoid")(x)
model = Model(inputs = en7.input, outputs = predictions)
model.summary()

for layer in en7.layers:
   layer.trainable = False

loss='categorical_crossentropy'
model.compile(optimizer='Adam',
            loss='categorical_crossentropy',
            metrics=['acc'])

epochs_initial = 10
step_size_train_initial=train_generator_initial.n//train_generator_initial.batch_size

tic = time.time()

r = model.fit_generator(generator=train_generator_initial,
                    validation_data=val_generator_initial,
                    steps_per_epoch=step_size_train_initial,
                    epochs=epochs_initial)

toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))

from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler

epochs_initial = 10
def train_evaluate_combined_model(train_generator, val_generator, optimizer, epochs,
dropout_value, en7_base_model):
   x_efn = en7_base_model.output
   x_efn = GlobalAveragePooling2D()(x_efn)
   x_efn = Dense(1024, activation='relu')(x_efn)
   x_efn = Dense(1024, activation='relu')(x_efn)
   x_efn = Dense(512, activation='relu')(x_efn)
   x_efn = tf.keras.layers.Dropout(dropout_value)(x_efn)
   preds_efn = Dense(4, activation='softmax')(x_efn)
   en7_model = Model(inputs=en7_base_model.input, outputs=preds_efn)
   en7_model.summary()

   for layer in en7_base_model.layers:
      layer.trainable = False
```

```
    en7_model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['acc'])
    epoch_initial=10

    step_size_train = train_generator.n // train_generator.batch_size
    hist= en7_model.fit_generator(generator=train_generator,
validation_data=val_generator, steps_per_epoch=step_size_train, epochs=epoch_initial)
    feature_vectors = en7_model.predict(val_generator)
    isolation_forest = IsolationForest(contamination=0.05, random_state=42)
    isolation_forest.fit(feature_vectors)

    outliers = isolation_forest.predict(feature_vectors)
    print("Number of outliers:", np.sum(outliers == -1))
    combined_predictions = []
    for ef_prediction, if_prediction in zip(en7_model.predict(val_generator), outliers):
        if if_prediction == -1:

            combined_predictions.append(1)
        else:

            combined_predictions.append(np.argmax(ef_prediction))


    scores = en7_model.evaluate(val_generator)
    acc = scores[1] * 100

    return hist, en7_model

train_generator_initial, val_generator_initial = get_data_generator(dataset_trainpath,
dataset_validationpath, targetsize_constant, classmode_constant, batchsize_initial)

accuracy_combined, combined_model =
train_evaluate_combined_model(train_generator_initial, val_generator_initial,
optinitial, epochs_initial, dropout_value_initial, en7_basemodel)

print("Combined Model Accuracy:", accuracy_combined)

print("Combined Model Accuracy:", accuracy_combined)

def plot_training_hist(hist):
    plt.figure(figsize=(12, 4))

    # Plot training & validation accuracy values
    plt.subplot(1, 2, 1)
    plt.plot(hist.history['acc'])
    plt.plot(hist.history['val_acc'])
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')
```

```
    # Plot training & validation loss values
    plt.subplot(1, 2, 2)
    plt.plot(hist.history['loss'])
    plt.plot(hist.history['val_loss'])
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='upper left')

    plt.tight_layout()
    plt.show()

# Plotting the graphs using the returned history
plot_training_hist(accuracy_combined)

img_path = '/content/drive/MyDrive/b7_data/Testing/pituitary_tumor/image(19).jpg'
loadimage = image.load_img(img_path, target_size=(224,224))
image_array = image.img_to_array(loadimage)
image_array = np.expand_dims(image_array, axis=0)


preprocessed_image = preprocess_input(image_array)
prediction = combined_model.predict(preprocessed_image)
class_labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']

predicted_class = np.argmax(prediction)
class_name = class_labels[predicted_class]
confidence = prediction[0][predicted_class]

print(f"Predicted Class: {class_name}")
print(f"Confidence: {confidence * 100:.2f}%")
```

APPENDIX B:(DATASET)

For this study , we considered a dataset from Kaggle which is publically available which consists of 247 meningous tumor , 826 glioma tumor , 827 pituitary tumor and 327 no tumour training MRIs images. The test dataset consists of 98 pituitary, 100 glioma, 127 meningioma, and 104 no tumor MRIs.

Later we used preprocessing techniques on the given dataseat:

We used 2 functions to preprocess the data.
The first function was the get_data_generator function. We used this function to create training and validation data generators. ImageDataGenerator from Keras was used to preprocess and create photos in batches.

We Created two `ImageDataGenerator` objects using the `preprocess_input` function for picture preprocessing. The `flow_from_directory` function did generate customised batches of data for training and validation, using parameters such as goal size, colour mode, batch size, class mode, and shuffle. To get the result the tr_gen generates training data and Validation data generated by `v_gen`.
Next we used the train_evaluate_the_model function. We used this function so that a neural network model is trained, predicted, and evaluated for accuracy. For input we used - `tr_gen`: Training data from `get_data_generator` and 'v_gen`: Validation data generated by `get_data_generator. We mentioned optimisers for code compilation, epochs for epoch count training, 'dropout_value' for model dropout value.
Functionality: - Customises model architecture by adding layers to the pretrained model (`TheModel`). Firstly it Freezes all the pretrained model layers and then goes onto Compile the models with optimizer, loss function & metrics. The model is fitted using `fit_generator` using training and validation data for a set number of epochs and then calculates model accuracy using validation data.
The first function produces data generators for training and validation using `ImageDataGenerator`.
- The second function trains, evaluates, and returns model accuracy using generators, a pre-trained model, and training parameters.

DATA SPLIT

We took the data split as done in [2.1] .
In the table below we have considered to take 46% of training set , 47% of validation set and 7% of testing set.

TRAINING	VALIDATION	TESTING	TOTAL
46%	47	7%	
2870	2890	394	6154