Brain Tumor Classification Using MR Images And Transfer Learning

Abstract:-

Brain tumor is the abnormal growth of a mass tissue in the brain or near it,which has the ability to spread and multiply uncontrollably effecting the functions of other organs in the body.Brain tumor classification is the crucial task to evaluate the tumors and make a treatment decision according to their classes. There are many imaging techniques used to detect the brain tumor. However, MRI is commonly used to its superior image quality. Deep Learning (DL) is the subfield of machine leaning and recently showed a remarkable performance, especially in classification problem.

In this paper, a deep learning model for classification of brain tumor from MRI images using Convolution Neural Network (CNN) based on transfer learning. The pre-trained model VGG16 and InceptionV3 are used to extract the deep features from the brain MRI. The experiment is conducted on a dataset of 7023 images which contains 4 types brain tumor (glioma, meningioma, no tumor, pituitary). Images from the dataset were cropped, preprocessed and augmented for accurate and fast training. Deep transfer learning models are trained and tested on a brain MRI dataset using optimization algorithm Adam. The performance of the transfer learning model is evaluated using the performance metrics such as accuracy, precision, recall, support, f1-score. We compare the results of accuracy, recall, f1-score values of the three models,namely CNN,InceptionV3 and VGG16.Our proposed CNN model based on VGG16 architecture using adam optimizer is better than other models. Finally,our proposed pre-trained model VGG16 was observed to be the best among the other implemented models and it achieved accuracy of 98%, on MRI large dataset. The proposed method is superior existing literature, indicating that it can be used for large size dataset and getting accurate and quick results of brain tumor..

Keywords*:* Brain Tumor Classification, Convolution Neural Network (CNN),Deep Learning (DL), Magnetic Resonance Images (MRI), Visual Geometry Group (VGG16),Inception-V3, Adam Optimizer.

I.INTRODUCTION

The brain is one of most complex organ in the human body, controlling the entire nervous system and working with billions of cells. The is most sensitive organ of our body. It controls core functions and responsible for many regulatory functions if human body such as memory, emotion, vision and reaction.

A brain tumor can be expressed as a tissue that occurs in a place where it should not be in our brain or an uncontrolled growth of any tissue where it should not be [1]. Brain tumor can happen in the brain tissue. Brain tumors also can happen near the brain tissue. Nearby locations include nerves, the pituitary gland, the pineal gland, and the membranes that cover the surface of the brain.

BTs are arranged under two significant heads, The one that is developed within the brain is termed as a primary brain tumor and correspond to 70% of all BTs on the other hand tumors that swells into the brain from some other body parts are called secondary brain tumors and forms the residual 30%, majority of which are of malignant type [7].

In the initial stage, cancer does not spread much due to the tumor, but in the later stage, there is a possibility of spreading to more parts of the body [2]. Many different types of primary brain tumor exist. Some brain tumor aren’t cancerous. These are called noncancerous brain tumor or benign brain tumor. Noncancerous brain tumor may grow over time and press on the brain tissue. Other brain tumors are brain cancers, also called malignant brain tumors. Brain cancers may grow quickly. The cancer cells can invade and destroy the brain tissue.Malignant tumors

are cancerous hence it should be detected at an early stage must be diagnosed properly [2].

Brain tumor range in size from very small to very large. Some brain tumors are found when they are very small because they cause symptoms that you notice right away. Other brain tumors grow very large before they're found. Some parts of the brain are less active than other. If a brain tumor starts in a part of the brain that's less active, it might not cause symptoms right away. The brain tumor size could become quite large before the tumor is detected.

1.1 Types of brain tumors:

Gliomas and relates brain tumors: Gliomas aare growth of cells that look like glial cells. The glial cells surround and support nerve cells in the brain tissue. Types of gliomas and related brain tumor include astrocytoma, glioblastoma, oligodendroglioma and ependymoma.

Meningiomas: Meningiomas are brain tumors that start in the membranes aroun the brain and spinal cord. Meningiomas are usually benign, but sometime they can be malignant. Meningiomas are the most common type of bengin brain tumor.

Pituitary tumors: Brain tumors can begin in and around the pituitart gland. This small gland is located near the base of the brain. Most tumors that happen in and around the pituitary gland are benign. Pituitary tumors happen in the pituitary gland itself. Carniopharyngioma is a type of brain tumor that happens near the pituitary gland.

In clinical studies on brain anatomy, MRI has become a crucial tool. The high resolution, contrast, and clear separation of the soft tissue enable doctors to identify specific diseases accurately [6] . MRI provides good quality images of the human body organs in 2D and 3D formats. MR imaging modality is nowadays considered to be one of the best accurate technique for MRI classification, due to its high-resolution images on the brain tissues, the modality is also utilized to diagnose various diseases due to its image quality [10].

There are several medical imaging techniques used to acquire information about tumors (tumor type, shape, size, location, etc.), which are needed for their diagnosis [9]..Therefore, any size of brain tumors are identified by the magnetic resonance imaging, also called MRI, use strong magnets to create pictures of the inside of the body. MRI is often used to detect brain tumor because it shows the brain more clearly than do other imaging tests.

Often a dye is injected into a vein in the arm before on MRI. The dye makes clearer pictures. These makes it easier to see smaller tumors. It can help your health care team see the difference between a brain tumor and healthy brain tissue.

Sometimes you need a special type of MRI to create more detailed pictures. One example is functional MRI. This special MRI shows which parts of the brain control speaking, moving and other important tasks. This helps your health care provider plan surgery and other treatments.

Another special MRI test is magnetic resonance spectroscopy. This test uses MRI to measure levels of certain chemicals in the tumor cells. Having too much or too little of the chemicals might tell your health care team about the kind of brain tumor you have.

Magnetic resonance perfusion is another special type of MRI. This test uses MRI to measure the amount of blood in different parts of the brain tumor. The part of the tumor that have a higher amount of blood may be the most parts of the tumor. Your health care team uses this information to plan your treatment.

The MRI dataset we considered for feature extraction has different modes of scans which can be classified based on their scanning planes.We used the MR Images of different planes namely,the axial(the plane that divides the brain scan to top and bottom halves),the coronal(the perpendicular plane) and saggital(the plane dividing the body into two halves).This makes the model to extract more number of features and easy to classify them based on the observed result.

1.2 SCOPE

Detection of brain tumors in real-time can be achieved through various imaging techniques such as magnetic resonance imaging (MRI), computed tomography (CT), and positron emission tomography (PET) scans.

MRI is the most commonly used imaging technique for detecting brain tumors. It uses a magnetic field and radio waves to create detailed images of the brain. CT scans use X-rays to create images of the brain, and PET scans use a radioactive tracer to show how the brain is functioning.

Real-time detection of brain tumors can be helpful for doctors in several ways. It allows them to quickly identify the location and size of the tumor, which can help guide treatment decisions. Real-time imaging can also be used during surgery to help surgeons precisely remove the tumor while minimizing damage to healthy brain tissue.

In addition, real-time monitoring of the tumor during and after treatment can help doctors evaluate the effectiveness of the treatment and make any necessary adjustments. Overall, real-time detection of brain tumors is a valuable tool in the diagnosis and treatment of brain tumors.

1.2 EXISTING SYSTEM

The existing systems are also computer based, but the accuracy of the existing system is low when compared to the proposed system. They require high computing energy and large memory.

Fail to detect the small and unobvious brain lesions Computationally exhaustive still need to work on light weight model for accurate brain tumor detection.

The size of the dataset used in the existing system is very low.

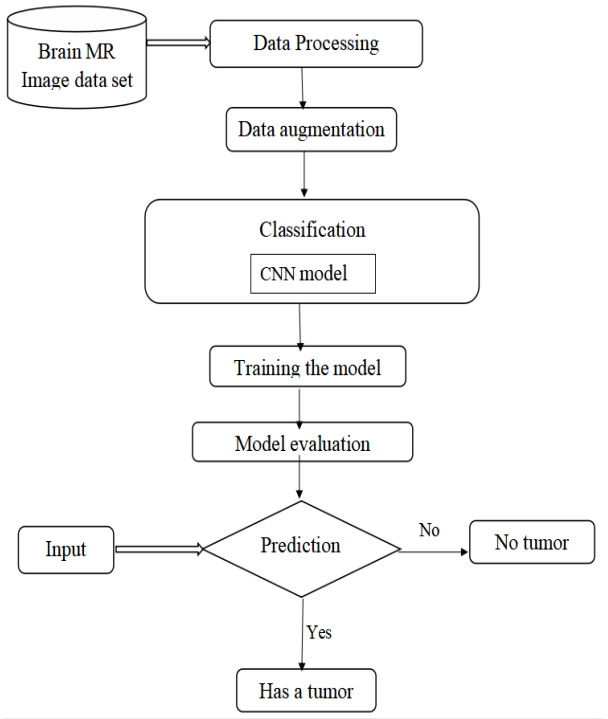


Fig.:1.1 Existing system block diagram

1.3 OBJECTIVE

Our objective is to build a system that works with Convolution Neural Network using Transfer Learning Techniques. The model is trained with augmentation methods and generate good accuracy among all the transfer learning and predefined architectures. In transfer learning algorithms based on highly complex neural network that mimic the human brain works to detect patterns in large unstructured data set. Transfer Learning can analyze images, videos, and unstructured data in which machine learning can't easily do. The model is able to achieve high accuracy on the target dataset, while still being computationally efficient.

