Sona College of Technology

Brain Tumor Detection From MRI images with IBM Watson studio

Done by ;

Team members :

Nallajonnala Ramanaidu

Shaik Suleman

Arla Venkat Royal

INDEX PAGE

1) INTRODUCTION

1.1) Overview

Brain tumor identification is a really challenging task in the early stages of life. These days the issue of brain tumor automatic identification is of great interest. A tumor is the unusual growth of the tissues. A brain tumor is a number of unnecessary cells growing in the brain or central spinal canal. It is the unrestrained progress of cancer cells in any portion of the body. Deep learning techniques can be used in order to detect the brain tumor of a patient using the MRI images of a patient's brain. In this application, we are helping the doctors and patients to classify the type of scan for the specific image given with the help of Neural Networks and store the patient's data.

1.2) Purpose

The purpose is to develop an automated system for enhancement, segmentation and classification of brain tumors. The system can be used by neurosurgeons and healthcare specialists to detect brain tumor. The system incorporates image processing, pattern analysis, and computer vision techniques and is expected to improve the sensitivity, specificity, and efficiency of brain tumor screening. The primary goal of this projects is to extract meaningful and accurate information from these images with the least error possible. The proper combination and parameterization of the phases enables the development of adjunct tools that can help on the early diagnosis or the monitoring of the tumor identification and locations**.**

2) LITERATURE SURVEY

2.1) Existing Problem

There are so many approaches or methods to solve this problem ,here we can discuss some methods.

- **Detection of Tumor in MRI Images Using Artificial Neural Networks**

Automatic defects detection in MR images is very important in many diagnostic and therapeutic applications. This work has introduced one automatic brain tumor detection method to increase the accuracy and yield and decrease the diagnosis time. The goal is classifying the tissues into two classes of normal and abnormal. MR images that have been used here are MR images from normal and abnormal brain tissues. This method uses from neural network to do this classification. The purpose of this project is to classify the brain tissues to normal and abnormal classes automatically, which saves the radiologist time, increases accuracy and yield of diagnosis.

- **Brain Tumor Detection Techniques Using Magnetic Resonance Images**

The brain tumor is an abnormal growth of cells inside the skull which causes damage to the other cells necessary for functioning human brain. Brain tumor detection is a challenging task due to the complex structure of the human brain. MRI images generated from MRI scanners using strong magnetic fields and radio waves to form images of the body which helps for medical diagnosis. This paper gives an overview of the various techniques used to detect the tumor in the human brain using MRI images.

- **A Neural Network-based Method for Brain Abnormality Detection in MR Images Using Gabor Wavelets**

Nowadays, automatic defects detection in MR images is very important in many diagnostic and therapeutic applications. This paper introduces a Novel automatic brain tumor detection method that uses T1, T2_weighted and PD, MR images to determine any abnormality in brain tissues. Here, it has been tried to give a clear description from brain tissues using Gabor wavelets, energy, entropy, contrast and some other statistic features such as mean, median, variance, correlation, values of maximum and minimum intensity. It is used from a feature selection method to reduce the feature space too. this method uses from neural network to do this classification. The purpose of this project is to classify the brain tissues to normal and abnormal classes automatically, which saves the radiologist time, increases accuracy and yield of diagnosis

