CSE 3505

FOUNDATION OF DATA ANALYTICS

VELLORE INSTITUTE OF TECHNOLOGY, CHENNAI

BRAIN TUMOR DETECTION USING MRI SCANS

by

Siddharth Suresh, 20BPS1042 (Vellore Institute of Technology, Chennai)

Kanishka Ghosh, 20BPS1125 (Vellore Institute of Technology, Chennai)

Harsh Deswal, 20BPS1145 (Vellore Institute of Technology, Chennai)

Submitted to

Dr. Priyadarshini R
Associate professor
Vellore Institute of Technology, Chennai

<u>**DECLARATION**</u>

I hereby declare that the report titled **Brain Tumor Detection Using MRI Scans** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr. Priyadarshini R**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Kanishka Ghosh

Reg. No. 20BPS1125

Signature of the Candidate

CERTIFICATE

Certified that this project report entitled **Brain Tumor Detection Using MRI Scans** is a bonafide work of **Harsh Deswal (20BPS1145), Siddharth Suresh (20BPS1042) , Kanishka Ghosh (20BPS1125)** and they carried out the Project work under my supervision and guidance for CSE3505 – Foundations of Data Analytics.

Dr. Priyadarshini R

SCOPE, VIT Chennai

ABSTRACT

The human brain is the major controller of the humanoid system. The abnormal growth and division of cells in the brain lead to a brain tumor, and the further growth of brain tumors leads to brain cancer. The brain tumor is one all the foremost common and, therefore, the deadliest brain diseases that have affected and ruined several lives in the world. Cancer is a disease in the brain in which cancer cells ascend in brain tissues. In human health, Computer Vision plays a significant role, which reduces the human judgment that gives accurate results. CT scans, X-Ray, and MRI scans are the common imaging methods among magnetic resonance imaging (MRI) that are the most reliable and secure. MRI detects every minute objects. Our paper aims to focus on the use of different techniques for the discovery of brain cancer using brain MRI. In this study, we performed pre-processing using the bilateral filter (BF) for removal of the noises that are present in an MR image. This was followed by the binary thresholding and Convolution Neural Network (CNN) segmentation techniques for reliable detection of the tumor region. Training, testing, and validation datasets are used. Based on our machine, we will predict whether the subject has a brain tumor or not. The resultant outcomes will be examined through various performance examined metrics that include accuracy, sensitivity, and specificity. It is desired that the proposed work would exhibit a more exceptional performance over its counterparts. MRI scans are the most common method used to diagnose brain tumors. These tumors are defined by means of the techniques for tissue analysis. However, there are a few things that can lower the quality of MRI images, such as the quality of an MRI device and the low resolution of the images. Additionally, it is challenging to locate tumors in low-resolution images. This limitation can be circumvented using a super-resolution technique. Using open-access datasets, this study proposes using Convolution Neural Network (CNN) algorithms to classify brain tumors based on Artificial Intelligence (AI).Using open-access datasets, this paper uses a novel Discrete Cosine Transform-based image fusion and Convolution Neural Network as a super-resolution and classifier framework to differentiate (or classify) tissue as tumor or not. The framework's performance was examined with and without the super-resolution method, and the super-resolution and ResNet50 architecture achieved an accuracy rate of 92.14 percent. The experiments on MRI images demonstrate that the proposed super-resolution framework is capable of increasing classification accuracy and is based on the Discrete Cosine Transform (DCT), CNN, and ResNet50 (also known as DCT-CNN-ResNet50).

KEYWORDS

Super-resolution; Convolution Neural Network; Discrete cosine transform; Brain tumor; Fusion; Image recognition

