A Project Report

BRAIN TUMOR DETECTION

Submitted in partial fulfillment of the requirements for the award of the degree

BACHELOR OF ENGINEERING

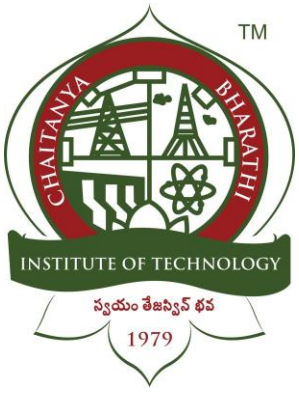
in

COMPUTER SCIENCE AND ENGINEERING

by

Rahul Mora(160116733045)

Syed Nayaz Moinuddin(160116733059)



Department of Computer Science and Engineering,

Chaitanya Bharathi Institute of Technology (Autonomous),

(Affiliated to Osmania University, Hyderabad)

Hyderabad, TELANGANA (INDIA) –500 075

[2019-2020]

CERTIFICATE

This is to certify that the project titled "**BRAIN TUMOR DETECTION** " is the bonafide work carried out by **RAHUL MORA (160116733045), SYED NAYAZ MOINUDDIN (160116733059)** students of B.E.(CSE) of Chaitanya Bharathi Institute of Technology(A), Hyderabad, affiliated to Osmania University, Hyderabad, Telangana(India) during the academic year 2019-2020, submitted in partial fulfillment of the requirements for the award of the degree in **Bachelor of Engineering (Computer Science and Engineering)** and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title. 2019-2020.

Supervisor
Ms A.Sangeetha

Head,CSE Dept
Dr. M Swamy Das

Place: HYDERABAD
Date: 05 May 2020

DECLARATION

We hereby declare that the project entitled “Brain Tumor Detection” submitted for the B.E(CSE) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

M. Rahul
(160116733045)

Syed Nayaz Moinuddin
(160116733059)

Place: Hyderabad

Date: 05 May 2020

ABSTRACT

Brain Tumour is an abnormal growth of cells inside the brain cranium which limits the function of the brain. The brain tumor can be treated with Chemotherapy i.e., by sending as anti-carcinogenic chemicals into the body through blood vessels, Radiation i.e., By killing the cancer cells by emitting Radioactive ways, and also by surgery to remove the tumor. But all these procedures are only desirable when the tumor is in its initial stages. When the tumor matures, the carcinogenic cells move to different body parts and form tumors there. So, It is indispensable to find out a tumor when it is in its initial stages only.

Over all 700,000 people are living with a brain tumor. Among them 68.2% are benign and 32.8% are malignant. Among malignant tumors, the survival rate is only 36% which implies that among every 3 people having a malignant tumor only one person has a fair chance of living. Moreover there is no significant cause that could lead to brain tumor. It would be a hectic process for a doctor or any medical representative to go through a huge number of MRI images to detect whether a tumor is present or not. So, we tried to build a system which detects the brain tumor in a MRI scanned image.

Most of the existing systems concentrated on building a system that can segment the tumor rather than detecting the tumor. But, segmentation of the tumor could be more efficient when an image with the tumor is fed to the system. In the paper Predictive modeling of brain tumor: A Deep learning approach[3], it was showed that when VGG-16, Resnet-52, and Inception-v3 models were used to detect a tumor in a MRI scanned image, the accuracies obtained are 90%, 95%, and 55% respectively.

We have attempted to train the dataset obtained from Kaggle using Inception-Resnet-v2 architecture. Inception-Resnet-v2 is one of the models having the least error rates over ImageNet dataset beating the human error rate as it combines both Inception and Resnet models.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crown all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project. We would like to express our sincere gratitude and indebtedness to our project guide, Smt. A. Sangeetha, Asst. Professor who has supported us throughout our project with patience and knowledge. We are also thankful to the Head of the department **Dr. M. Swamy Das** for providing excellent infrastructure and a conducive atmosphere for completing this project successfully. We are also extremely thankful to our Project coordinators **Dr. K. Sagar**, Professor and **Smt E. Kalpana**, Asst. Professor, Dept. of CSE, for their valuable suggestions and interest throughout the course of this project. We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed. We sincerely acknowledge and thank all those who gave directly or indirectly their support in the completion of this work.

M. Rahul (160116733045)

Syed Nayaz Moinuddin (160116733059)

LIST OF FIGURES

LIST OF TABLES

PAGE INDEX

1. INTRODUCTION

1.1 Problem Definition including the significance and objective:

Brain Tumour is an abnormal growth of cells inside the brain cranium which limits the function of the brain. The brain tumor can be treated with Chemotherapy i.e., by sending in anti-carcinogenic chemicals into the body through blood vessels, Radiation i.e., By killing the cancer cells by emitting Radioactive ways, and also by surgery to remove the tumor. But all these procedures are only desirable when the tumor is in its initial stages. When the tumor matures, the carcinogenic cells move to different body parts and form tumors there. So, it is indispensable to find out a tumor when it is in its initial stages only.

Magnetic Resonance Imaging (MRI) is a medical imaging technique for obtaining detailed images of the human body. The images of the brain obtained from MRI scans can be used to detect brain Tumor by using various image processing techniques.

1.2 Methodologies

Object Detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Data Augmentation

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Training deep learning neural network models on more data can result in more skillful models. The

number of images can be increased by performing various operations like rotation, shearing, zooming, changing the brightness level or flipping.

Denoising

The goal of denoising is to estimate the original image by suppressing noise from a noise-contaminated version of the image. Image noise may be caused by different intrinsic and extrinsic conditions which are often not possible to avoid in practical situations. Therefore, image denoising plays an important role in a wide range of applications such as image restoration, image registration, image segmentation, and image classification, where obtaining the original image content is crucial for strong performance.

Stochastic Gradient Descent:

Gradient Descent can be described as an iterative method that is used to find the values of the parameters of a function that minimizes the cost function as much as possible. The parameters are initially defined as a particular value and from that, Gradient Descent is run in an iterative fashion to find the optimal values of the parameters, using calculus, to find the minimum possible value of the given cost function. In SGD, the gradient of the cost function of a single example at each iteration is found out instead of the sum of the gradients of the cost function of all the examples. In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with significantly shorter training time.

Dropout:

Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of

downstream neurons is temporarily removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

1.3 Outline of the results

The performance of the optimized custom CNN and pretrained models are evaluated toward the challenge of classifying MRI images with tumor and without tumor. The model is trained and optimized to minimize the categorical crossentropy loss and categorize the MRI images to their respective classes. The image undergoes some transformations and the trained CNN will be used to determine whether the tumor is present in the image or not. To evaluate the reliability of this model the following parameters were used: Mean Square Error (MSE) and accuracy.

1.4 Scope of the project

Brain tumour is an abnormal or uncontrolled growth of cells found in the brain. In a healthy human body, normal cells grow old or die and new cells take their place. Sometimes, this process goes wrong. Some unnecessary new cells are produced when the body doesn't need them, and old or damaged cells don't die as they should. The formation of these extra cells often forms a mass of tissue called a growth or tumour.

This can be treated with Chemotherapy i.e., by sending in anti-carcinogenic chemicals into the body through blood vessels, Radiation i.e., By killing the cancer cells by emitting Radioactive ways, and also by surgery to remove the tumor. But all these procedures are only desirable when the tumor is in its initial stages. When the tumor matures, the carcinogenic cells move to different body parts and form tumors there. So, it is indispensable to find out a tumor when it is in its initial stages only. Computer Aided Detection can be used to detect tumors in its early stages.