2.2) Proposed Solution

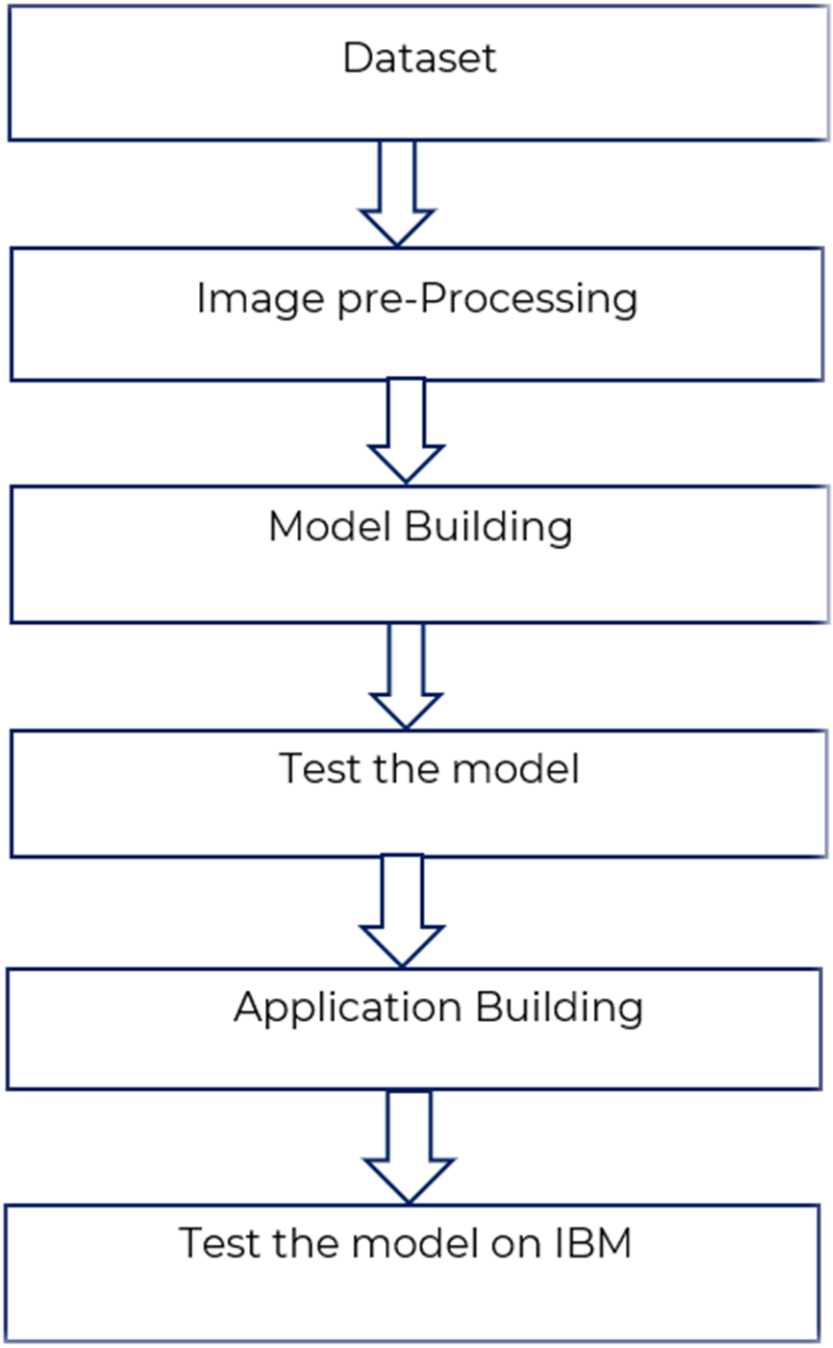
The proposed system has mainly six modules. They are

- Dataset Collection
- Image pre-processing
- Model Building
- Test the model
- Application Building
- Test the model on IBM

In dataset we can take multiple MRI images and take one as input image. In pre-processing image to encoded the label and resize the image. In split the data we set the image as 80% Training Data and 20% Testing Data. Then build CNN model train deep neural network for epochs. Then we created a '.h5' for building web application using flask , using that MRL images we predict the tumor in flask web application . After that using the data we do IBM deployment in IBM Watson studio there we get another file ,using that file we build we application using flask and we can predict the brain tumor using dataset ,here we are using deep neural networks in predicting if the tumor is present or not. And then we build web applications using the Flask framework to show that a patient is effected by brain tumor

3) THEORETICAL ANALYSIS

3.1) Block Diagram



3.2 .1) Software designing

Anaconda Navigator :

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution includes data-science packages suitable for Windows, Linux, and macOS. Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

Jupyter Notebook :

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text mathematics, plots and rich media, usually ending with the ". ipynb" extension.

Tensor flow :

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras :

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

Flask :

Flask is a microframework written in python. It is classified as a microframework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-

relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

3.2 .2) Hardware designing

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 8 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Window

4) EXPERMENTAL INVESTIGATIONS

The analysis of the project that would be developed by our hands. It consists of six steps where the execution starts from taking an input image from the data set followed by the image pre-processing, model building,testing the model and then Save the model and its dependencies,Building a Web application using flask that integrates with the model built using IBM Watson studio. Finally, the output is observed after all the above mentioned steps are completed**.**

4.1) Dataset collection :

The data set used is has been downloaded from Kaggle which consists of nearly 250 images that are used to test and train the system.

4.2) Image pre-processing :

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.

The Brain MRI image dataset has been downloaded from the Kaggle. The MRI dataset consists of around 250 MRI images, including normal, benign, and malignant. These MRI images are taken as input to the primary step. The pre-processing is an essential and initial step in improving the quality of the brain MRI Image. The critical steps in pre-processing are the reduction of impulsive noises and image resizing. In the initial phase,ImageDataGenerator class is used to load the images with different modifications like considering the zoomed image, flipping the image and rescaling the image to a range of 0 and 1

- Configure ImageDataGenerator Class.

In image processing, image acquisition is done by retrieving an image from dataset for processing. It is the first step in the workflow sequence because, without an image no processing is possible. The image that is acquired is completely unprocessed. Here we process the image using the file path from the local device

- Applying ImageDataGenerator functionality to the trainset and test set.