CONTENTS

INTRODUCTION

Brain tumor is one of the most rigorous diseases in medical science. An effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. The biopsy procedure requires the neurosurgeon to drill a small hole into the skull from which the tissue is collected. There are many risk factors involving the biopsy test, including bleeding from the tumor and brain causing infection, seizures, severe migraine, stroke, coma and even death. But the main concern with the stereotactic biopsy is that it is not 100% accurate which may result in a serious diagnostic error followed by a wrong clinical management of the disease. Tumor biopsy being challenging for brain tumor patients, non-invasive imaging techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, development of systems for the detection and prediction of the grade of tumors based on MRI data has become necessary. But at first sight of the imaging modality like in Magnetic Resonance Imaging (MRI), the proper visualization of the tumor cells and its differentiation with its nearby soft tissues is somewhat difficult task which may be due to the presence of low illumination in imaging modalities or its large presence of data or several complexity and variance of tumors-like unstructured shape, viable size and unpredictable locations of the tumor. Automated defect detection in medical imaging using machine learning has become the emergent field in several medical diagnostic applications. Its application in the detection of brain tumors in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. Studies in the recent literature have also reported that automatic computerized detection and diagnosis of the disease, based on medical image analysis, could be a good alternative as it would save radiologist time and obtain a tested accuracy. Furthermore, if computer algorithms can provide robust and quantitative measurements of tumor depiction, these automated measurements will greatly aid in the clinical management of brain tumors by freeing physicians from the burden of the manual depiction of tumors. Medical imaging is the technique and process of creating visual representations of the interior of a body for clinical analysis and medical intervention, as well as visual representation of the function of some organs or tissues. Medical imaging seeks to reveal internal structures hidden by the skin and bones, as well as to diagnose and treat disease. Medical imaging also establishes a database of normal anatomy and physiology to make it possible to identify abnormalities. Medical imaging processing refers to handling images by using the computer. This processing includes many types of techniques and operations such as image gaining, storage, presentation, and communication. This process pursues disorder identification and management. This process creates a data bank of the regular structure and function of the organs to make it easy to recognize the anomalies. This process includes both organic and radiological imaging which uses electromagnetic energies (X-rays and gamma), sonography, magnetic scopes, and thermal and isotope imaging. There are many other technologies used to record information about the location and function of the body. Those techniques have many limitations compared to those modulates which produce images. An image processing technique is the usage of a computer to manipulate the digital image. This technique has many benefits such as elasticity, adaptability, data storing, and communication. With the growth of different image resizing techniques, the images can be kept efficiently. This technique has many sets of rules to perform in the images synchronously. The 2D and 3D images can be processed in multiple dimensions. We can use a Deep Learning architecture CNN (Convolution Neural Network) generally known as NN (Neural Network) and VGG 16(visual geometry group) Transfer learning to detect the brain tumor. The performance of the model is to predict whether the image tumor is present or not in the image. If the tumor is present it returns yes otherwise returns no.

LITERATURE REVIEW

Krizhevsky et al. 2012 achieved state-of-the-art results in image classification based on transfer learning solutions upon training a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, he achieved top-1 and top-5 error rates of 37.5% and 17.0% which was considerably better than the previous state-of-the-art. He also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. The neural network, which had 60 million parameters and 650,000 neurons, consisted of five convolutional layers, some of which were followed by max-pooling layers, and three fully-connected layers with a final 1000-way SoftMax. To make training faster, he used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers he employed a recently-developed regularization method called ―dropout‖ that proved to be very effective.

Simonyan & Zisserman 2014 they investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. These findings were the basis of their ImageNet Challenge 2014 submission, where their team secured the first and the second places in the localization and classification tracks respectively. Their main contribution was a thorough evaluation of networks of increasing depth using architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers after training smaller versions of VGG with less weight layers.

Pan & Yang 2010 survey focused on categorizing and reviewing the current progress on transfer learning for classification, regression and clustering problems. In this survey, they discussed the relationship between transfer learning and other related machine learning techniques such as domain adaptation, multitask learning and sample selection bias, as well as covariate shift. They also explored some potential future issues in transfer learning research. In this survey article, they reviewed several current trends of transfer learning.

Szegedyet al.2015 proposed a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. His results seem to yield solid evidence that approximating the expected optimal sparse structure by readily available dense building blocks is a viable method for improving neural networks for computer vision.

He et al., 2015b introduced the ResNet, which utilizes ―skip connections‖ and batch normalization. He presented a residual learning framework to ease the training of networks that are substantially deeper than those used previously. He explicitly reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. He provided comprehensive empirical evidence showing that these residual networks are easier to optimize and can gain accuracy from considerably increased depth. On the ImageNet dataset he evaluated residual nets with a depth of up to 152 layers—8×deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won 1st place on the ILSVRC 2015 classification task. He also presented analysis on CIFAR-10 with 100 and 1000 layers.

PROBLEM STATEMENT

Our study deals with automated brain tumor detection and classification. Normally the anatomy of the brain is analyzed by MRI scans or CT scans. The aim of the paper is tumor identification in brain MR images. The main reason for detection of brain tumors is to provide aid to clinical diagnosis. The aim is to provide an algorithm that guarantees the presence of a tumor by combining several procedures to provide a foolproof method of tumor detection in MR brain images. The methods utilized are filtering, erosion, dilation, threshold, and outlining of the tumor such as edge detection.The focus of this project is MR brain images tumor extraction and its representation in simpler form such that it is understandable by everyone. The objective of this work is to bring some useful information in simpler form in front of the users, especially for the medical staff treating the patient. The aim of this work is to define an algorithm that will result in an extracted image of the tumor from the MR brain image. The resultant image will be able to provide information like size, dimension and position of the tumor, and its boundary provides us with information related to the tumor that can prove useful for various cases, which will provide a better base for the staff to decide the curing procedure. Finally, we detect whether the given MR brain image has a tumor or not using the Convolution Neural Network.