2. Literature Survey

TABLE 1. **LITERATURE SURVEY**

Author	Dataset	Feature Extraction Method	Classification Method	Accuracy
Cinar et al. [1]	253 MR images	Improved ResNet50	CNN	97.01%
Rai et al. [2]	253 MR images	Combination of Le-Net and U-Net known as LU-Net	Fully connected layers and sigmoid activation function	98%
Islam et al. [3]	253 MR images	Super pixels and Principal Component Analysis (PCA)	Tk-means Clustering	95%
Das et al. [4]	253 MR images	Deep-CNN	Deep-CNN	98%
Togacar et al. [5]	253 MR images	Hypercolumn, attention module and residual block	CNN	96.05%
Jia et al. [6]	500 MR images	Fully Automatic Heterogeneous Segmentation (FAHS)	SVM	98.51%
Mehrota et al. [7]	696 MR images	Transfer Learning (TL) Techniques	CNN	99.04%
Kaplan et al. [8]	3064 MR images	Conventional local binary pattern operator	SVM	93.43%
Francisco et al. [9]	3064 MR images	CNN	Multi-pathway convolutional neural network (CNN)	97.3%
Ullah et al. [10]	71 MR images	DWT	Feed-forward neural network	95.8%
Lu et al. [11]	66 MR images	MobileNetV2	MobileNet-ELM-CBA, MobileNet-SNN-CBA and MobileNet-RVFL-CBA	98.33%
Rajan et al. [12]	41 MR images	Adaptive Gray-Level Co-Occurrence Matrix (AGLCM)	SVM	98%
Preethi et al. [13]	20 MR images	GLCM + Wavelet GLCM	Deep neural network (DNN)	99.3%
Hemanth et al. [14]	220 MR images	CNN	CNN	94.5%
Proposed	7023 images	VGG-16	CNN	98%

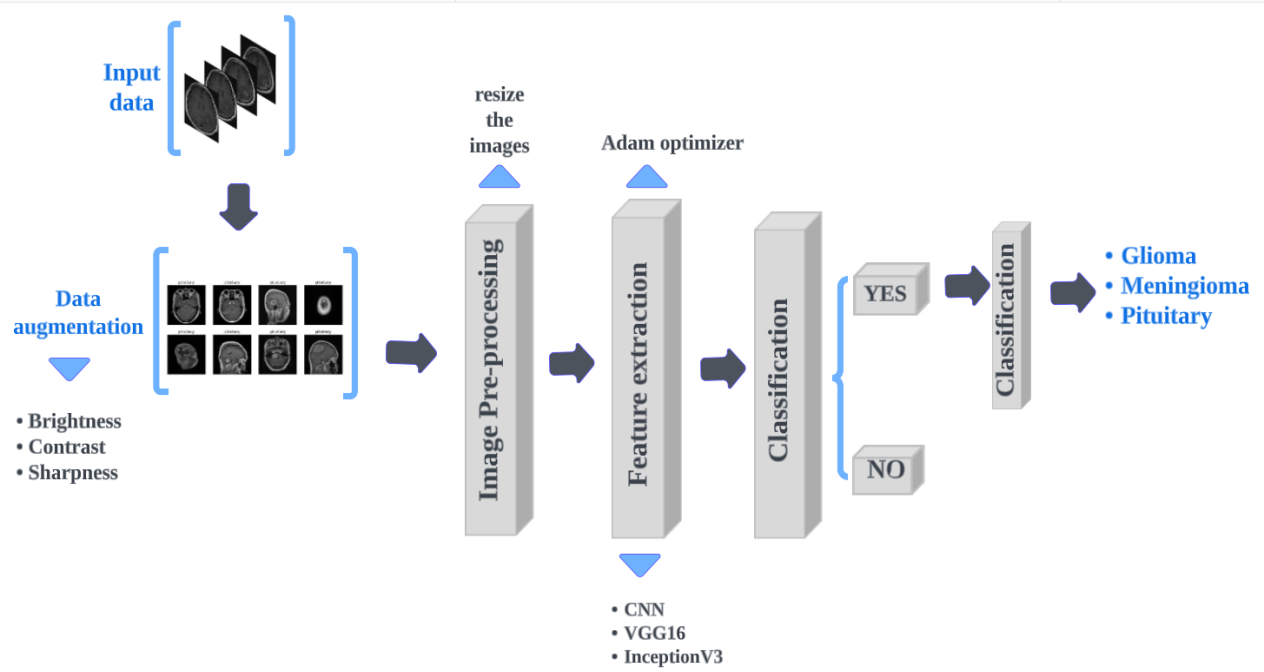


Figure3.1. System framework

3. PROPOSED SYSTEM

Our objective is to build a system that works with Convolution Neural Network using Transfer Learning Techniques. The model is trained with augmentation methods and generate good accuracy among all the transfer learning and predefined architectures.

Loading the dataset: The first step is to load the dataset into memory. This can be done using Python libraries such as NumPy or Pandas. The dataset should be divided into training and test sets.

Augmentation and image pre-procesing: Data augmentation techniques such as and cropping can be used to artificially increase the size of the dataset and reduce overfitting. This step can be done using Python libraries such as Keras or OpenCV.

Feature extraction: Feature extraction involves extracting important features from the images that can help in distinguishing between different types of tumors. This can be done using pre-trained models such as VGG16 or InceptionV3, which have been trained on large datasets of images and can extract meaningful features from the input images.

Classification: Once the features have been extracted, a classification model can be trained to classify the images into different tumor types.

Further classification of tumor type: After the images have been classified into different tumor types, further classification can be done to identify the specific type of tumor.

Overall, this process involves loading the dataset, augmenting the data, extracting important features, classifying the images to identify if there is a tumor or not, and further classifying the specific types of tumor if there is a tumor. This process requires a combination of image processing, machine learning, and deep learning techniques and can be computationally intensive.

3.1 CNN MODEL

A Convolutional Neural Network (CNN) is a type of neural network that is primarily used for image classification tasks. It is inspired by the structure and functioning of the visual cortex in the human brain.

A CNN consists of multiple layers of filters, also called kernels or weights, which are applied to the input image to extract features[8]. Each filter performs a convolution operation by sliding over the input image and calculating dot products at each location . The result of this operation is a feature map that highlights the presence of specific features in the input image. The filters in the early layers of the CNN extract simple features such as edges and curves, while the filters in the later layers extract more complex features.

After the feature extraction, the output feature maps are passed through a series of fully connected layers to classify the image[12]. During training, the weights of the filters are adjusted through backpropagation, where the error between the predicted and actual output is used to update the weights.

CNNs are widely used for various image recognition tasks such as object detection, segmentation, and classification [13].They have also been successfully applied to other domains such as natural language processing and speech recognition.

3.1.1 CNN MODEL ALGORITHM

The algorithm for a typical CNN model is:

1. CNN model initialization

2. Convolution Layer - Applies convolution operations to the input image.

3. Activation function. An activation function (typically ReLU) is applied to introduce nonlinearity.

4. Full layer. Reduces the dimensionality to spatially reduce the resolution of an image.

5. Repeat steps 2-4 to create more feature maps.

6. Flatten - Flatten the output of the previous layer to create a one-dimensional feature vector.

7. Fully Connected Layer - connects all neurons from the previous layer to the current layer.

8. Activation function. Apply an activation function to the fully connected layer (usually ReLU).

9. Screening. We randomly remove some neurons to prevent overfitting.

10. Repeat steps 7-9 to create more fully connected layers.
11. Output layer. Classify the input data into categories using the softmax function.

12. Compile the model - specify the optimizer, loss function and estimated metrics.

13. Train the model - Fit the model to the training data.

14. Model evaluation - model evaluation on test data

15. expect. Use the trained model to make predictions based on new data.

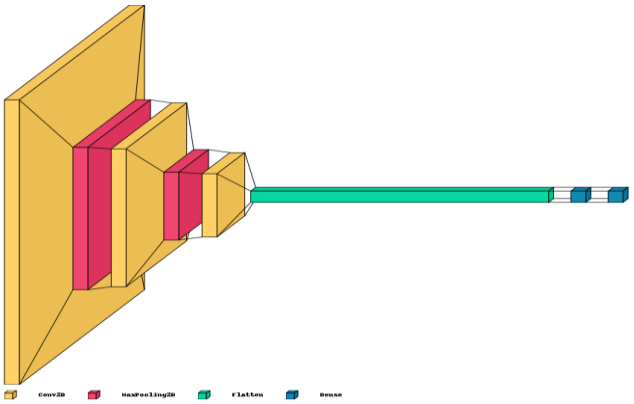


Figure 3.2 Layered view of CNN model used

3.2 TRANSFER LEARNING MODEL

Transfer learning is a technique used in machine learning and deep learning, where a model learned in one task is reused for a second task. Rather than training the model from scratch, transfer learning allows us to first use the model as a baseline and then adjust the model for a new task by adjusting its weights and biases.

Transfer learning is particularly useful when limited information is available for new tasks, as it allows us to use information learned from a larger and more diverse dataset . For example, we can take a previous image classification model that has been trained to recognize various objects and use that as a starting point for recognizing new images, such as identifying types of cars or flowers. Reusing the process of removing the set of

pre-training models and training the final set from scratch only on new data.

3.2.1 Working of transfer learning

Transfer learning is commonly used in deep learning tasks where the amount of available data is limited. Instead of training a deep learning model from scratch, transfer learning allows us to leverage the pre-trained weights of an existing model, which has already been trained on a large dataset. We can use this pre-trained model as a feature extractor to extract features from our own dataset and then train a new model on top of these extracted features[15]. This can save a significant amount of time and computational resources.