1.5 ORGANIZATION OF THE REPORT

This introduction section is followed by the Literature Survey. The literature survey explains the current existing systems. It also introduces domain specific terminology which forms the background to understand this project. It discusses in depth about some existing solutions' core aspect which also forms the basis for many other solutions. The section also discusses the drawbacks in all the solutions exhaustively. The literature survey section is followed by Design of the Proposed System section. This section discusses the evolution and design of the proposed solution. Next section discusses the implementation of the design discussed in the previous section. The Data Flow Diagrams and Flowcharts are discussed in this section of the project. The algorithm is also discussed in this section. The data set being used, the features of the data set, and their significance are mentioned. The testing process is also included. The next section deals with the result analysis. The system is executed over the test cases and the results are analyzed and discussed. The final 4 section deals with limitations and recommendations. The references showing the base papers used in this project are then mentioned.

2. LITERATURE SURVEY

2.1 Introduction to the problem domain

The use of computers to help radiologists in the acquisition (e.g. CT, MRI, US, computed radiography), management and storage, and reporting of medical images is well established. More recently, computer programs have been developed and approved for use in clinical practice that aid radiologists in detecting potential abnormalities on diagnostic radiology exams. This application has been termed computer-aided (or assisted) detection, commonly referred to as CAD. As used in this overview, the term 'computer-aided detection' refers to pattern recognition software that identifies suspicious features on the image and brings them to the attention of the radiologist, in order to decrease false-negative readings.

CAD is an interdisciplinary technology combining elements of artificial intelligence and computer vision with radiological and pathology image processing. A typical application is the detection of a tumor. For instance, some hospitals use CAD to support preventive medical check-ups in mammography (diagnosis of breast cancer), the detection of polyps in the colon, and lung cancer.

2.2 Existing Systems

Most of the existing systems make use of various machine learning and deep learning architectures to perform tumor segmentation. Motivated by the performance of recent convolutional neural networks(CNN) various models were built using VGG-16 with skip connections and bootstrapping loss, ResNet-50 with bootstrapping loss, U-net with bootstrapping loss, Residual U-net with various combinations of bootstrapping loss, softmax loss and hierarchical dice loss. It was observed that Residual U-net with bootstrapping loss achieved the highest precision of 0.9761. Other models make use of Batch Normalization and other regularization techniques to improve their accuracy.

2.3 Related Work

In reference [1]

Brain Tumor Segmentation Based on Refined Fully Convolutional Neural Networks with A Hierarchical Dice Loss

In this paper, Jiachi Zhang et al. have used BRATS(Brain Tumor Segmentation) 2013 and 2015 datasets to train over the VGG network with bootstrapping loss, U-net with bootstrapping loss, ResNet50 with bootstrapping loss, Residual U-net with bootstrapping loss, softmax loss and hierarchical dice loss. Among these networks, Residual U-net with bootstrapping loss has acquired the highest precision of 0.9761.

In reference[2]

Predictive modeling of brain tumor: A Deep learning approach

A dataset from Kaggle was taken on which data Augmentation was performed to increase the size of the dataset, then the dataset was preprocessed using crop normalization. The preprocessed data was used to train VGG-16, Inception-V3 and Resnet-50 out of which it was observed that Resnet-50 acquired a higher accuracy of 95.

In reference[3]

Automatic Brain Tumor Detection and Segmentation

The proposed method was tested and evaluated on the BRATS 2015 datasets which contain 220 high-grade gliomas (HGG) and 54 low-grade gliomas (LGG) patient scans. The dataset underwent data augmentation and then fed to U-net which employed skip connections architecture. To minimize the cost function Stochastic Gradient Descent was used and Adam was to estimate the parameters.

In reference[4]

Brain Tumor MRI Segmentation and Classification Using Ensemble Classifier

In this paper, Parasuraman et al. proposed an ensemble method for classifying the brain tumor. The dataset consisting of MRI scanned images of the brain is trained over a Feed-Forward Artificial Neural Network(FFAN), Support Vector Machine(SVM), Extreme Learning Machine(ELM) and an Ensemble model consisting of the above-mentioned models. It was observed that the ensemble model was more accurate than others with an accuracy of 91.17.

2.4 Tools/Technologies used

2.4.1 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor. The key benefits of using Tensorflow are:

1. The single biggest benefit TensorFlow provides for machine learning development is abstraction. Instead of dealing with the nitty-gritty details of implementing algorithms, or figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application. TensorFlow takes care of the details behind the scenes.

2. TensorFlow offers additional conveniences for developers who need to debug and gain introspection into TensorFlow apps. The eager execution mode lets you evaluate and modify each graph operation separately and transparently, instead of constructing the entire graph as a single opaque object and evaluating it all at once.

3. The TensorBoard visualization suite lets you inspect and profile the graphs run by way of an interactive, web-based dashboard.

2.4.2 Keras

Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines. Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was "designed for human beings, not machines," and "follows best practices for reducing cognitive load."

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training.

2.4.3 OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

openCV is used to manipulate images and perform basic preprocessing before passing in images to the model for training or predictions. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers..

It promotes OpenVisionCapsules, which is a portable format, compatible with all other formats.

2.4.3 Google Colab

Google Colab is a free cloud service provided by Google Inc and provides Machine Learning and Deep Learning developers with free CPU,GPU and TPU runtimes. It is directly synced with Google drive which provides a lot of convenience to the developers to store their models and datasets and use them directly without much hassle.

3.DESIGN OF PROPOSED SYSTEM

3.1 Block Diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams. Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagrams and layout diagrams used in electrical engineering, which show the implementation details of electrical components and physical construction.

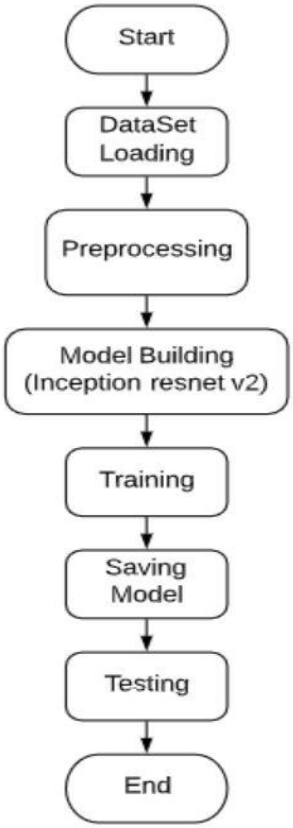


Figure 3.1.1 Block Diagram

The above model shows the flow of steps involved in building the model.The steps include loading the dataset, preprocessing, building the Inception resnet v2 model and then training the model.

3.2 Module Description

3.2.1 Preprocessing:

(i) Data Augmentation:

Data Augmentation is used to increase the number of images by performing various operations like rotation, shearing, zooming, changing the brightness level of the images and flipping the images which can be done by using ImageDataGenerator class in Keras library which generates batches of images, for example

ImageDataGenerator(rotation_range=40,shear_range=0.2,horizonta_flip=True, brighteness_level=(0.5,1)) will generate images by rotating them by 40 degrees clockwise,shearing the image, Flipping it horizontally and by increasing the brightness level of the image.

(ii) Denoising:

This involves denoising of images where the noise in the images is suppressed which can be done by using fastNLmeansDenoising method in which each pixel value is replaced by average of most resembling pixels in the image.A window is considered around the pixel and similar windows are found in the image the resemblance between them is found by finding the euclidean distance between the windows then the pixel value is replaced by the average of the similar pixels in the image.