Specify the path of both the folders in the flow_from_directory method.

4.3) Model Building :

The neural network model is to be built by adding different network layers like convolution, pooling, flattening, dropout and neural layers.

For this step we need to import Keras and other packages that we're going to use in building the CNN. Import the following packages: Sequential is used to initialize the neural network. Adding Convolution layer is used to make the convolutional network that deals with the images. Adding Pooling layer is used to add the pooling layers. Flatten layer is the function that converts the pooled feature map to a single column that is passed to the fully connected layer. Adding fully connected layer which includes hidden layer to the neural network.

SEQUENTIAL:

To initialize the neural network, we create an object of the Sequential class.

Ex; model=Sequential()

CONVOLUTION:

To add the convolution layer, we call the add function with the classifier object and pass in Convolution2D with parameters. The first argument feature_detectors which is the number of feature detectors that we want to create. The second and third parameters are dimensions of the feature detector matrix. We used 256 feature detectors for CNNs. The next parameter is input shape which is the shape of the input image. The images will be converted into this shape during pre-processing. If the image is black and white it will be converted into a 2D array and if the image is coloured it will be converted into a 3D array. In this case, we'll assume that we are working with coloured images. Input_shape is passed in a tuple with the number of channels, which is 3 for a coloured image, and the dimensions of the 2D array in each channel. If you are not using a GPU it's advisable to use lower dimensions to reduce the computation time. The final parameter is the activation function. Classifying images is a nonlinear problem. So, we use the rectifier function to ensure that we don't have negative pixel values during computation.

Ex; model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))

POOLING:

The Pooling layer is responsible for reducing the spatial size of the convolved feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Generally, we use max pooling. In this step we reduce the size of the feature map. Generally, we create a pool size of 2x2 for max pooling. This enables us to reduce the size of the feature map while not losing important image information.

Ex; model.add(MaxPooling2D(pool_size=(2,2)))

FLATTENING:

In this step, all the pooled feature maps are taken and put into a single vector for inputting it to the next layer. The Flatten function flattens all the feature maps into a single long column.
Ex; model.add(Flatten())

FULLY CONNECTION:

The next step is to use the vector we obtained above as the input for the neural network by using the Dense function in Keras. The first parameter is output which is the number of nodes in the hidden layer. You can determine the most appropriate number through experimentation. The higher the number of dimensions the more computing resources you will need to fit the model. A common practice is to pick the number of nodes in powers of two. The next layer we have to add is the output layer. In this case, we'll use the sigmoid activation function since we expect a binary outcome. If we expected more than two outcomes, we would use the SoftMax function.

4.4) Test the model :

The model is to be tested with different images to know if it is predicting correctly. . In split the data we set the image as 80% Training Data and 20% Testing Data. Then build CNN model train deep neural network for epochs

4.5) Application Building

After the model is built, we will be integrating it into a web application so that normal users can also use it. The users need to give the scan to know if the tumor is present or not.