For example, we could use a pre-trained convolutional neural network like VGG or Inception to extract features from images in our own dataset. We can then train a smaller neural network on top of these features to classify the images. The pre-trained model already learned to recognize low-level features such as edges, curves, and patterns. [16] By reusing this knowledge, we can focus our efforts on training a smaller network to learn the higher-level features specific to our dataset, rather than starting from scratch and attempting to learn everything from scratch. This approach can lead to better results and faster convergence compared to training a model from scratch.

3.3 VGG

VGG (Visual Geometry Group) is a convolutional neural network architecture that was proposed by the Visual Geometry Group at the University of Oxford in 2014. It is a very deep neural network with 16-19 layers, and it achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014, which is a benchmark for image classification. The VGG architecture is known for its simplicity and uses only 3x3 convolutional filters stacked on top of each other, which makes it easier to understand and implement compared to some of the more complex neural network architectures.

3.3.1. Transfer learning model using VGG16

Transfer learning using VGG16 involves using a pre-trained VGG16 model as a base network for a new image classification task. The pre-trained VGG16 model was originally trained on a large dataset of images and can extract features from images with high accuracy. Instead of training a new model from scratch, transfer learning allows us to leverage the learned features of the pre-trained VGG16 model as a starting point for our new classification task.

To use VGG16 for transfer learning, we typically remove the last fully connected layer and replace it with a new fully connected layer that outputs the number of classes in our new task.[17] We then freeze all the weights of the VGG16 model layers and only train the new fully connected layer. This allows us to quickly fine-tune the pre-trained model for our specific task with a small amount of training data, leading to faster and more accurate results.

3.3.2 VGG16 Algorithm

VGG16 is a convolutional neural network architecture that was developed by the Visual Geometry Group at the University of Oxford. The architecture is composed of 16 layers, hence the name VGG16. The VGG16 architecture is composed of a stack of convolutional layers, each followed by a max pooling layer, and three fully connected layers at the end.

The architecture consists of the following layers:

1. Input layer: The input is a 224x224 RGB image.

2. Convolutional layer 1-2: Two convolutional layers with 64 filters of size 3x3, each followed by a max pooling layer of size 2x2.

3. Convolutional layer 3-4: Two convolutional layers with 128 filters of size 3x3, each followed by a max pooling layer of size 2x2.

4. Convolutional layer 5-7: Three convolutional layers with 256 filters of size 3x3, each followed by a max pooling layer of size 2x2.

5. Convolutional layer 8-10: Three convolutional layers with 512 filters of size 3x3, each followed by a max pooling layer of size 2x2.

6. Convolutional layer 11-13: Three convolutional layers with 512 filters of size 3x3, each followed by a max pooling layer of size 2x2.

7. Fully connected layer 1-3: Three fully connected layers with 4096 neurons each, followed by a dropout layer and

a final fully connected layer with 1000 neurons (corresponding to the 1000 ImageNet classes).

The VGG16 algorithm is trained on the ImageNet dataset, which consists of over 1 million labeled images from 1000 different categories. During training, the algorithm learns to classify images into one of these categories. After training, the algorithm can be used to classify new images by passing them through the network and obtaining a predicted class label.

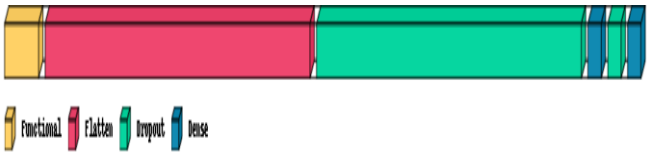


Figure 3.3. Layered view of proposed VGG16 model

3.4 INCEPTION Model

The Inception model is a deep convolutional neural network architecture that was introduced by Google researchers in 2014. It was designed to achieve state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset, which contains over a million images with 1000 classes. The Inception model is characterized by its use of inception modules, which are blocks of layers that perform multiple convolutions with different kernel sizes in parallel. The output feature maps from these parallel convolutions are then concatenated and passed through a single 1x1 convolutional layer before being fed into the next inception module[19]. This approach allows the network to capture features at different scales and resolutions, while also keeping the number of parameters relatively low compared to other models. Since then, the Inception model has been updated to various versions, including Inception v2, v3, v4, and the latest Inception-ResNet-v2, which combines Inception and ResNet architectures. The Inception model and its variants have been used in a variety of computer systems. Visual tasks including image classification, object detection and semantic segmentation.

3.4.1 InceptionV3 Model :

Inception-v3 is a convolutional neural network architecture that was introduced by Google in 2015. It is an improved version of the original Inception model and is designed to be both deeper and more computationally efficient than its predecessor.[20]The main idea behind Inception-v3 is to use a combination of small and large convolutions in parallel to capture features at multiple scales, which helps the network learn more efficient representations of the input image.

Inception-v3 also includes a number of other features such as factorization, aggressive regularization, and improved training techniques, which all contribute to its improved performance over the original Inception model. Inception-v3 has been widely used for various computer vision tasks such as image classification, object detection, and semantic segmentation.

3.4.2 Transfer learning using InceptionV3 model

Transfer learning using Inception v3 involves using the pre-trained Inception v3 model as a feature extractor and then training a new classifier on top of it [20]. The pre-trained Inception v3 model has already learned to extract useful features from images, so we can use its convolutional layers to extract features from our own dataset without having to train a new model from scratch.

The process involves freezing the weights of the pre-trained Inception v3 model and then adding a new classifier on top of it. The new classifier is then trained using the extracted features. This approach can save a lot of time and computational resources, especially when dealing with limited training data.

3.4.3 InceptionV3 Algorithm:

The algorithm for Inception-v3 can be summarized as follows:

1. Convolutional layer: The input image is fed into a convolutional layer with a small filter size (3x3), using a stride of 2. This is followed by a max pooling layer with a filter size of 3x3 and a stride of 2.

2. Inception modules: A series of inception modules are stacked together to extract features from the input image. An inception module consists of 1x1, 3x3, and 5x5 convolutions along with a max pooling layer, all concatenated along the channel dimension. Batch normalization and ReLU activations are applied after each convolutional layer.

3. Auxiliary classifiers: Two auxiliary classifiers are added to the network after some of the inception modules

to improve the gradient flow and regularize the network. Each auxiliary classifier consists of a global average pooling layer, a fully connected layer, a ReLU activation, and a softmax layer.

4. Final layers: The output of the last inception module is fed into a global average pooling layer, which averages the feature maps along the spatial dimensions. The resulting feature vector is fed into a fully connected layer with 1,024 units and a ReLU activation, followed by a dropout layer with a rate of 0.4. The final output layer is a softmax layer with the number of units equal to the number of classes.

5. Training: The network is trained using backpropagation and stochastic gradient descent.The weights of the network are updated using the Adam optimizer.

6. Fine-tuning: The trained network can be fine-tuned on a new dataset by unfreezing some of the layers and adjusting the learning rate. The lower layers are fine-tuned with a lower learning rate, while the higher layers are fine-tuned with a higher learning rate. This process helps to improve the performance of the network on the new dataset.

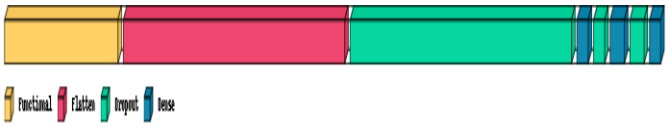


Figure 3.4. layered view of proposed Inception model

4. SYSTEM ANALYSIS

This System Analysis is closely related to requirements analysis. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made." This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

4.1 FUNCTIONAL REQUIREMENTS

The System after careful analysis has been identified to be present with the following modules.

4.1.1 User Module:

The user uploads the image that asked by the model which is in suitable form in terms of png,

jpg, other image forms. The model takes this image and stores in its respective places.

4.1.2 Model Module:

This module maintains all the information to preprocess it according to the training data and

displays it to the user for confirmation. The trained model get imported in-order to predict.

4.1.3 Display Module:

This module is an interface between user and the model. The model gives the accuracy of the different classes that the given image belongs. The module finds the maximum support from that and maps to the corresponding feeling.

4.2 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirements specification plays an important role in system analysis. Systems can be designed for demanding environments only when requirements specifications are clearly defined. The definition of requirements is largely up to the users of the existing system. Because this is the person who will be using the system. This is because you need to know your requirements in advance to design a system to meet them. It is very difficult to change a system once developed, and it is pointless to develop a system that does not meet user requirements.

The requirements specifications for all systems can be summarized as follows:

- The system must be compatible with the existing system.
- The system must be accurate.
- The system must be better than the existing system.
- The current system fully relies on users to fulfill all their obligations.

4.2.1 Software Requirements:

Python : The programming language employed during this project is python. Python is an interpreted, object-

oriented, high-level general-purpose programming language. Python is usually used as a scripting language for web applications. Python is dynamically-typed and garbage-collected. Python follows OOPs concepts. The main aim of OOP in python is to create reusable code. Important OOPs concepts that python follows are class, object, method, polymorphism, encapsulation, data abstraction, inheritance. Python is used in software development, back-end development, data science and writing system scripts among other things. Python supports modules and packages, which encourages program modularity and code reuse. It's also used for research and computing and even has several science-specific libraries or science-friendly. \

- Pip - de facto standard package-management system used to install and manage software packages written in Python.
- OpenCV-python - It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

4.2.3 Libraries used:

TensorFlow: TensorFlow is a free and open-source software library for data flowing symbolic differentiation and differentiable programming across a spread of multiple tasks. TensorFlow could be a python language library for fast numerical computing. It's a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries which simplify the process built on top of TensorFlow. It is a symbolic math library, and it is also used for machine learning applications like neural networks.