PROPOSED MODEL

Our brain tumor images are composed of 128 by 128 pixels which make up for 16,384 pixels. Each pixel is fed as input to each neuron of the first layer. Neurons of one layer are connected to neurons of the next layer through channels .Each of these channels is assigned a numerical value known as weight‘. The inputs are multiplied to the corresponding weight and their sum is sent as input to the neurons in the hidden layer. Each of these neurons is associated with a numerical value called the bias’s which is then added to the input sum. This value is then passed through a threshold function called the activation function‘. The result of the activation function determines if the neuron will get activated or not. An activated neuron transmits data to the neurons of the next layer over the channels. In this manner the data is propagated through the network this is called forward propagation‘. In the output layer the neuron with the highest value fires and determines the output. The values are basically probable. The predicted output is compared against the actual output to realize the _error‘ in prediction. The magnitude of the error gives an indication of the direction and magnitude of change to reduce the error. This information is then transferred backward through our network. This is known as back propagation‘. Now based on this information the weights are adjusted. This cycle of forward propagation and backpropagation is iteratively performed with multiple inputs. This process continues until our weights are assigned such that the network can predict the type of tumor correctly in most of the cases. This brings our training process to an end. NN may take hours or even months to train but time is a reasonable trade-off when compared to its scope Several experiments show that after preprocessing MRI images, neural network classification algorithm was the best more specifically CNN(Convolutional Neural Network) as compared to Support Vector Machine(SVM),Random Forest Field.

COMPONENTS OF THE MODEL

1. GOOGLE COLAB: The Basics. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Fig 1. Google Colab

2. RestNet50 Model : Residual network (ResNet) architecture was proposed by He Kaiming in 2015. ResNet consists of adding residual values and residual blocks to the architectural model. ResNet architecture kept the performance of the model while increasing depth. In addition, the number of parameters, which is an important factor in computational complexity, was reduced in this model.

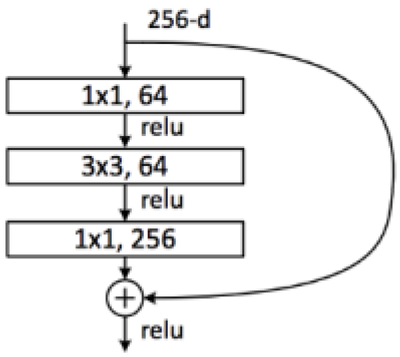


Fig 2. ResNet50 Architecture

3. Streamlit : Python-based open-source app framework Streamlit.It expedites the development of web applications for data science and machine learning.Scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib, and other major Python libraries are compatible with it.

BLOCK DIAGRAM

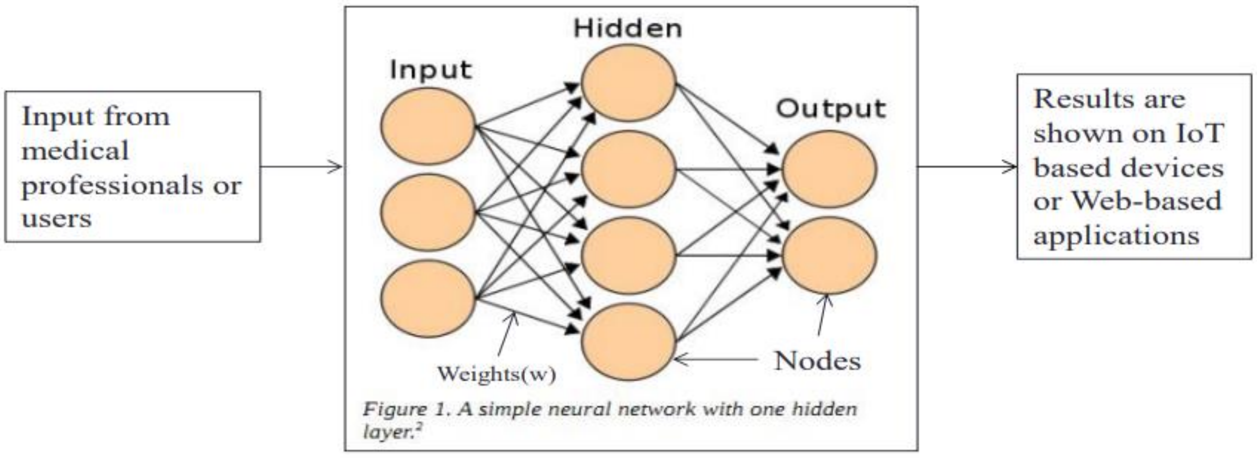

Input from medical professionals or users
Hidden
Input
Output
Weights(w)
Nodes
Figure 1. A simple neural network with one hidden layer.2
Results are shown on IoT based devices or Web-based applications