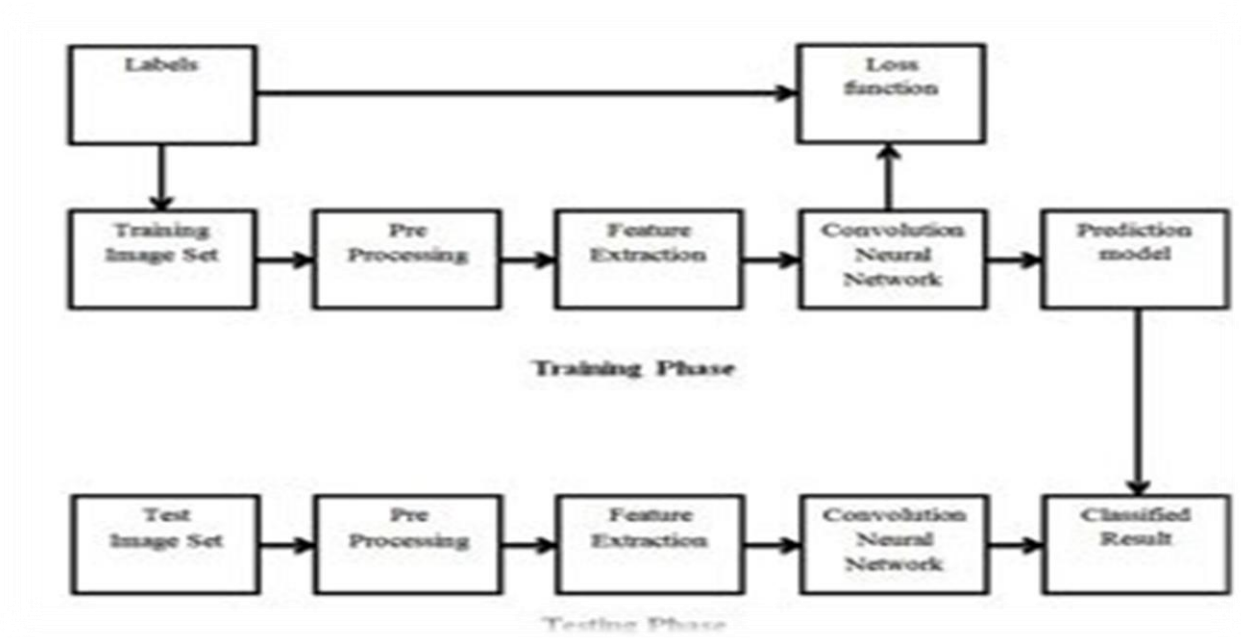
4.6) Test the model in IBM

At finally test the model in IBM using IBM Watson studio and then build a '.h5' using this file we can predict weather a patient is effected brain tumor or not in web application using flask

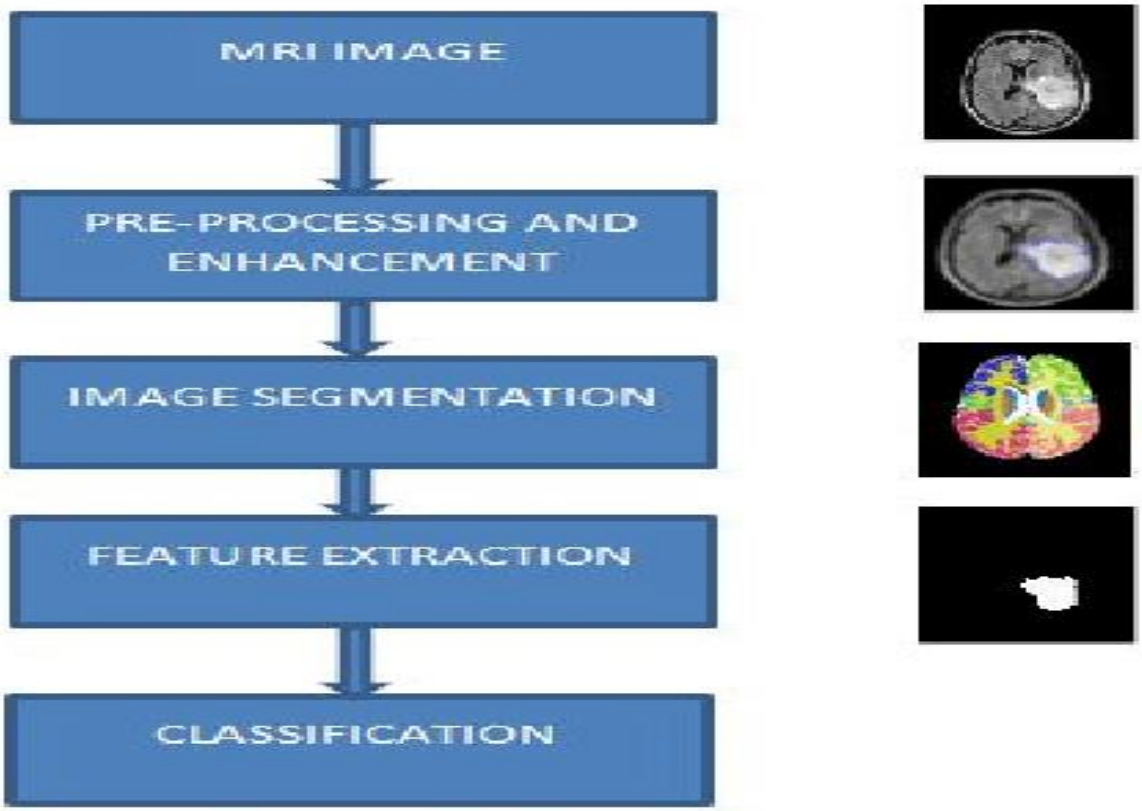
Finally , we detect whether the given MRI brain image has tumor or not using IBM Watson studio.