(iii)Resizing:

The Inception ResNet v2 model takes an input of size 299 by 299 and the pixel values should be floats instead of integers; this can be done using Keras library which has a ‘preprocess_input’ function defined for all models.

3.2.2 Image labeling and loading

This module iterates over the dataset assigning a label to each image in the form of one hot encoding which refers to splitting the column into two columns which contain either “0” or “1” depending on the category of the input for example a categorical variable tumor is represented by the vector [0,1] and non-tumor will be represented by [1,0].

3.2.3 Model Building

A Inceptionresnetv2 model is built which can be done by using Keras library.Since the number of output classes is two i.e. (presence of a tumor or no tumor), the architecture is modified such that the output layer contains only two neurons. The architecture is modified by adding a fully connected layer added to the top layer with relu activation function with a dropout layer and a layer with two neurons above it with softmax activation function which will be representing probabilities o the output and the model is compiled using Stochastic Gradient descent was used to reduce the cost function.

3.2.4 Training and Testing

This model involves splitting the dataset obtained from the previous state into training, testing, updating of weights, calculating loss and performance metrics after training and then testing the trained model using the testing dataset.

3.3 Theoretical Foundation

DataAugmentation

This is a technique which can be used to expand the size of a training dataset by creating modified versions of images in the dataset which can be done by rotating, shearing, zooming, flipping the images and by changing the brightness of the images using this the number of images were increased to 1075 out of which 680 images are with tumor and 395 images are without tumor.

Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Adam Optimizer

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent.

They are:

• **Adaptive Gradient Algorithm (AdaGrad)** that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

• **Root Mean Square Propagation (RMSProp)** that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy). Adam realizes the benefits of both AdaGrad and RMSProp. Instead of adapting the parameter learning rates based on the

average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). 12 Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages. The initial value of the moving averages and beta1 and beta2 values close to 1.0 result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

Stochastic Gradient Descent:

Gradient Descent can be described as an iterative method that is used to find the values of the parameters of a function that minimizes the cost function as much as possible. The parameters are initially defined as a particular value and from that, Gradient Descent is run in an iterative fashion to find the optimal values of the parameters, using calculus, to find the minimum possible value of the given cost function. In SGD, the gradient of the cost function of a single example at each iteration is found instead of the sum of the gradients of the cost function of all the examples. In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with significantly shorter training time.

Dropout:

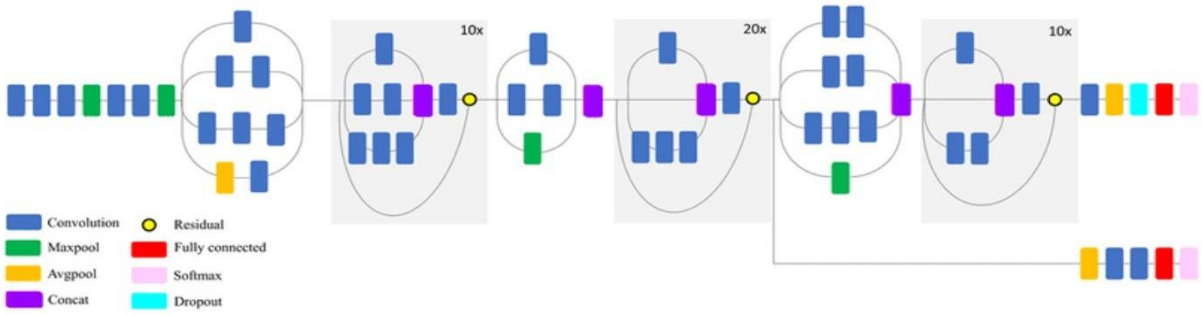
Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer

BackPropagation:

The algorithm is used to effectively train a neural network through a method called chain rule. In simple terms, after each forward pass through a network, backpropagation performs a backward pass while 13 adjusting the model's parameters (weights and biases) and this updation of weights is done using the errors which are calculated at each of the layers in the neural network.

InceptionResnetV2:

The network is 164 layers deep and takes a 299x299 size image as an input. As the name suggests, it is a combination of both inception and resnet models. From the figure 3.3.1, we can see that the initial layers of the network are composed of convolutional layers and maxpooling layers. Then it is followed by an inception block, followed 10 inception-resnet blocks, followed by an inception block, followed by 20 inception-resnet blocks, followed by an inception block then followed by 10 inception-resnet blocks, then followed by a convolutional layer, average pooling layer and a dropout layer for regularization, followed by a fully connected layer followed by a softmax layer to get the probabilities of each class.



3.3.1 InceptionResnetV2 architecture

3.4 Flowchart

A flowchart is a type of diagram that represents an algorithm, workflow or process.Flowchart can also be defined as a diagrammatic representation of an algorithm ie step by step approach to solve a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. The two most ommon types of boxes in a flowchart are: a processing step, usually called activity, and denoted as a rectangular box. a decision, usually denoted as a diamond.

There are four general types of flow charts:

1. Document flowcharts, showing controls over a document-flow through a system.
2. Data flowcharts, showing controls over a data-flow in a system.
3. System flowcharts, showing controls at a physical or resource level.
4. Program flowchart, showing the controls in a program within a system Notice that every type of flowchart focuses on some kind of control, rather than on the particular flow itself.

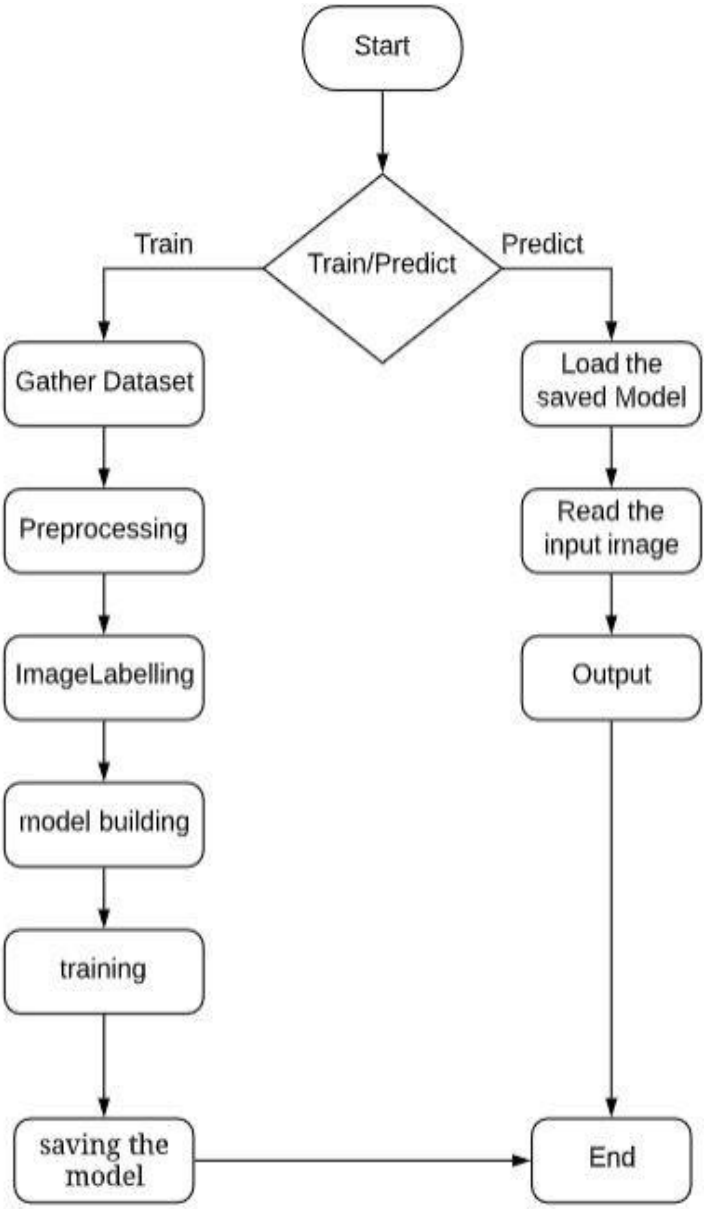


Figure 3.4.1 Activity Diagram

3.5 UML diagrams

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Building blocks of the UML The vocabulary of the UML encompasses three kinds of building blocks.