Fig 3. CNN architecture

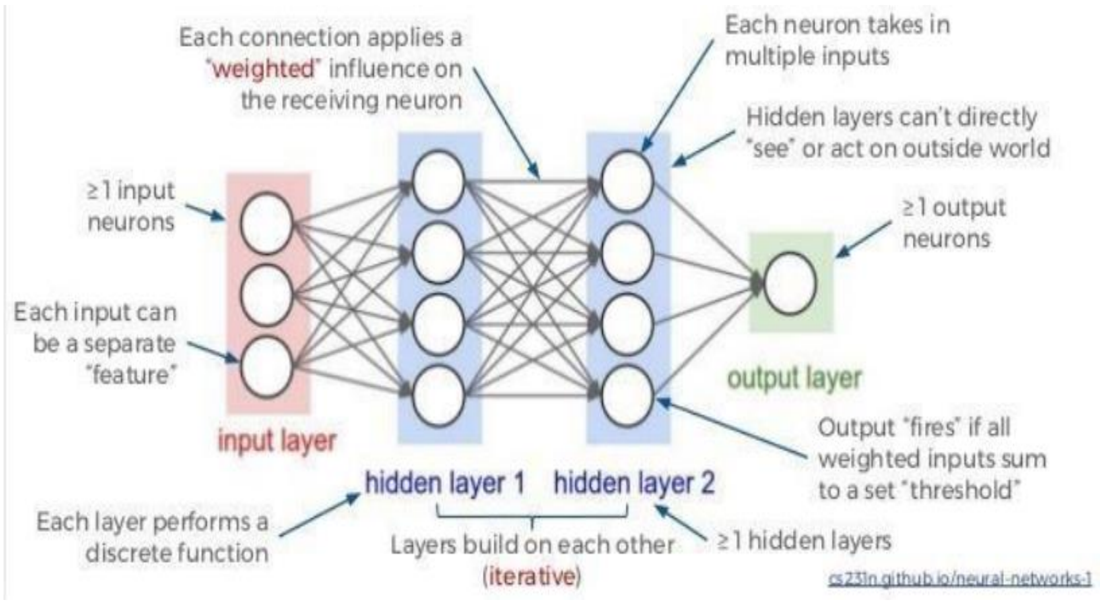

Each connection applies a "weighted" influence on the receiving neuron
Each neuron takes in multiple inputs
Hidden layers can't directly "see" or act on outside world
≥1 input neurons
≥1 output neurons
Each input can be a separate "feature"
output layer
input layer
Output "fires" if all weighted inputs sum to a set "threshold"
hidden layer 1
hidden layer 2
Each layer performs a discrete function
Layers build on each other (iterative)
≥1 hidden layers
cs231n.github.io/neural-networks-1