5) FLOWCHAT :

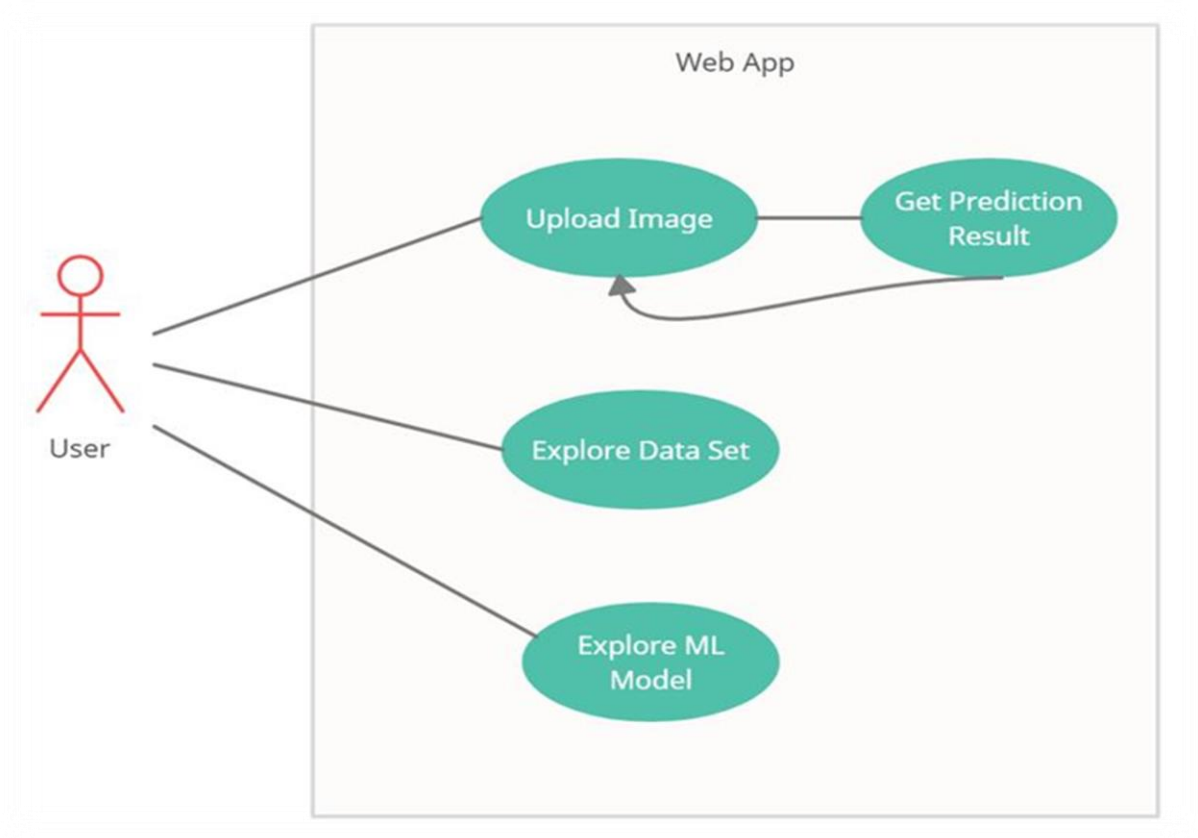
5.1) Block Diagram;

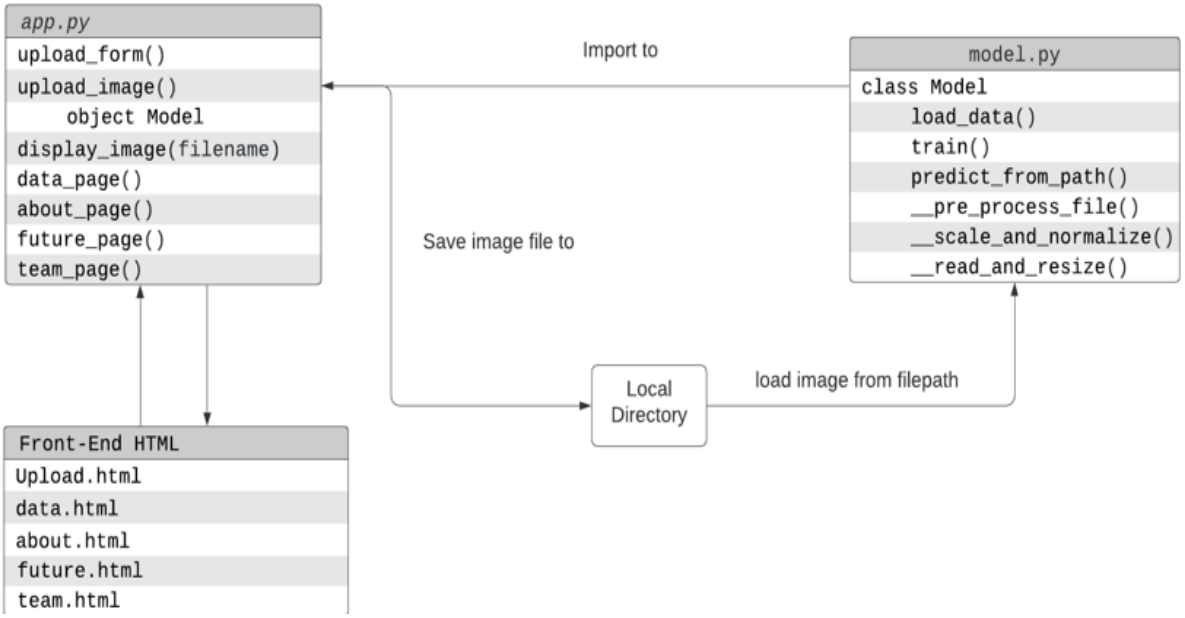


5.2) Flow Diagram:



5.3) Use case Diagram :


Web App
Upload Image
Get Prediction Result
User
Explore Data Set
Explore ML Model


app.py
upload_form()
upload_image()
object Model
display_image(filename)
data_page()
about_page()
future_page()
team_page()
Import to
model.py
class Model
load_data()
train()
predict_from_path()
__pre_process_file()
__scale_and_normalize()
__read_and_resize()
Save image file to
Local Directory
load image from filepath
Front-End HTML
Upload.html
data.html
about.html
future.html
team.html

6) RESULT :

Train&Test the Data

```
In [36]: model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
2/2 [==============================] - 2s 959ms/step - loss: 2.2916 - accuracy: 0.5056 - val_loss: 1.2011 - val_accuracy: 0.479
5
Epoch 2/10
2/2 [==============================] - 1s 604ms/step - loss: 1.4341 - accuracy: 0.5000 - val_loss: 0.5572 - val_accuracy: 0.753
4
Epoch 3/10
2/2 [==============================] - 1s 719ms/step - loss: 0.6735 - accuracy: 0.6778 - val_loss: 1.3551 - val_accuracy: 0.479
5
Epoch 4/10
2/2 [==============================] - 1s 672ms/step - loss: 0.8867 - accuracy: 0.6667 - val_loss: 0.9335 - val_accuracy: 0.479
5
Epoch 5/10
2/2 [==============================] - 1s 564ms/step - loss: 0.6447 - accuracy: 0.6944 - val_loss: 0.5196 - val_accuracy: 0.821
9
Epoch 6/10
2/2 [==============================] - 1s 601ms/step - loss: 0.6377 - accuracy: 0.6111 - val_loss: 0.5169 - val_accuracy: 0.753
4
Epoch 7/10
2/2 [==============================] - 1s 600ms/step - loss: 0.6453 - accuracy: 0.6111 - val_loss: 0.5563 - val_accuracy: 0.753
4
Epoch 8/10
2/2 [==============================] - 1s 708ms/step - loss: 0.5721 - accuracy: 0.7000 - val_loss: 0.6855 - val_accuracy: 0.575
3
Epoch 9/10
2/2 [==============================] - 1s 740ms/step - loss: 0.5727 - accuracy: 0.7056 - val_loss: 0.6805 - val_accuracy: 0.575
3
Epoch 10/10
2/2 [==============================] - 1s 582ms/step - loss: 0.5664 - accuracy: 0.7167 - val_loss: 0.5736 - val_accuracy: 0.671
2
```

```
Out[36]: <tensorflow.python.keras.callbacks.History at 0x1689cd2c7c0>
```

```
In [37]: model.save("brain_tumor.h5")
```

```
In [38]: import numpy as np
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image
```