NUMPY: NumPy is a library for the Python programming language, adding support for big, multi-dimensional arrays and matrices, together with an oversized collection of high-level mathematical functions to control arrays. NumPy is open-source software. Python binds the widely used computer vision library OpenCV utilizing NumPy arrays to store and operate data.

Sklearn: Scikit-learn also referred to as sklearn is a free software machine learning library for the Python programming language. It's built upon a number of technologies like pandas, NumPy and Matplotlib. Sklearn library features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is meant to interoperate with the Python numerical and scientific libraries NumPy

Matplotlib: Matplotlib is a plotting library for the Python artificial language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots. Matplotlib is meant to be as usable as MATLAB, with the power to use Python, and also the advantage of being free and open-source. There are different types of representations of matplotlib. Some of them are line plot, subplot, histograms, paths, bar plots etc.

Keras: Keras is an open-source software library that has a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras has several implementations of neural network building blocks. Keras is used to convert dataset images into arrays. By converting it to arrays it becomes easy to build a deep learning modelLibraries : OpenCV, Scikit-Learn

4.2.4 IDE :

Google Colab: -Google Colab is a cloud-based integrated development environment (IDE) that allows users to write, execute, and collaborate on Python code in a Jupyter Notebook-like interface. It provides a virtual machine with pre-installed libraries and packages, eliminating the need for local installations and configurations.

4.3 Hardware Requirements:

4.3.1 Processor :

A processor is a logic circuitry which helps in processing the basic instructions in order to 23 drive a computer. A processor is basically a central processing unit (CPU) which is regarded as the most crucial integrated circuitry in a computer. It is responsible for interpreting most of the computer's commands. The basic elements of a processor include the arithmetic logic unit (ALU), the floatingpoint

unit (FPU), Registers, L1 and L2 cache memory which collectively perform most basic arithmetic, logic and I/O operations, as well as allocate commands for other chips and components running in a computer.

4.3.2 Hard Disk:

512MB (Minimum): The hard disk is generally a data storage device that stores and retrieves data using magnetic storage. Data is stored on their surfaces in concentric tracks. The Hard disk drive was introduced in 1956 by IBM. A computer's hard drive is a device consisting of several hard disks, read/write heads, a drive motor to spin the disks, and a small amount of circuitry, all sealed in a metal case to protect the disks from dust. RAM: 4GB (Minimum): RAM is an abbreviation for Random Access Memory, which is mainly responsible for storing data, program and program result. The Ram is also referred to as main memory, primary memory or system's memory. The RAM is of two types, Static RAM (SRAM) and Dynamic RAM (DRAM). The data stored in the memory is only present until the machine is working, as soon as the machine is switched off, the whole data is erased (Volatile memory).

4.3.3 Operating System- Windows 7/8/10:

Operating System is an intermediate system between computer user and computer hardware. Operating systems provide an environment where a user can execute software programs conveniently and efficiently. The Operating System also manages the hardware part of the computer. The hardware should supply suitable mechanisms to ensure the right operation of the computer system and to avoid programs of the users from interfering with actual operation of the system.

5. SYSTEM DESIGN

System design refers to the process of defining the architecture, components, interfaces, and behavior of a software system. It involves transforming the requirements and specifications into a detailed design that can be implemented. UML diagrams, including use case diagrams, are commonly used in system design to visually represent the system's structure and behavior.

5.1 UML DIAGRAMS

UML (Unified Modeling Language) is a standardized modeling language used for software development. It provides a set of graphical notations to represent various aspects of a system. Use case diagrams, one of the UML diagrams, depict the interactions between actors (users or external systems) and the system itself

5.1.1 Usecase diagram

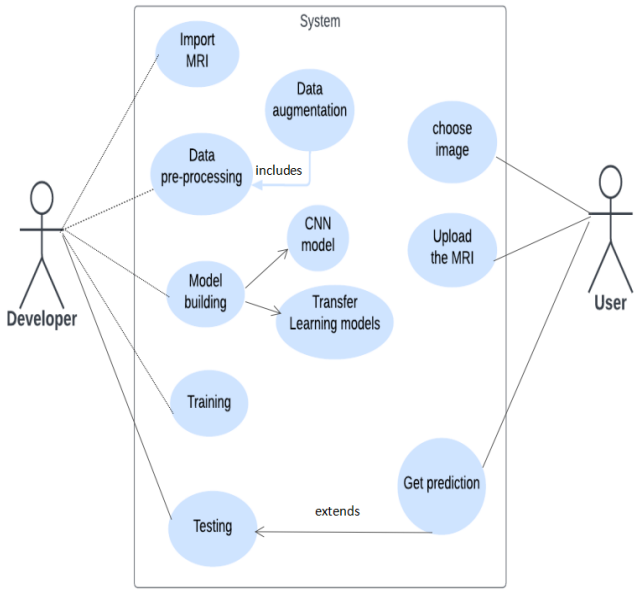


Fig 5.1 Usecase diagram

A use case diagram is a visual representation that helps identify the different scenarios and interactions between the users and the system being developed. For a brain tumor detection project, it can help to identify the users, define the different scenarios in which the system will be used, and prioritize the development of functionalities based on their importance. In short, a use case diagram is a valuable tool for designing and developing a brain tumor detection system that meets the needs of its users.

5.1.2. Class diagram

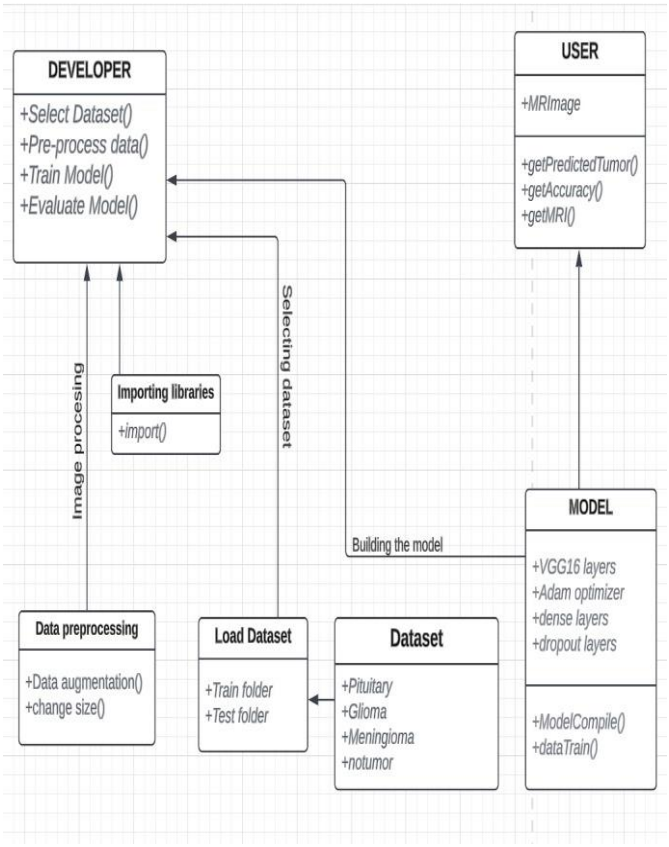
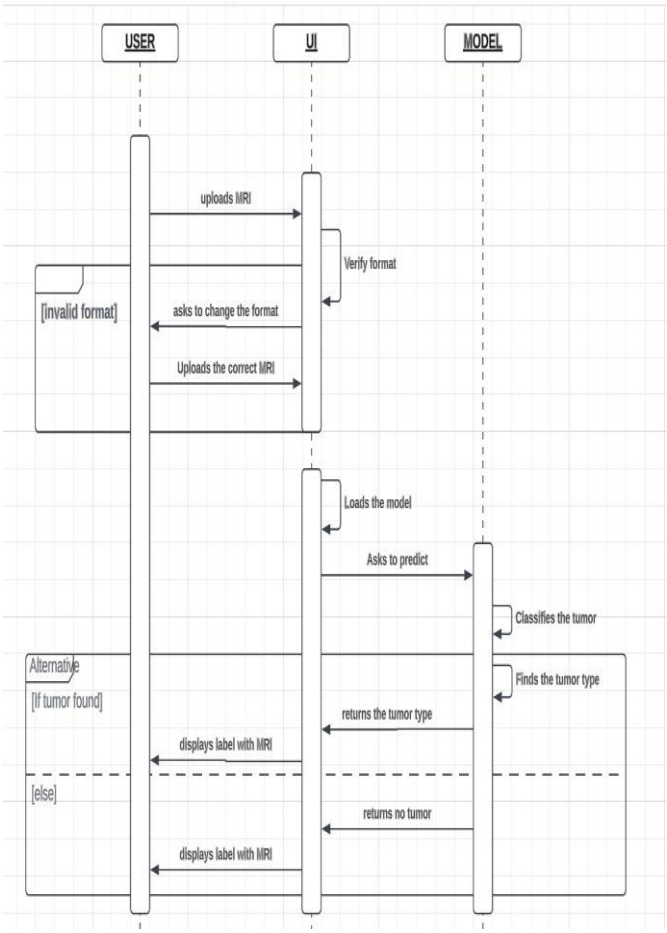


Fig 5.2 Class diagram

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the static structure of a system by showing the classes, their attributes, methods, and relationships among them. It is commonly used in software engineering to visualize and design the structure of a software system.

Overall, a class diagram provides a high-level overview of a system's structure and helps to identify the major components and their relationships, which is helpful for software design, analysis, and development.