1. Things.
2. Relationships
3. Diagrams.

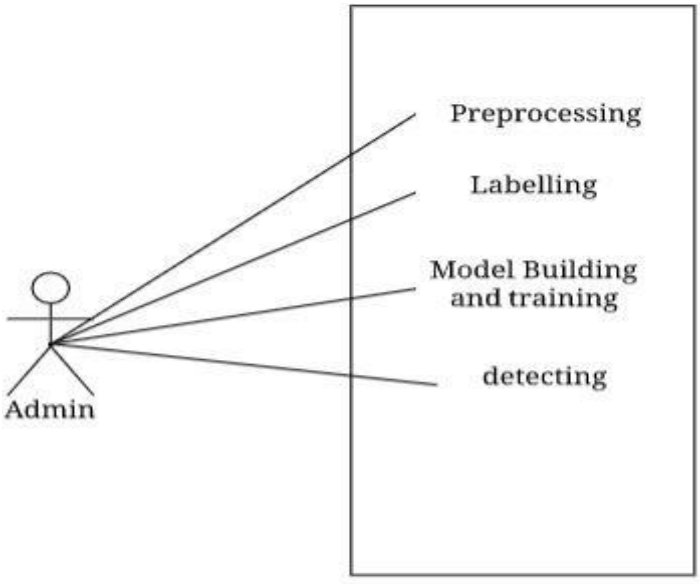


Figure 3.5.1 Use case diagram

A use case diagram is a dynamic or behavior diagram in UML . Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system 16 needs to perform. In this context, a "system" is

something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

3.6 State Diagram

State diagrams require that the system described is composed of a finite number of states sometimes, this is indeed the case, while at other times this is a reasonable abstraction.A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams.

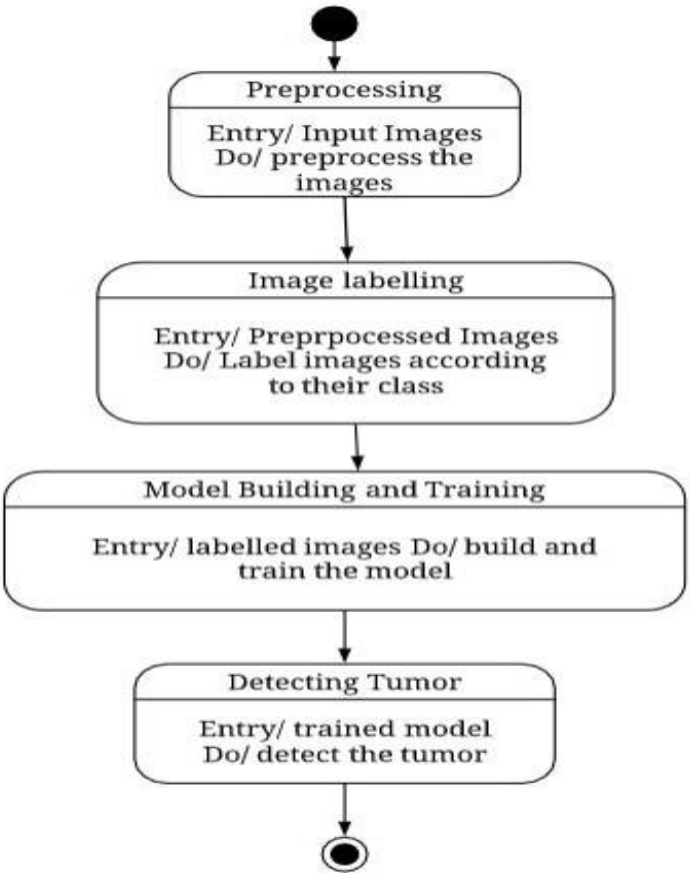


Figure 3.6.1 State Diagram

(i)Preprocessing:

Data Augmentation is performed to increase the number of images in the dataset and denoising is done to suppress the noise in the image

(ii) Image labeling:

This module iterates over the preprocessed dataset assigning a label to each image in the form of one hot encoding which refers to splitting the column into two columns which contain either “0” or “1” depending on the category of the input, for example, a categorical variable tumor is represented by the vector [0,1] and non-tumor will be represented by [1,0].

(iii) Model Building:

A Inceptionresnetv2 model is built by using Keras library. Since the number of output classes is two i.e. (presence of a tumor or no tumor), the architecture is modified such that the output layer contains only two neurons and The dataset is split into training and testing sets using train_test_split method of Scikit-Learn library and model was trained using the training dataset.

(iv) Detecting Tumor

The trained model is loaded and the input image is taken to predict whether it consists of a tumor or not.

4 IMPLEMENTATION OF THE PROPOSED SYSTEM

4.1 Design and Steps/Criteria

The dataset was taken from Kaggel which consisted of 253 images out of which 98 images of the brain were without tumor and 155 images were of brain with tumor. Then we performed DataAugmentation on the dataset which gave 1075 images out of which 680 images were of brain with tumor and 395 images were without tumor.

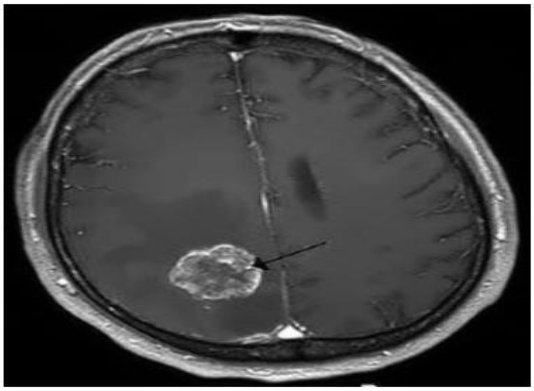


Figure 4.1.1 Brain image with tumor

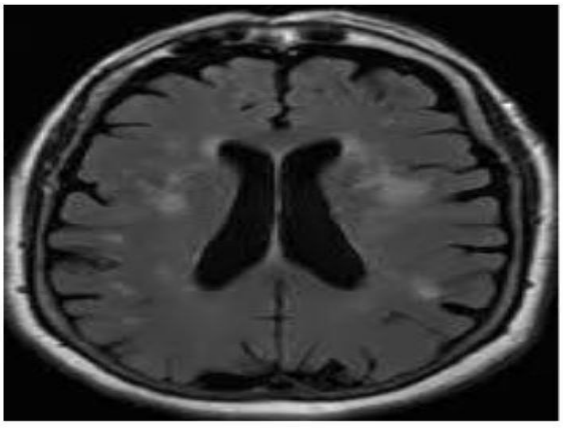


Figure 4.1.2 Brain image without tumor

Then we performed denoising on the dataset which reduced the noise in the images using fastNlMeans method. Then we imported the InceptionResnetV2 model from Keras

library which was already trained on imagenet dataset. The architecture was modified by adding a dense layer to it with relu activation function and a dropout layer was added to overcome overfitting. Since the dataset has only two classes i.e. brain with tumor or without tumor we modified our model such that the output layer consisted of two neurons. This layer was added with a softmax activation function which produces probability of the image having a tumor or not. We used a Stochastic Gradient descent optimizer used to reduce the loss while training the model. Then the images are labelled, the images in the dataset are iterated and labelled ie a brain image with tumor is labelled as [0,1] and image without tumor is labelled as [1,0].Similarly the images in the testing dataset are labelled. Then we trained the model using the images in the training dataset and labels given to them. Then the model was saved, this model was loaded and the output for the images in the testing dataset were predicted and An accuracy of 89.47 was obtained for the model on the testing dataset.