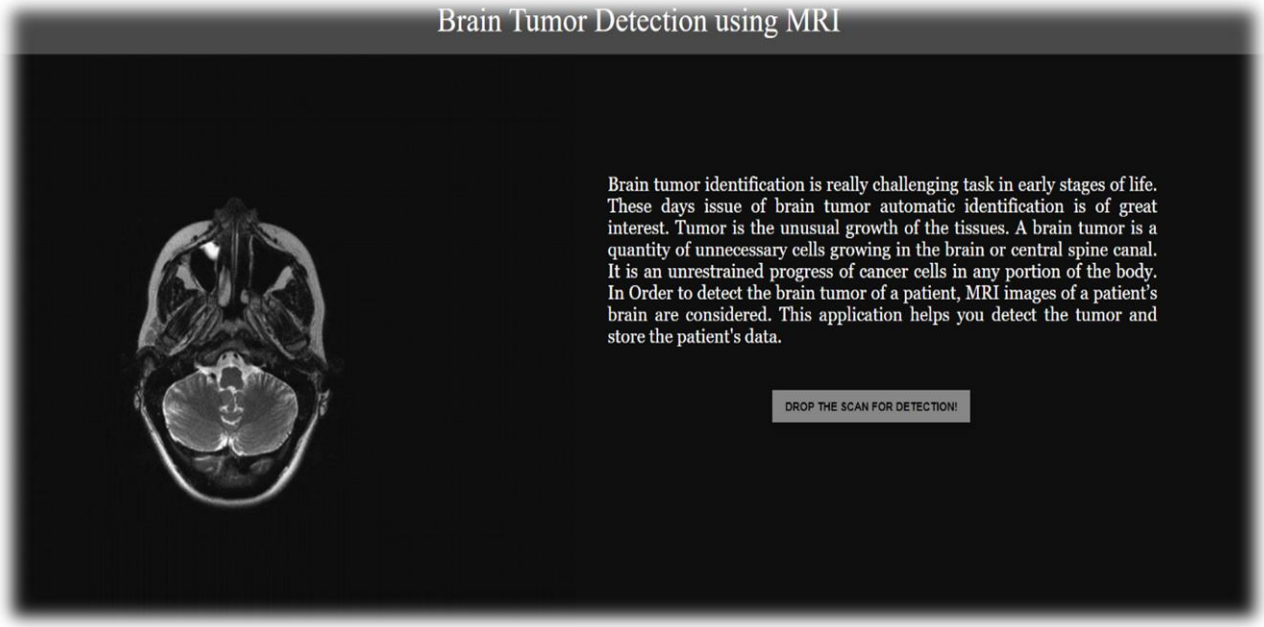
```
In [46]: model=load_model('brain_tumor.h5')
         img=image.load_img(r"F:\ML&DL-Inter\Brain_Tumor_Train_Test_Folders\test_set\no\no 94.jpg",target_size=(64,64))
         x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)
         pred=model.predict_classes(x)
         pred
```

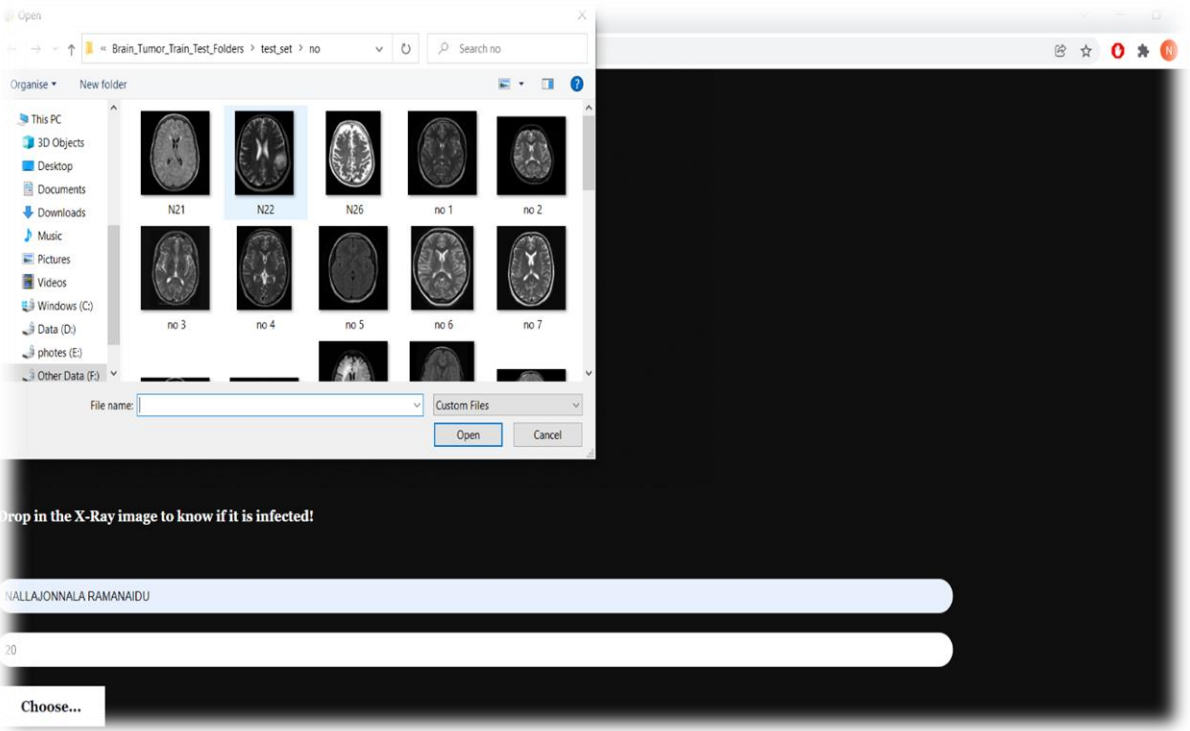
```
Out[46]: array([0], dtype=int64)
```

```
In [47]: index=['no', 'yes']
         print(index[pred[0]])
```

```
no
```

Flask Implementation :


Brain Tumor Detection using MRI
Brain tumor identification is really challenging task in early stages of life. These days issue of brain tumor automatic identification is of great interest. Tumor is the unusual growth of the tissues. A brain tumor is a quantity of unnecessary cells growing in the brain or central spine canal. It is an unrestrained progress of cancer cells in any portion of the body. In Order to detect the brain tumor of a patient, MRI images of a patient's brain are considered. This application helps you detect the tumor and store the patient's data.
DROP THE SCAN FOR DETECTION!