5.1.3. Sequence diagram



5.3 Sequence diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that illustrates how objects interact with each other in a system over time.Here,the sequence diagram helps to visualize the flow of information and actions involved in the detection process.The diagram shows how the user sends image data to the computer algorithm, which then processes the data to detect the presence of a tumor.

6. SYSTEM IMPLEMENTATION

System implementation refers to the process of translating a conceptual system design into a functioning software or hardware system. It involves the actual development, configuration, and deployment of the system components. In the context of machine learning and data science, system implementation often includes incorporating workflow management, data augmentation techniques, and data generation methods

6.1 WORKFLOW

It involves the systematic movement of information, resources, efficient execution of work. Workflow management aims to streamline and optimize these processes, ensuring that work progresses in a logical and organized manner to achieve desired outcomes.

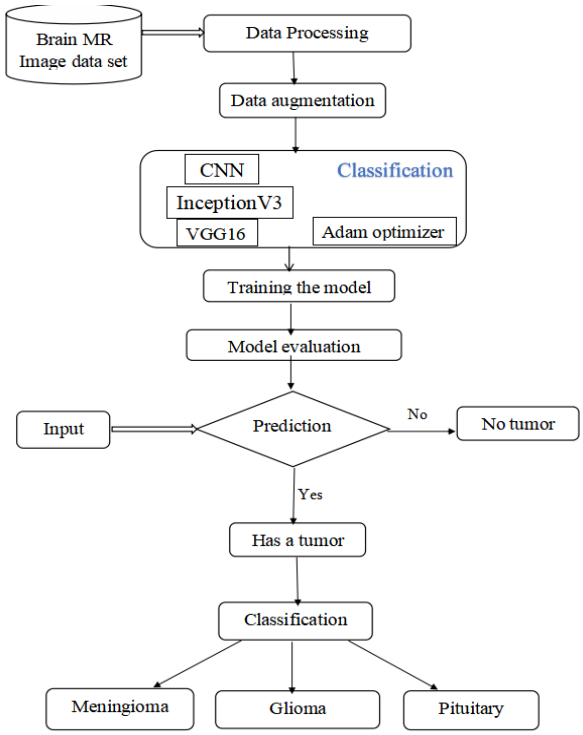


Fig 6.1 System Workflow

6.2 DATASET

The dataset used contains 7023 images under the training and testing data. This dataset contains 1311 MR images of brain tumors that are used to test the model and 5712 MR images of brain tumors are used to train the model. These images used here are of 4 different types.i.e.,the MR Image folder with no brain tumor, the Glioma brain tumor, the Meningioma brain tumor and the Pituitary brain tumor.

The dataset contains 27.9% of no tumor MR images,23.1% of Glioma tumor MR images, 23.4% of meningioma tumor MR images and 25.5% of pituitary MR images.so,the dataset is reasonably balanced.

The dataset is taken from kaggle (https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset).

6.3 DATA PREPROCESSING

6.3.1 Data augmentation

Data augmentation involves creating new data samples by applying metamorphoses to being data samples. This can help to increase the size and diversity of the data set, which can be particularly useful when the available data is limited or imbalanced. Some common ways used in data addition include image flipping, gyration, cropping, zooming, and color adaptations.[21] Data augmentation can help to ameliorate the conception and performance of machine learning models, reduce overfitting, and make the models more robust and adaptable to new data. It can also help to reduce the need for collecting further data, which can be time-consuming and precious. Images are comprised of matrices of pixel values.

Black and white images are single matrix of pixels, whereas color images have a separate array of pixel values for each color channel, similar as red, green, and blue. Pixel values are frequently unsigned integers in the range between 0 and 255. Although these pixel values can be presented directly to neural network models in their raw format, this can affect in challenges during modeling, similar as in the slower than anticipated training of the model. rather, there can be great benefit in preparing the image pixel values previous to modeling, similar as simply spanning pixel values to the range 0- 1 to centering and indeed homogenizing the values. This can be achieved by dividing all pixels values by the largest pixel value; that's 255. Although the brilliance, discrepancy and sharpness of the image.

6.3.2 Data Generation

This defines three functions for data processing and a generator function for loading the data in batches.

The `encode_label` function takes a list of labels and converts each label to its corresponding index in `unique_labels`, which is a list of the unique labels in the dataset.

The `decode_label` function takes a list of label indices and converts each index back to its corresponding label in `unique_labels`.

The `datagen` function is a generator function that takes in a list of file paths (`paths`), a list of labels (`labels`), a batch size (`batch_size`), and the number of epochs (`epochs`). It loops over the number of epochs and generates batches of data with the specified batch size. For each batch, it loads the corresponding images using `open_images`, encodes the labels using `encode_label`, and yields the batch as a tuple of `(images, labels)`.

This generator function is useful for loading batches of data in a memory-efficient way, which is especially important for large datasets that cannot fit into memory all at once. It also allows for data augmentation and label encoding to be applied to the data in real-time during training.

6.4 CNN MODEL

The `build_model` function defines and returns a convolutional neural network (CNN) model using the Keras API with a `Sequential` model.[7] The model contains a series of layers that perform convolution, pooling, and dense operations to extract and classify features from the input image data.

The model architecture contains the following layers:

- A `Conv2D` layer with 32 filters, each with a size of 3x3 and ReLU activation function. This layer takes in an input image with shape (IMAGE_SIZE, IMAGE_SIZE, 3) and produces an output tensor with a shape of (IMAGE_SIZE-2, IMAGE_SIZE-2, 32), since the filters are applied without padding, resulting in a reduction in output shape due to the loss of pixels at the edges.

- A `MaxPooling2D` layer with a pool size of 2x2, which performs down-sampling of the input image along the spatial dimensions.

- A `Conv2D` layer with 64 filters and a 3x3 filter size with ReLU activation function.

- Another `MaxPooling2D` layer.

- A `Conv2D` layer with 32 filters and a 3x3 filter size with ReLU activation function.

- A `Flatten` layer, which flattens the output tensor of the previous layer into a 1D array for input into the dense layers.

- A `Dense` layer with 16 units and a ReLU activation function.

- Another `Dense` layer with 4 units and a softmax activation function, which outputs the class probabilities for the four types of brain tumors.

The model is compiled with the `adam` optimizer, `sparse_categorical_crossentropy` loss function, and `accuracy` as the evaluation metric.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 126, 126, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 28, 28, 32)        18464

 flatten_2 (Flatten)         (None, 25088)             0

 dense_4 (Dense)             (None, 16)                401424

 dense_5 (Dense)             (None, 4)                 68

=================================================================
Total params: 439,348
Trainable params: 439,348
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.2 CNN Summary

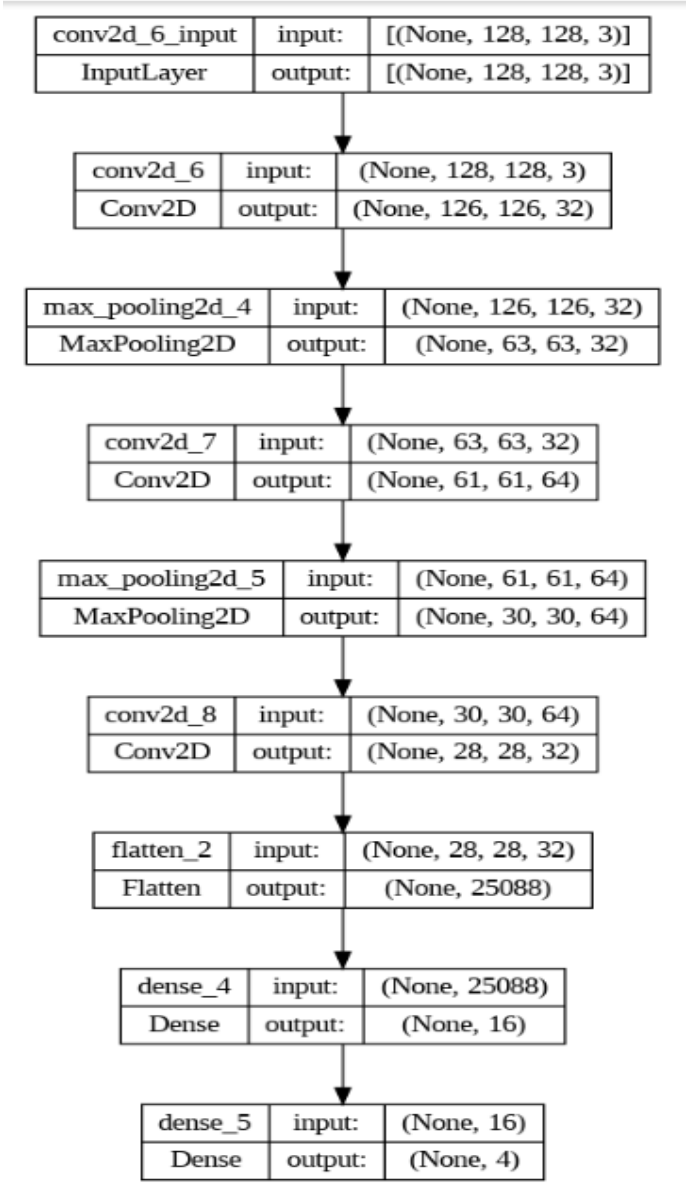


Fig 6.3 CNN Architecture

6.5 TRANFER LEARNING

In deep learning, sometimes we use a transfer learning approach in which instead of making a scratched CNN model for the image classification problem, a pre-trained CNN model that is already modeled on a huge benchmark dataset like ImageNet is reused. Instead of starting the learning process from scratch, the transfer learning leverages previous learning. [6] It is a powerful technique used in machine learning that involves taking a pre-trained model and modifying it to fit a new task. It has become increasingly popular in recent years due to its ability to save time and computational resources, while still achieving high accuracy in the new task.