We trained the model on different dataset, First we trained the model on the dataset without performing augmentation, denoising on the dataset. An accuracy of 76.31 was obtained for this model.

We trained another model on the dataset by using Stochastic Gradient Descent without performing dataAugmentation, Denoising on the dataset. We got an accuracy of 78.94 on the testing dataset.

Then we trained the model on the dataset by performing DataAugmentation, Denoising the images and using stochastic gradient descent this increased the accuracy to 89.47.

Then we built a model by adding a dense layer with relu activation function followed by a dropout layer.We trained the model on the dataset on which DataAugmentation denoising was performed and This model gave an Accuracy of 89.47 on the testing dataset.

4.2 Algorithms/Pseudo code

4.2.1 Libraries

Used During the implementation we made use of some python packages which possess many in-built methods that provide many useful functionalities. They are:

Numpy It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

google.colab : This is the python library which is having many utilities that are related to Google colab.

PIL : PIL which abbreviates to Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux and is developed by Fredrik Lundh.

Os : The OS module in Python provides functions for creating and removing a directory, fetching its contents, changing and identifying the current directory, etc. ***os.listdir()*** method in python is used to get the list of all files and directories in the specified directory.

Tensorflow: TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

```
from google.colab import drive

from tensorflow.keras.applications.inception_resnet_v2 import InceptionResnetv2

from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input

from tensorflow.keras.layers import GlobalAveragepoling2D,Dense,dropout

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import SGD,Adam

import os

from PIL import Image

import numpy as np
```

4.2.2 Data Augmentation and Denoising

Data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Training deep learning neural network models on more data can result in more skillful models. The number of images can be increased by performing various operations like rotation, shearing, zooming, changing the brightness level and flipping which can be done using ImageDataGenerator class in kerass.Then denoising was performed on the images in the dataset which is a technique used to suppress the noise in the images.

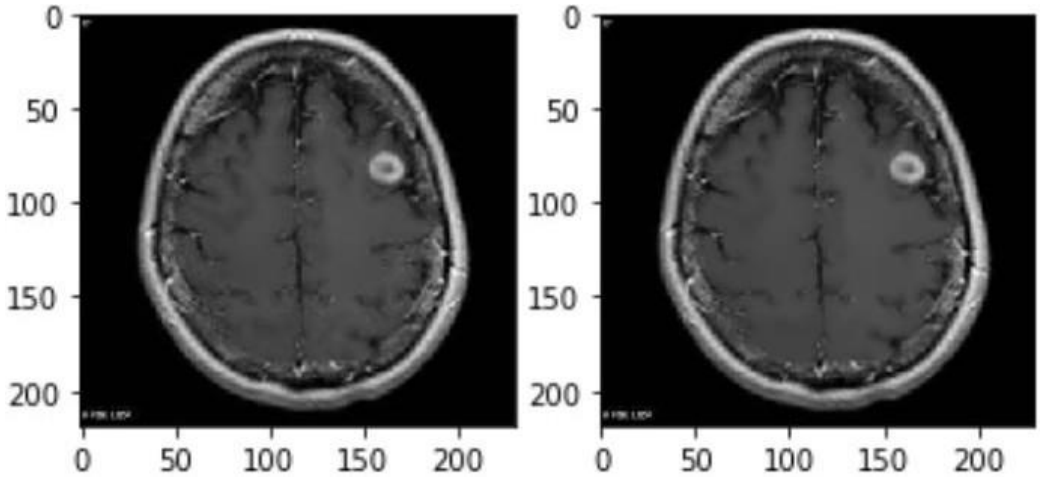
```
datagen = ImageDataGenerator(rotation_range = 40, shear_range = 0.2, horizontal_flip =
                             True, brightness_range = (0.5, 1.5))
for f in os.listdir(path):
img = load_img(path+"\\"+f)
x = img_to_array(img)
```

```
  x = x.reshape((1, ) + x.shape)

  i = 1
  for batch in datagen.flow(x, batch_size = 1,
                save_to_dir =path where the images to be saved,
                save_prefix ='image', save_format ='jpeg'):
   i += 1
   if i > 5:
      Break

#Denoising
 p1 = path to the images
 p2 = path where the denoised images should be saved
 for f in os.listdir(p1):
    img=cv2.imread(p1+"\\"+f)
  dst=cv2.fastNlMeansDenoising(img,h=3)
  img=Image.fromarray(dst)
  gray=img.convert('L')

  gray.save(p2+"\\yes\\"+f)
```



Original Image Denoised Image

Figure 4.2.2.1 Comparing original and Denoised Images

Figure 4.2.2.1 shows the original image and the image obtained after denoising the original image. The denoised image highlights the edges in the image and also the tumor present in the image is highlighted.

4.2.3 Model Building

An VGG-16 model, Resnet-52 model and Inception Resnetv2 model were built for this project. We imported the models from the keras.application package.