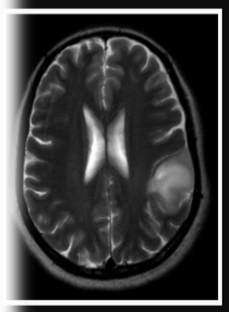

Open
Brain_Tumor_Train_Test_Folders › test_set › no
Search no
Organise
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Data (D:)
photes (E:)
Other Data (F:)
N21
N22
N26
no 1
no 2
no 3
no 4
no 5
no 6
no 7
File name:
Custom Files
Open
Cancel
rop in the X-Ray image to know if it is infected!
NALLAJONNALA RAMANAIDU
20
Choose...

Drop in the X-Ray image to know if it is infected!

NALLAJONNALA RAMANAIDU

20

Choose...



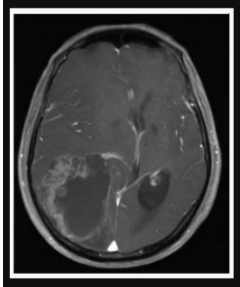
Prediction : You are perfectly fine

Drop in the X-Ray image to know if it is infected!

suleman

21

Choose...



Prediction : You are infected! Please Consult Doctor

7) ADVANTAGES & DISADVANTAGES

Advantages :

- Simple and computationally fast
- Unsupervised,always converge the boundarious of tumor
- Topological changes are naturally possible

Disadvantages :

- Limited applicability to enhancing tumor
- Long computational time, sensitive to noice
- Topological changes are computationally expensive

8) APPLICATIONS :

- The main aim of the applications is tumor identification.
- The main reason behind the development of this application is to provide proper treatment as soon as possible and protect the human life which is in danger.
- This application is helpful to doctors as well as patient.
- The manual identification is not so fast, more accurate and efficient for user. To overcome those problem this application is design.
- It is user friendly application.

9) CONCLUSION :

We proposed a computerized method for the segmentation and identification of a brain tumor using the IBM Watson studio. The input MR images are tested and trained by image pre-processing .These images are pre-processed using an adaptive modelling technique such as sequential, convolution, pooling, flattening for the elimination of noises that are present inside the original image.And then we obtained a '.h5' file which we used in flask to create a web application inorder to detect brain tumor is present or not. The proposed model had obtained an accuracy of 84% and yields promising results without any errors and much less computational time.

10) FUTURE SCOPE :

It is observed on extermination that the proposed approach needs a vast training set for better accurate results; in the field of image processing, the gathering of dataset is a tedious job, and, in few cases, the datasets might not be available. In all such cases, the proposed algorithm must be robust enough for accurate recognition of tumor regions from MRI Images. The proposed approach can be further improvised through in cooperating weakly trained algorithms that can identify the abnormalities with a minimum training data and also self-learning algorithms would aid in enhancing the accuracy of the algorithm and reduce the computational time

11) Source code :

```
import numpy as np
```

```
import os
from keras.preprocessing import image
import pandas as pd
import tensorflow as tf
# Flask utils
from flask import Flask,request, render_template
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend import set_session
from tensorflow.python.keras.models import load_model
global graph
sess = tf.compat.v1.Session()
graph=tf.compat.v1.get_default_graph()
# Define a flask app
app = Flask(__name__)
set_session(sess)
# Load your trained model
model = load_model('brain_tumor.h5')
print('Model loaded. Check http://127.0.0.1:5000/')
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('brainintro.html')
@app.route('/predict1', methods=['GET'])
def predict1():
    # Main page
    return render_template('brainpred2.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
  if request.method == 'POST':
        # Get the file from post request
```

```
        df= pd.read_csv('patient.csv')
        f = request.files['image']
        name=request.form['name']
        age=request.form['age']
         # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join( basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(64, 64))
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        #with graph.as_default():
        #set_session(sess)
        prediction = model.predict(x)[0][0]
        print(prediction)
        if prediction==0:
            text = "You are perfectly fine"
            inp = "No tumor"
        else:
            text = "You are infected! Please Consult Doctor"
            inp="Tumor detected"
df=df.append(pd.DataFrame({'name':[name],'age':[age],'status':[inp]}),ignore_index=True)
        df.to_csv('patient.csv',index = False)
        return text
if __name__ == '__main__':
    app.run(debug=False,threaded = False)
```