6.5.1 VGG16 model

This builds a new neural network model using the VGG16 model as a base.

First, the VGG16 model is loaded with pre-trained weights from ImageNet and all layers are set to non-trainable [17]. Then, the last three layers of the VGG16 model are set to trainable, so that they can be fine-tuned on a new classification task.

Next, a new model is defined using the Sequential API. The first layer of the new model is an input layer that expects image data with shape `(IMAGE_SIZE, IMAGE_SIZE, 3)` (3 channels for RGB color). The second layer is the pre-trained VGG16 model, which processes the input image and extracts useful features.

The third layer is a `Flatten` layer, which converts the output of the previous layer into a one-dimensional array. The fourth layer is a `Dropout` layer, which randomly drops some of the output units of the previous layer during training, to prevent overfitting.

The fifth layer is a `Dense` layer with 128 units and `relu` activation function, which is a common choice for intermediate layers in neural networks. The sixth layer is another `Dropout` layer. Finally, the last layer is a `Dense` layer with as many units as the number of unique labels in the classification task, and a `softmax` activation function, which outputs a probability distribution over the labels.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten_3 (Flatten)         (None, 8192)              0

 dropout (Dropout)           (None, 8192)              0

 dense_6 (Dense)             (None, 128)               1048704

 dropout_1 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 4)                 516

=================================================================
Total params: 15,763,908
Trainable params: 8,128,644
Non-trainable params: 7,635,264
_________________________________________________________________
```

Fig 6.4 VGG16 Summary

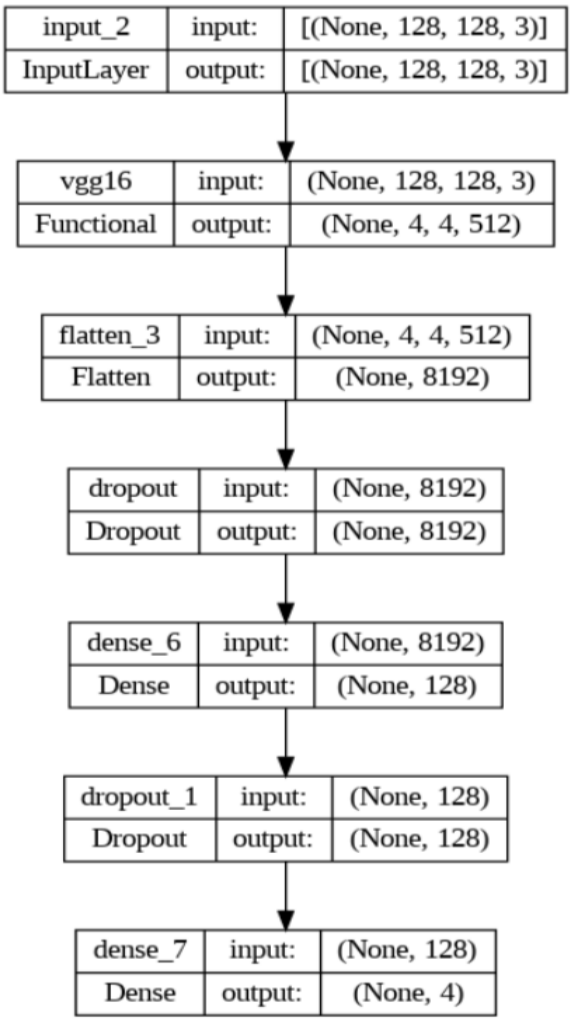


Fig 6.5 VGG16 Architecture

6.5.2 InceptionV3 model:

Inception v3 is a deep convolutional neural network architecture that uses a combination of convolutional, pooling, and normalization layers to learn features from input images. The network consists of several Inception modules that allow it to extract features at different spatial scales and resolutions.

The key innovation in Inception v3 is the use of "Inception modules," which are multi-branch convolutional networks with different filter sizes.[20] These modules can capture a wide range of feature sizes and help reduce the number of parameters in the model. The Inception modules also include 1x1 convolutions, which can help reduce the computational cost of the network.

Inception v3 also uses several other techniques to improve performance, such as:

1.Factorizing convolutions: Instead of using large convolutions, Inception v3 uses factorized convolutions, which break down the convolution into smaller convolutions, reducing the number of parameters and improving efficiency.

2.Batch normalization: Normalizing the inputs to each layer can help improve training stability and speed up convergence.

3.Regularization: Inception v3 uses L2 regularization to reduce overfitting.

4.Auxiliary classifiers: The network includes two auxiliary classifiers that provide additional supervision during training and can improve performance.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 2, 2, 2048)        21802784

 flatten_4 (Flatten)         (None, 8192)              0

 dropout_2 (Dropout)         (None, 8192)              0

 dense_8 (Dense)             (None, 128)               1048704

 dropout_3 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 256)               33024

 dropout_4 (Dropout)         (None, 256)               0

 dense_10 (Dense)            (None, 4)                 1028

=================================================================
Total params: 22,885,540
Trainable params: 1,082,756
Non-trainable params: 21,802,784
_________________________________________________________________
```

Fig 6.6 InceptionV3 Summary

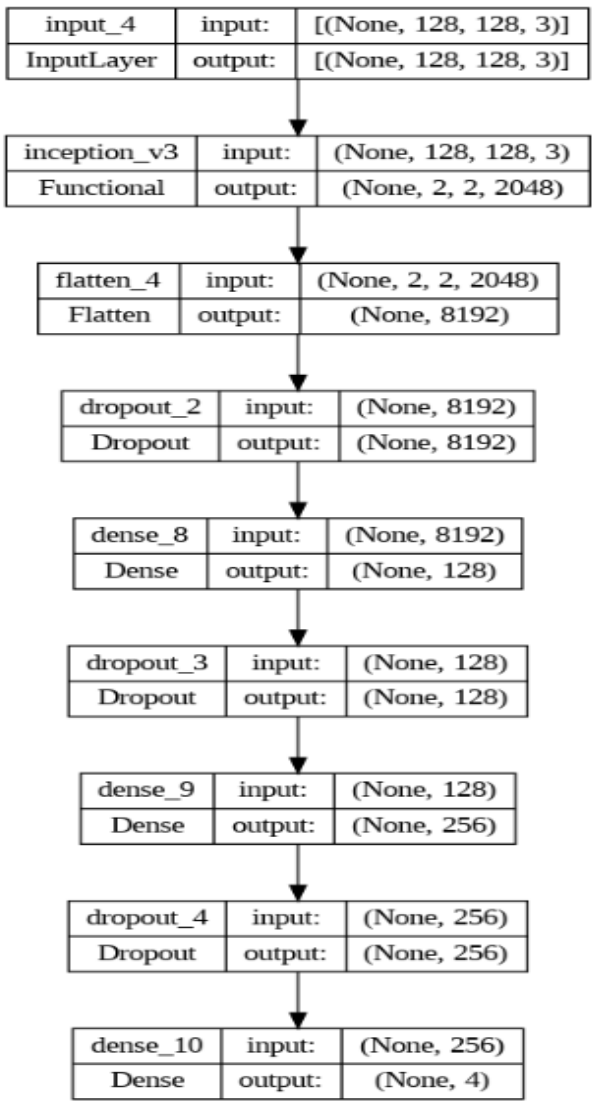


Fig 6.7 InceptionV3 Architechture

6.6 MODEL COMPILATION

For the Keras model, I used the following parameters by setting the optimizer, loss function, and metric to be used during training.

Optimizer: This parameter defines the optimization algorithm to be used during training. In this case, the Adam optimizer with a learning rate of 0.0001 is used. The Adam optimizer is a popular optimization algorithm that uses adaptive learning rates and momentum to converge faster than other optimization methods.

Loss: This parameter defines the loss function to be used during training. In this case, the sparse_categorical_crossentropy loss function is used. This is a common loss function used for multi-class classification problems where the target values are integers. It computes the crossentropy loss between the predicted class probabilities and the true class labels.

metrics: This parameter specifies the evaluation metric to be used during training and testing. In this case, the sparse_categorical_accuracy metric is used. This metric calculates the accuracy of the model predictions, which is the percentage of correct predictions out of all predictions made. After the model is compiled, it is ready to be trained with the training data using the fitting method[19]. During training, the optimizer adjusts the model's weights and biases to minimize a given loss function. Metrics specified at compile time are calculated and displayed after each training epoch to monitor model performance.

6.7 MODEL TRAINING

Here the model uses the fit() function, which takes in the following arguments:

datagen(train_paths, train_labels, batch_size=batch_size, epochs=epochs): This argument specifies the data generator function that generates batches of training data. The datagen() function generates batches of images and their corresponding labels using the open_images() function to read the image files and the encode_label() function to convert the label strings to integer-encoded labels. epochs=epochs: This argument specifies the number of epochs for which the model will be trained.

steps_per_epoch=steps: This argument specifies the number of batches to be processed per epoch. Since the datagen() function generates batches of size batch_size, the number of steps per epoch is computed as the total number of training samples divided by the batch size.

During training, the model will iterate over the batches of training data generated by the datagen() function and update its weights based on the specified optimizer and loss function in the model.compile() function. The fit() function also records the training and validation metrics specified in the model.compile() function for each epoch in a history object, which can be used for visualization and analysis.