4.2.3.1 VGG-16 Architecture

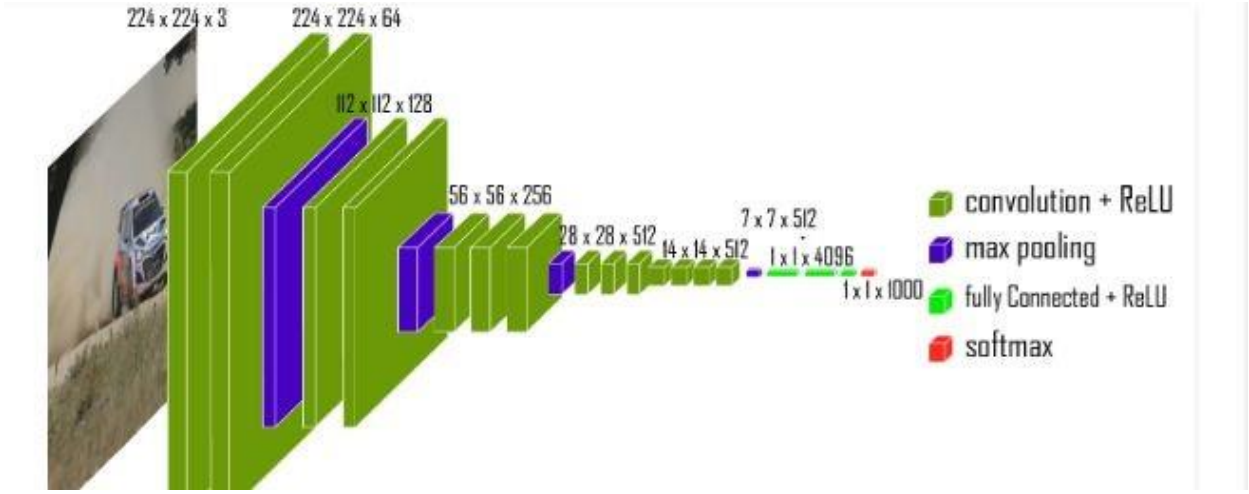


Figure 4.2.3.1.1 VGG-16 Architecture

From the figure 4.2.3.1.1,we can see that the vgg-16 model takes 224x224x3 size image as an input. Initial two layers are convolutional layers, followed by a maxpooling layer, followed by 2 convolutional layers, followed by a maxpooling layer, followed by 3 convolutional layers followed by a maxpooling layer, followed by 6 convolutional layers, followed by a maxpooling layer, then 3 fully connected layers and then followed by a softmax layer to produce probabilities of each class.

```
from tensorflow.keras.applications.vgg16 import VGG16
vgg=VGG16(weights="imagenet",include_top=False)
x=vgg.output
x=GlobalAveragePooling2D()(x)
x=Dropout(0.20)(x)
x=Dense(1024,activation="relu")(x)
preds=Dense(2,activation="softmax")(x)
model=Model(inputs=vgg.input,outputs=preds)
model.summary()
```

4.2.3.2 Resnet-50:

Resnet When the number of layers in CNN kept increasing, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly. Such degradation is not caused by overfitting or by adding more layers to a deep network leads to higher training error. The deterioration of training accuracy shows that not all systems are easy to optimize.

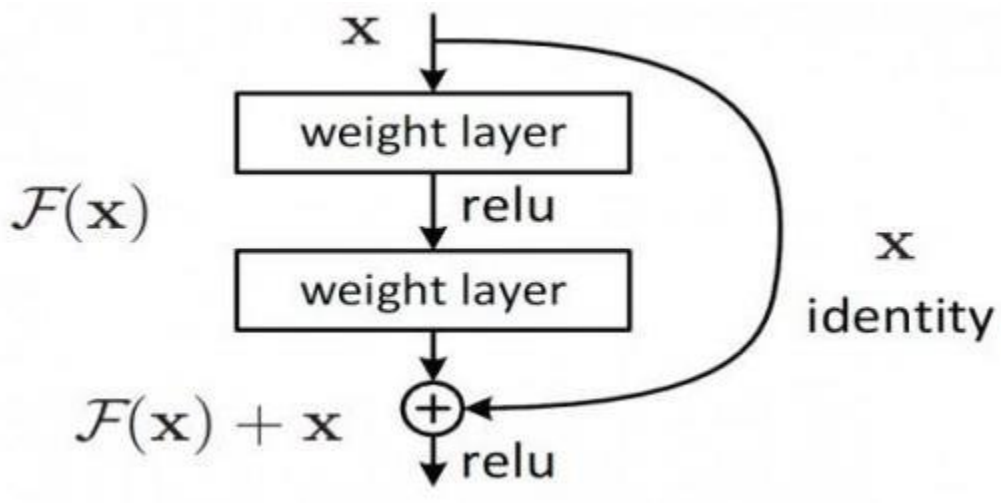


Figure 4.2.3.2.1 skip connections architecture

Skip connection in Resnet To overcome this problem, Microsoft introduced a deep residual learning framework. Instead of hoping every few stacked layers directly fit a

desired underlying mapping, they explicitly let these layers fit a residual mapping. The formulation of F(x)+x can be realized by feedforward neural networks with shortcut connections. Shortcut connections are those skipping one or more layers as shown above. The shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. Resnet hence helps in overcoming problems like,

- ResNets are easy to optimize, but the “plain” networks (that simply stack layers) shows higher training error when the depth increases.
- ResNets can easily gain accuracy from greatly increased depth, producing results which are better than previous networks.

```
from tensorflow.keras.applications.resnet import Resnet50

resnet=Resnet50(weights="imagenet",include_top=False)
x=resnet.output
x=GlobalAveragePooling2D()(x)
x=Dropout(0.20)(x)
x=Dense(1024,activation="relu")(x)
preds=Dense(2,activation="softmax")(x)
model=Model(inputs=resnet.input,outputs=preds)
model.summary()
```

Modified Inception-Resnet-v2 model:

In the earlier section, we have discussed the architecture of the Inception-resnet-v2 model. The architecture of the inception-resnet-v2 model was modified by adding a fully connected layer on top of the model followed by another layer to suit the dataset i.e., a layer with two neurons was added where each represents a class in the dataset. A dropout layer was added to the model which is a regularization technique in which the input from the previous layer is dropped.

```
ir=InceptionResNetV2()
x=ir.output
x=GlobalAveragePooling2D()(x)
x=Dense(512,activation="relu")(x)
preds=Dense(2,activation="softmax")(x)
model=Model(inputs=ir.input, outputs=preds)
```

4.2.4 Model Training

Training the model implies that we input our image dataset to the model and check the performance of the model by using loss function which compares output of the model with the actual output and calculates how much the model's predicted value differs from the actual output.

The images in the dataset are labelled by iterating over the preprocessed dataset assigning a label to each image in the form of one hot encoding which refers to splitting the column into two columns which contain either “0” or “1” depending on the category of the input, for example, a categorical variable tumor is represented by the vector [0,1] and non-tumor will be represented by [1,0].The model is compiled using Stochastic Gradient Descent.Then we train our model with the training dataset.

```
 p1="drive/My Drive/final_dataset/yes"
p2="drive/My Drive/final_dataset/no
for f in os.listdir(p1):
im=cv2.imread(p1+"/"+f)
label=[0,1]
im=cv2.resize(im,dsize=(299,299))
im=preprocess_input(im)
x.append(im)
y.append(label)
for f in os.listdir(p2):
im=cv2.imread(p2+"/"+f
label=[1,0]
```

```
 im=cv2.resize(im,dsize=(299,299))
 im=preprocess_input(im)
 x.append(im)
 y.append(label)
model.fit(x,y,epochs=200)
```

4.3 Testing Process

In this step our model is given the testing dataset which has not been used for training the model. The model predicts the output for the images in the test dataset and these predicted values are compared with the expected output from which we can determine how much the predicted values are deviating from the actual values and the accuracy of the model can be calculated based on the values predicted by our model.

The trained model is tested on the test dataset and we make use of crossentropyloss to determine the loss of the model on the test data and thereby the accuracy of the model on the test data is found.

```
p2="drive/My Drive/testing"
res=[]
for f in x1:
im=cv2.imread(p2+"/"+f)
im=cv2.resize(im,dsize=(299,299))
im=preprocess_input(im)
im=im.reshape(1,299,299,3)
ta=model.predict(im)
res.append(ta)
for i in range(len(ans)):
ans[i][0]=round(ans[i][0])
ans[i][1]=round(ans[i][1])
From sklearn.metrics import accuracy_score
accuracy_score(ans,y1)
```