Here,we used a batch size of 32 that allows the model to process 32 samples at once during training, reducing memory usage and enabling efficient training on a large

dataset. The datagen() function generates batches of data on the fly, further optimizing memory usage and enabling training on large datasets that may not fit into memory

6.8 MODEL EVALUATION

Evaluation is an important step in machine learning to assess the performance of a trained model on unseen data. It helps us to determine if the model has learned to generalize well to new data or if it is overfitting to the training data.

We evaluate the performance of the trained model on a test set by calculating the predicted labels and comparing them with the true labels. This allows us to calculate various performance metrics such as accuracy, precision, recall, and F1 score.

The batch_size parameter determines the number of test samples processed in each batch. Using batches helps to reduce memory usage and speeds up the evaluation process. The steps variable calculates the total number of batches needed to process all the test data.

By using the datagen() function to generate batches of test data on the fly, we can efficiently evaluate the model's performance without having to load all the test data into memory at once. This is particularly useful when dealing with large datasets that cannot fit into memory.

The predicted output from the model is a probability distribution over all possible classes, which is converted to class labels using the np.argmax() function. These predicted labels are then appended to the y_pred list.

The true labels for each batch of test data are obtained by decoding the encoded labels using the decode_label() function. These true labels are then appended to the y_true list.

At the end of the loop, the y_pred and y_true lists contain the predicted and true labels for all test data. These lists can then be used to calculate evaluation metrics such as accuracy, precision, recall, and F1 score.

Overall, the evaluation step is crucial to ensure that the trained model is performing well on new data and is a necessary step before deploying the model in a real-world application.[21] The efficiency of the evaluation process is improved by using batch processing and on-the-fly data generation, which can help reduce memory usage and speed up the evaluation process.

The metrics we have used for the evaluation of our model are:

$$\text{Accuracy} = (TN+TP)/(TN+FP+FN+TP) \quad (1)$$

$$\text{Precision} = TP/(FP+TP) \quad (2)$$

$$\text{Recall} = TP/(TP+FN) \quad (3)$$

$$\text{F1 score} = TP/(TP+1/2(FP+FN)) \quad (4)$$

Where,

TP = True-positive,

TN = True-negative,

FP = False- positive,

FN = False-negative

In the above formulas, true positive value refers to the images which are initially labelled as true and are also true after the prediction of the model. Here, true refers to images with tumor and false refers to images without tumor. False-positive value refers to the images which are initially labelled as true and becomes false after the prediction by the model. False-positive value refers to the images which are initially labelled as false and are also false after the prediction of the model. False-negative value refers to the images which are initially labelled as false and are true after the prediction of the model.

7. RESULTS

The result here explains how the model is evaluated by calculating various metrics like accuracy, precision, recall and f1 score. It also shows a graphical representation of accuracy vs loss which illustrates the accuracy of our model.

7.1 PREDICTING THE MRI

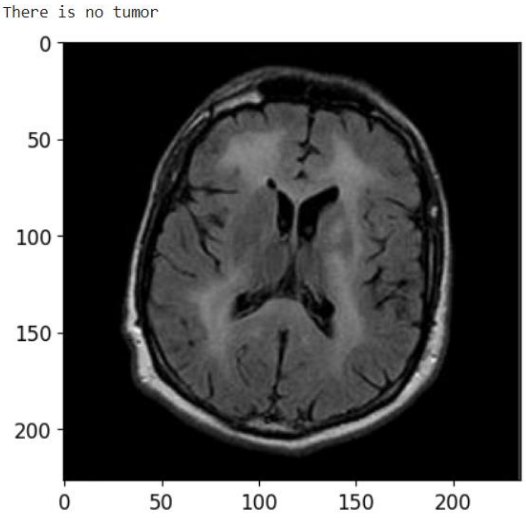


Fig 7.1 MR Image predicted as no tumor

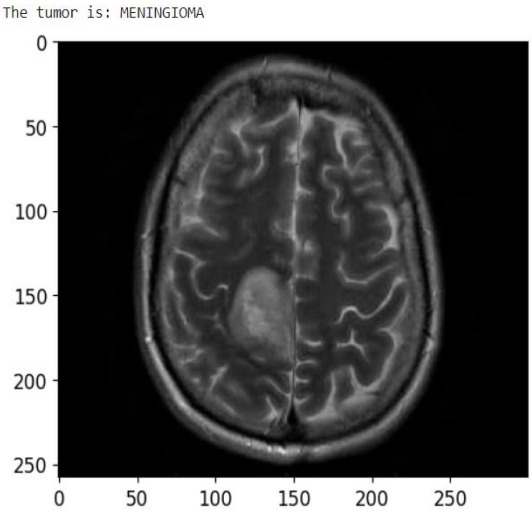


Fig 7.2 MR Image predicted as Meningioma

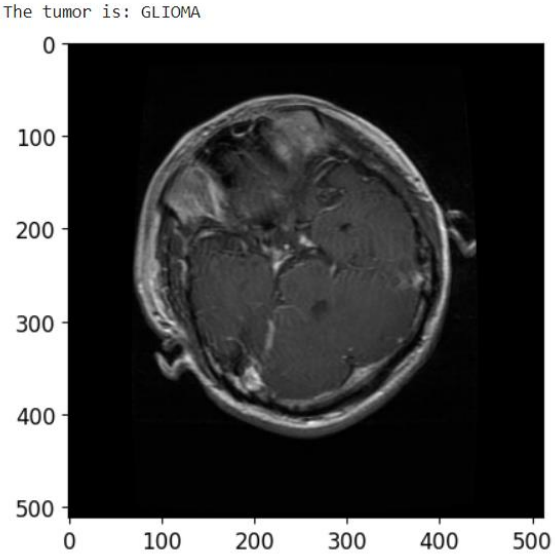


Fig 7.3 MR Image predicted as Glioma

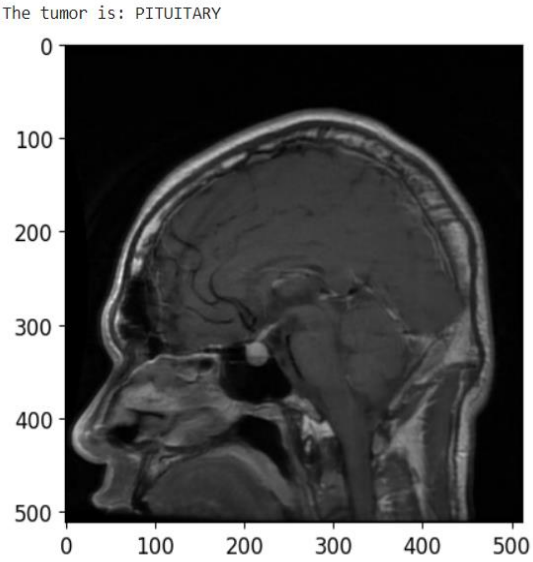


Fig 7.4 MR Image predicted as Pituitary

The MR image is predicted correctly as expected. The image is displayed after the prediction with the label on the top of the image as result.

7.2 MODEL TRAINING HISTORY

This is the graph displaying the Accuracy and loss of the model with the red colored curve for loss and green colored graph for the accuracy. Our model achieved a high accuracy compared to other models.

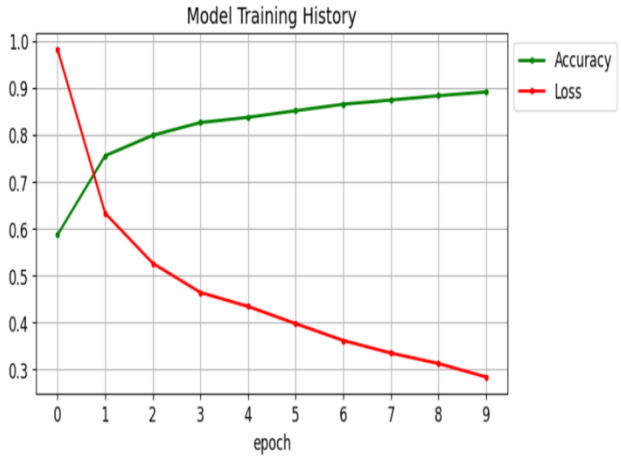


Figure 7. CNN Model Training

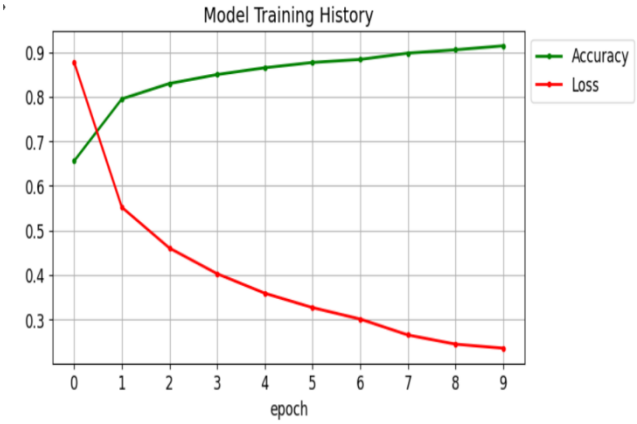


Figure 8. Inception V3 Model Training

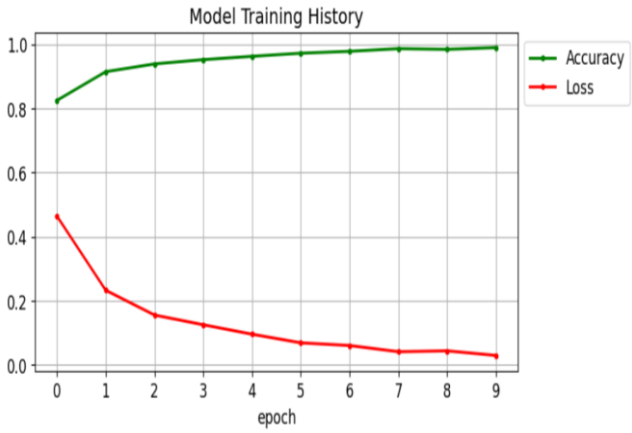


Figure 9. VGG-16 Model Training

Here, in our system ,we trained the model using 10 epochs and achieved an overall accuracy of 98% on testing 1311 MR Images, i.e., 300 glioma MR images,306 meningioma MR images,405 no tumor images and 300 pituitary images. We got the highest accuracy for pituitary which is 100%.

	precision	recall	f1-score	support
glioma	0.96	0.96	0.96	300
meningioma	0.95	0.96	0.95	306
notumor	1.00	1.00	1.00	405
pituitary	0.99	0.98	0.99	300
accuracy			0.98	1311
macro avg	0.98	0.98	0.98	1311
weighted avg	0.98	0.98	0.98	1311

Figure 10. VGG-16 Classification Report

7.3 CLASSIFICATION REPORT

Here, in our system ,we trained the models using 10 epochs and achieved an overall accuracy of 98% for VGG16 on testing 1311 MR Images, i.e., 300 glioma MR images,306 meningioma MR images,405 no tumor images and 300 pituitary images. We got the highest accuracy for pituitary which is 100%.

	precision	recall	f1-score	support
meningioma	0.75	0.78	0.76	306
pituitary	0.93	0.93	0.93	300
notumor	0.95	0.92	0.94	404
glioma	0.82	0.82	0.82	300
accuracy			0.87	1310
macro avg	0.86	0.86	0.86	1310
weighted avg	0.87	0.87	0.87	1310

Fig 7.8 CNN Classification Report

	precision	recall	f1-score	support
glioma	0.93	0.83	0.88	300
meningioma	0.81	0.89	0.85	306
notumor	0.98	0.97	0.98	404
pituitary	0.91	0.95	0.93	300
accuracy			0.91	1310
macro avg	0.91	0.91	0.91	1310
weighted avg	0.92	0.91	0.91	1310

Fig 7.9 InceptionV3 Classification Report

	precision	recall	f1-score	support
glioma	0.95	0.98	0.97	300
meningioma	0.97	0.95	0.96	306
notumor	0.99	1.00	1.00	404
pituitary	0.99	0.98	0.99	300
accuracy			0.98	1310
macro avg	0.98	0.98	0.98	1310
weighted avg	0.98	0.98	0.98	1310

Fig 7.10 VGG16 Classification Report

8. CONCLUSION

The CNN(Convolution neural network)model and the transfer learning models VVG16 and InceptionV3 are built and the corresponding results are compared.

The proposed model built is a convolutional neural network (CNN) based on the VGG16 architecture, with a dense classification layer on top. The model is trained using image data for classifying brain tumor images into different types.

During the training process, the model is compiled using the Adam optimizer, sparse categorical crossentropy loss, and sparse categorical accuracy metric. The training is performed using a generator that batches and augments the image data to improve the model's performance. The trained model is then evaluated on a test set to measure its accuracy in classifying brain tumor images.

This system is designed to classify brain tumor images into four categories (no tumor, pituitary tumor, meningioma tumor, glioma tumor) using the VGG16 convolutional neural network architecture. The model was trained using a dataset of brain MRI images and achieved an accuracy of 98% on the test set, which indicates that the model performs well in classifying the different types of tumors.

In conclusion, this model can be a useful tool in assisting radiologists and medical practitioners in diagnosing brain tumors and making informed decisions about treatment options for patients.

9. FUTURE SCOPE

Detection of brain tumors in real-time can be achieved through various imaging techniques such as magnetic resonance imaging (MRI), computed tomography (CT), and positron emission tomography (PET) scans.

MRI is the most commonly used imaging technique for detecting brain tumors. It uses a magnetic field and radio waves to create detailed images of the brain. CT scans use X-rays to create images of the brain, and PET scans use a radioactive tracer to show how the brain is functioning.

Real-time detection of brain tumors can be helpful for doctors in several ways. It allows them to quickly identify the location and size of the tumor, which can help guide treatment decisions. Realtime imaging can also be used during surgery to help surgeons precisely remove the tumor while minimizing damage to healthy brain tissue.

In addition, real-time monitoring of the tumor during and after treatment can help doctors evaluate the effectiveness of the treatment and make any necessary adjustments. Overall, real-time detection of brain tumors is a valuable tool in the diagnosis and treatment of brain tumors.

REFERENCES:

[1] A. Çinar and M. Yildirim, ‘‘Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture,’’ Med. Hypotheses, vol. 139, Jun. 2020, Art. no. 109684.

[2] H. M. Rai and K. Chatterjee, ‘‘Detection of brain abnormality by a novel Lu-Net deep neural CNN model from MR images,’’ Mach. Learn. Appl., vol. 2, Dec. 2020, Art. no. 100004.

[3] M. K. Islam, M. S. Ali, M. S. Miah, M. M. Rahman, M. S. Alam, and M. A. Hossain, ‘‘Brain tumor detection in MR image using superpixels, principal component analysis and template based K-means clustering algorithm,’’ Mach. Learn. Appl., vol. 5, Sep. 2021, Art. no. 100044

[4] T. K. Das, P. K. Roy, M. Uddin, K. Srinivasan, C.-Y. Chang, and S. Syed-Abdul, ‘‘Early tumor diagnosis in brain MR images via deep convolutional neural network model,’’ Comput., Mater. Continua, vol. 68, no. 2, pp. 2413–2429, 2021.

[5] M. Toğaçar, B. Ergen, and Z. Cömert, ‘‘BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model,’’ Med. Hypotheses, vol. 134, Jan. 2020, Art. no. 109531.

[6] Z. Jia and D. Chen, ‘‘Brain tumor identification and classification of MRI images using deep learning techniques,’’ IEEE Access, early access, Aug.13,2020,doi:10.1109/ACCESS.2020.3016319.

[7] R. Mehrotra, M. A. Ansari, R. Agrawal and R. S.Anand, “A Transfer Learning approach for AI-based classification of brain tumors,” Mach. Learn. Appl. 2020, 2, 10–19.

[8] K. Kaplan, Y. Kaya, M. Kuncan and H.M. Ertunç,” Brain tumor classification using modified local binary patterns (LBP) feature extraction methods,” Med. Hypotheses 2020, 139, 109696.

[9] J.P. Francisco, Z.M. Mario and R.A. Miriam, “ A Deep Learning Approach for Brain Tumor Classification and Segmentation Using a Multiscale Convolutional Neural Network,” Healthcare 2021, 9, 153.

[10] Z. Ullah, M.U. Farooq, S.H. Lee and D. An, “A Hybrid Image Enhancement Based Brain MRI Images Classification Technique,” Med. Hypotheses 2020, 143, 109922.

[11] S.Y. Lu, S.H. Wang and Y.D. Zhang, “ A classification method for brain MRI via MobileNet and feedforward network with random weights,” Pattern Recognit. Lett. 2020, 140, 252–260

[12] P.G. Rajan and C. Sundar,” Brain tumor detection and segmentation by intensity adjustment,”J. Med. Syst. 2019, 43, 1–13.

[13] S. Preethi and P. Aishwarya, “Combining Wavelet Texture Features and Deep Neural Network for Tumor Detection and Segmentation Over MRI,”. J. Intell. Syst. 2019, 28, 571–588.

[14] D.J. Hemanth, J. Anitha, A. Naaji, O. Geman and D.E. Popescu, “A modified deep convolutional neural network for abnormal brain image classification”, IEEE Access 2018, 7, 4275–4283

[15] C. Shorten, T. M. Khoshgoftaa, “A survey on image data augmentation for deep learning,” J. Big Data, 6 (2019), 60.

[16] A. Rehman, S. Naz, M. I. Razzak, F. Akram, M. Imran, “A Deep Learning-Based Framework for Automatic Brain Tumors Classification Using Transfer Learning,” Circuits Syst. Signal Process., 39 (2020), 757–775.

[17] M. Talo, U. B. Baloglu, O. Yldrm, U. R. Acharya, “Application of deep transfer learning for automated brain abnormality classification using MRI images, “Cognitive Systems Research, 54 (2019), 176–188.

[18] S. Basheera, M. S. S. Ram, “Classification of brain tumors using deep features extracted using CNN,” J. Phys., 1172 (2019), 012016.

[19] N. Abiwinanda, M. Hanif, S. Hesaputra, A. Handayani, T. R. Mengko, “Brain tumor classification using convolutional neural network,

"World Congress on Medical Physics and Biomedical Engineering, Springer, Singapore, 2019.

[20] V. Romeo, R. Cuocolo, C. Ricciardi, L. Ugga, S. Cocozza, F. Verde, et al,"Prediction of tumor grade and nodal status in oropharyngeal and oral cavity squamous-cell carcinoma using a radiomic approach, Anticancer Res.," 40 (2020), 271–280.

[21]] A. Gudigar, U. Raghavendra, T. R. San, E. J. Ciaccio, and U. R. Acharya, ''Application of multiresolution analysis for automated detection of brain abnormality using MR images: A comparative study,'' Future Gener. Comput. Syst., vol. 90, pp. 359–367, Jan. 2019.