5. RESULTS/OUTPUT AND DISCUSSIONS

Training the model

The dataset is divided into two sub datasets. One of them is used to train the model which is run through a certain number of epochs and the other test dataset is used to test the model. The loss is calculated using categorical crossentropy.

```
Epoch 1/200
32/32 [==============================] - 37s 1s/step - loss: 0.7115 - accuracy: 0.4955
Epoch 2/200
32/32 [==============================] - 33s 1s/step - loss: 0.6473 - accuracy: 0.6454
Epoch 3/200
32/32 [==============================] - 33s 1s/step - loss: 0.6130 - accuracy: 0.6783
Epoch 4/200
32/32 [==============================] - 33s 1s/step - loss: 0.5806 - accuracy: 0.7153
Epoch 5/200
32/32 [==============================] - 33s 1s/step - loss: 0.5517 - accuracy: 0.7373
Epoch 6/200
32/32 [==============================] - 33s 1s/step - loss: 0.5402 - accuracy: 0.7672
Epoch 7/200
32/32 [==============================] - 33s 1s/step - loss: 0.4976 - accuracy: 0.7982
Epoch 8/200
32/32 [==============================] - 33s 1s/step - loss: 0.4731 - accuracy: 0.8312
Epoch 9/200
32/32 [==============================] - 33s 1s/step - loss: 0.4395 - accuracy: 0.8472
Epoch 10/200
32/32 [==============================] - 33s 1s/step - loss: 0.4329 - accuracy: 0.8591
Epoch 11/200
32/32 [==============================] - 33s 1s/step - loss: 0.4070 - accuracy: 0.8751
Epoch 12/200
32/32 [==============================] - 33s 1s/step - loss: 0.3731 - accuracy: 0.8951
Epoch 13/200
32/32 [==============================] - 33s 1s/step - loss: 0.3567 - accuracy: 0.8951
```

Figure 5.1.1 Training Results

Finding Accuracy of the model

After training the model using the dataset we test our model using using the testing dataset which was not used for training and calculate the accuracy based upon the output produced by the model

.

```
[ ]  from sklearn.metrics import accuracy_score,mean_squared_error
     print("Accuracy :", accuracy_score(ans,y1)*100)

     Accuracy : 89.47368421052632
```

Figure 5.1.2 Accuracy Score

Predicting class of the input image

The path of the image is passed to the model which predicts whether the tumor is present in the given image or not.

```
]  for i in ans:
       if i[0]>i[1]:
           print("No tumor")
       else:
           print("Tumor")

   No tumor
```

5.1.3 Output Class

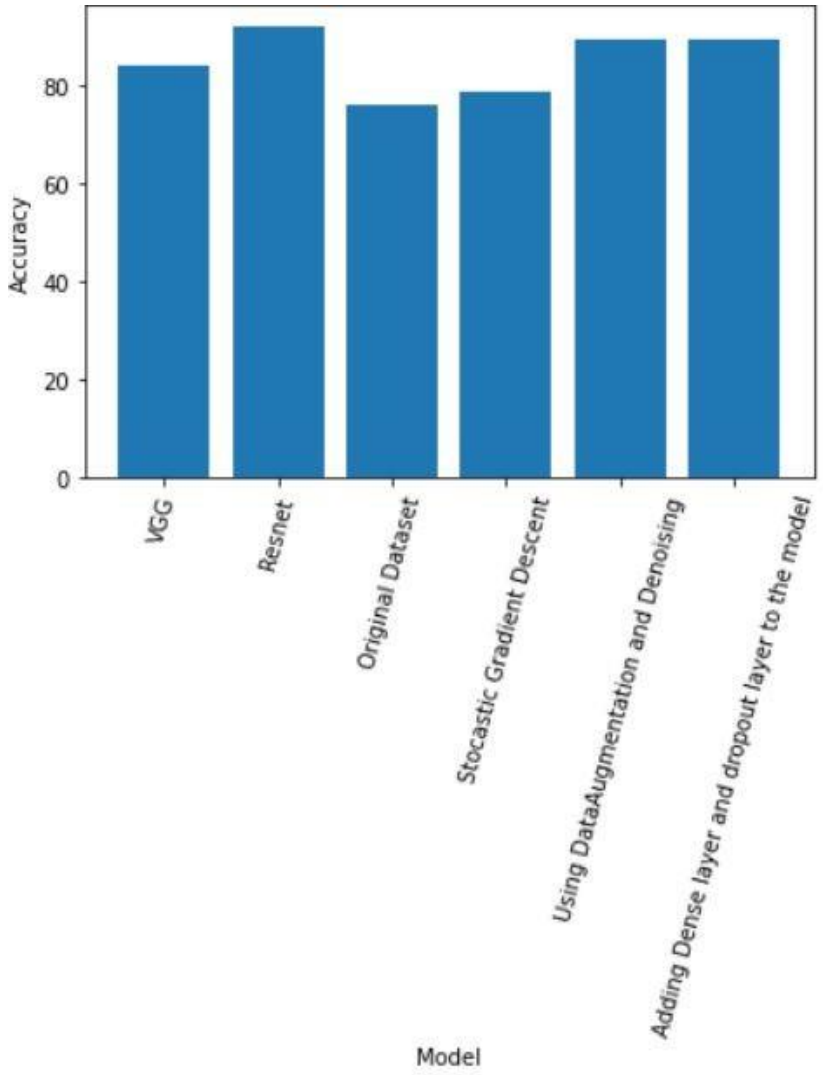
Performance of various models on dataset

The dataset has been tested on different permutations and each of the models has tested and given different performance measures on the same dataset. The following table provides an overview about the various parameters and optimizers that have been tested and the various accuracies that have been recorded for these models.

Model	Accuracy
VGG model trained with augmented dataset and using stochastic gradient descent	82.41
Resnet model trained with augmented dataset and using stochastic gradient descent	92.10
InceptionResnetv2model without data Augmentation,no denoising,no optimizer,no dropout	76.31
inception Resnetv2 model with SGD optimizer,nodataAugmentation,no denoising,no dropout	78.94
Inception Resnetv2 model with Adam optimizer,nodataAugmentation,no denoising,no dropout	78.94
Inception Resnetv2 model with SGD optimizer, with dataAugmentation and denoising,no dropout	89.47
Inception Resnetv2 model with SGD optimizer,dataAugmentation, denoising and dropout	89.47

5.1.4 Performance of various models on the dataset

The Inception Resnet model with Stochastic gradient descent and a fully connected layer with a dropout layer gave an accuracy of 89.47% on the testing dataset.



5.1.5 Bar Graph showing accuracy change with different models

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

A brain tumor is always undesirable and fatal no matter what stage it is. But, detecting a tumor in its early stages could help a person gain his life. So, in order to detect a tumor in its early stages, we tried to build a system using Convolutional Neural Networks(CNN). The system that we have built makes use of the Inception_Resnet_v2 model to detect the presence of tumors in the MRI scanned image of the brain. When a dataset consisting of MRI scanned images of the brain is trained over the Inception_Resnet_v2 model, the accuracy obtained was 89.47% , the VGG model gave an accuracy of 84.21 and an accuracy of 94% was obtained by using resnet model The accuracy and precision of the model can be further increased by increasing the size of the dataset.

6.2 Future Work

As an extension of our work, we would like to,

- Make an UI for our program, which could ease the work of prediction for non-technical people.
- To use Image segmentation to segment the tumor from the detected image and also try to predict the severity or the stage of the tumor.
- Increase the accuracy of the model by using Ensemble methods, i.e., combination of two or more models, as they are proven to be more efficient than a single model.
- Use grid search to identify the appropriate values of Hyper Parameters.

8. References

[1] Brain Tumor Segmentation Based on Refined Fully Convolutional Neural Networks with A Hierarchical Dice Loss Jiachi Zhang, Xiaolei Shen, Tianqi Zhuo, Hong Zhou Key Laboratory for Biomedical Engineering of Ministry of Education, Zhejiang University of China December 25, 2017

[2] Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, Yike Guo 1 Data Science Institute, Imperial College London, SW7 2AZ, London, UK 2 Neurosciences Research Centre, Molecular and Clinical Sciences Institute, St. George's, University of London, London SW17 0RE, UK 3 National Heart and Lung Institute, Imperial College London, SW7 2AZ, London, UK

[3] Predictive modeling of brain tumor: A Deep learning approach
PRIYANSH SAXENA, ABV-Indian Institute of Information Technology and Management, Gwalior, IN AKSHAT MAHESHWARI, ABV-Indian Institute of Information Technology and Management, Gwalior, IN SHIVANI TAYAL, Indian Statistical Institute, Delhi Centre, IN SAUMIL MAHESHWARI, ABV-Indian Institute of Information Technology and Management, Gwalior, IN.
Brain Tumor MRI Segmentation and Classification Using Ensemble Classifier by Parasuraman Kumar, B. Vijay Kumar.

[4] Brain Tumor MRI Segmentation and Classification Using Ensemble Classifier by Parasuraman Kumar, B. Vijay Kumar.

APPENDIX

DataAugmentation

```
from keras.preprocessing.image import ImageDataGenerator,array_to_img, img_to_array,
                                      load_img
import os


path="C:\\Users\\Desktop\\braintumor\\dataset\\yes”
datagen = ImageDataGenerator(rotation_range = 40, shear_range = 0.2, horizontal_flip =
True, brightness_range = (0.5, 1.5))
for f in os.listdir(path):
    img = load_img(path+"\\"+f)
    x = img_to_array(img)
    x = x.reshape((1, ) + x.shape)

    i = 1
    for batch in datagen.flow(x, batch_size = 1,
                        save_to_dir =path where the images to be saved,
                        save_prefix ='image', save_format ='jpeg'):
        i += 1
        if i > 5:
            break

path="C:\\Users\\Desktop\\braintumor\\dataset\\no”
datagen = ImageDataGenerator(rotation_range = 40, shear_range = 0.2, horizontal_flip =
True, brightness_range = (0.5, 1.5))
for f in os.listdir(path):
    img = load_img(path+"\\"+f)
```

```
    x = img_to_array(img)
    x = x.reshape((1, ) + x.shape)

    i = 1
    for batch in datagen.flow(x, batch_size = 1,
                              save_to_dir =path where the images to be saved,
                              save_prefix ='image', save_format ='jpeg'):
        i += 1
        if i > 5:
            break
```

Denoising

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from PIL import Image
import PIL

datagen = ImageDataGenerator(
        rotation_range = 40,
        shear_range = 0.2,
        horizontal_flip = True,
        brightness_range = (0.5, 1.5))
path="C:\\Users\\Desktop\\braintumor\\dataset\\yes"
for f in os.listdir(path):
    img = load_img(path+"\\"+f)
    x = img_to_array(img)
```

```
    x = x.reshape((1, ) + x.shape)

    i = 1
    for batch in datagen.flow(x, batch_size = 1,
                                                                    save_to_dir
='C:\\Users\\nayaz\\Desktop\\braintumor\\augmented_dataset\\yes',
                        save_prefix ='image', save_format ='jpeg'):
        i += 1
        if i > 5:
            break



    img=cv2.imread(p3+"\\"+f)
    dst=cv2.fastNlMeansDenoising(img,h=3)
    img=Image.fromarray(dst)
    gray=img.convert('L')
    gray.save(p2+"\\no\\"+f)
```

Model Training code

```
from google.colab import drive

drive.mount('/content/drive')

from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2

import os
```

```
from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input

from tensorflow.keras.layers import GlobalAveragePooling2D,Dense,Dropout

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import SGD,Adam



ir=VGG(weights="imagenet",include_top=False)



x=ir.output

x=GlobalAveragePooling2D()(x)



x=Dropout(0.20)(x)

x=Dense(1024,activation="relu")(x)

preds=Dense(2,activation="softmax")(x)

model=Model(inputs=ir.input,outputs=preds)

model.summary()

model.compile(optimizer=SGD(lr=0.0001,momentum=0.9),loss="categorical_crossentropy",metrics=
['accuracy'])

from PIL import Image

import cv2

import numpy as np
```

```
p1="drive/My Drive/final_dataset/yes"

p2="drive/My Drive/final_dataset/no"

x=[]

y=[]

for f in os.listdir(p1):

    im=cv2.imread(p1+"/"+f)

    label=[0,1]

    im=cv2.resize(im,dsize=(299,299))



    im=preprocess_input(im)



    x.append(im)

    y.append(label)

for f in os.listdir(p2):

    im=cv2.imread(p2+"/"+f)



    label=[1,0]

    im=cv2.resize(im,dsize=(299,299))
```

```
    im=preprocess_input(im)
    x.append(im)
    y.append(label)


x=np.array(x)
y=np.array(y)
model.fit(x,y,epochs=200)
```

Building Vgg16 model

```
from google.colab import drive
drive.mount('/content/drive')
from tensorflow.keras.applications.vgg import VGG16
import os
from tensorflow.keras.applications.vgg import preprocess_input
from tensorflow.keras.layers import GlobalAveragePooling2D,Dense,Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD,Adam


ir=VGG16(weights="imagenet",include_top=False)
```

```
x=ir.output

x=GlobalAveragePooling2D()(x)



x=Dropout(0.20)(x)

x=Dense(1024,activation="relu")(x)

preds=Dense(2,activation="softmax")(x)

model=Model(inputs=ir.input,outputs=preds)

model.summary()

model.compile(optimizer=SGD(lr=0.0001,momentum=0.9),loss="categorical_crossentropy",metrics=
['accuracy'])

from PIL import Image

import cv2

import numpy as np

p1="drive/My Drive/final_dataset/yes"

p2="drive/My Drive/final_dataset/no"

x=[]

y=[]

for f in os.listdir(p1):

    im=cv2.imread(p1+"/"+f)

    label=[0,1]
```

```
    im=cv2.resize(im,dsize=(299,299))


    im=preprocess_input(im)


    x.append(im)
    y.append(label)
for f in os.listdir(p2):
    im=cv2.imread(p2+"/"+f)


    label=[1,0]
    im=cv2.resize(im,dsize=(299,299))


    im=preprocess_input(im)
    x.append(im)
    y.append(label)


x=np.array(x)
y=np.array(y)
model.fit(x,y,epochs=200)
```

Model Testing

```
import os

import cv2

p2="drive/My Drive/testing"

x1=[]

#y=[]

res=[]

for f in os.listdir(p2):

    im=cv2.imread(p2+"/"+f)

    im=cv2.resize(im,dsize=(299,299))

    im=preprocess_input(im)

    im=im.reshape(1,299,299,3)

   # ta=model.predict(im)

   #res.append(ta)

   x1.append(f)



x1=(sorted(x1))
```

```
print(x1)

y1=[]

for i in range(len(x1)):

  if 'N' in x1[i]:

    y1.append([1,0])

  else:

  y1.append([0,1])

 # print(x1[i],y1[i])

p2="drive/My Drive/testing"



#y=[]

res=[]

for f in x1:

    im=cv2.imread(p2+"/"+f)

    im=cv2.resize(im,dsize=(299,299))

    im=preprocess_input(im)

    im=im.reshape(1,299,299,3)

    ta=model.predict(im)

    res.append(ta)
```

```
    #x1.append(f)



ans=[]

for i in range(len(res)):

ans.append(list(res[i][0]))

ans

from sklearn.metrics import accuracy_score,mean_squared_error

print("Accuracy :", accuracy_score(ans,y1)*100)
```