Fig 4. VGG16 architecture

FLOWCHART

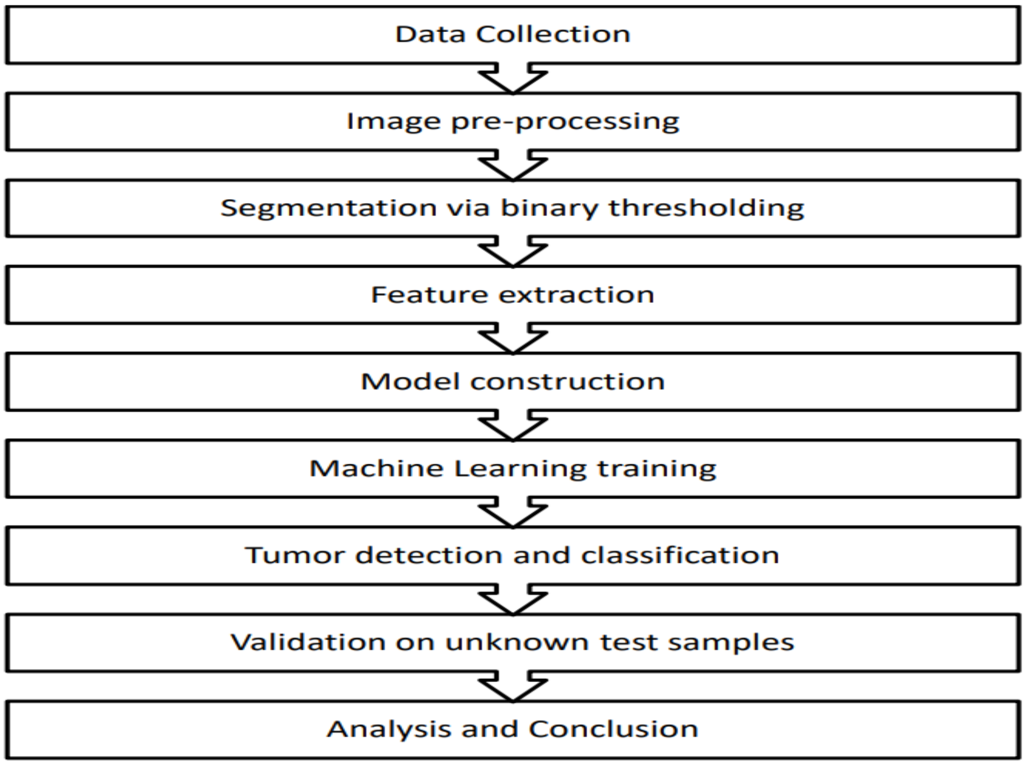

Data Collection
Image pre-processing
Segmentation via binary thresholding
Feature extraction
Model construction
Machine Learning training
Tumor detection and classification
Validation on unknown test samples
Analysis and Conclusion

Fig 5. Project workflow

MODULE FLOW

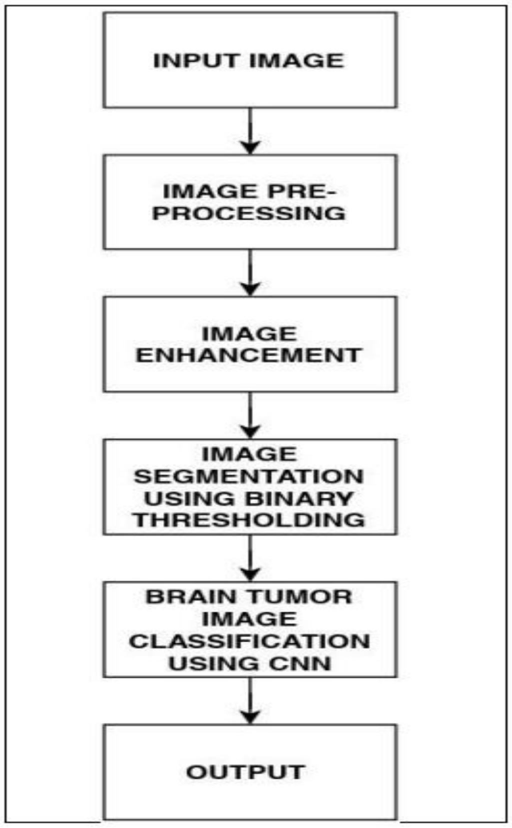

INPUT IMAGE
IMAGE PRE-PROCESSING
IMAGE ENHANCEMENT
IMAGE SEGMENTATION USING BINARY THRESHOLDING
BRAIN TUMOR IMAGE CLASSIFICATION USING CNN
OUTPUT

Fig 6. Module workflow

This provides the architecture of the system 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, image enhancement, Image segmentation using binary thresholding and the brain tumor classification using Convolutional Neural Network. Finally, the output is observed after all the above-mentioned steps are completed.

Module 1: Image Preprocessing

The Brain MRI image dataset has been downloaded from the Kaggle. The MRI dataset consists of around 1900 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 preprocessing are the reduction of impulsive noises and image resizing. In the initial phase, we convert the brain MRI image into its corresponding gray-scale image. The removal of unwanted noise is done using the adaptive bilateral filtering technique to remove the distorted noises that are present in the brain picture. This improves the diagnosis and increases the classification accuracy rate.Bilateral filter is a non-linear, noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight is based on the Gaussian distribution. Bilateral filtering smooth images while conserving edges utilizing a nonlinear grouping of neighboring image pixels. This filtering technique is simple, local, and concise. It syndicates a gray level grounded on their likeness and the symmetrical nearness and chooses near values to farther values in both range and domain.

Module 2: Image Enhancement

Image enhancement is a technique used to improve the image quality and perceptibility by using computer-aided software. This technique includes both objective and subjective enhancements. This technique includes points and local operations. The local operations depend on the district input pixel values. Image enhancement has two types: spatial and transform domain techniques. The spatial techniques work directly on the pixel level, while the transform technique works on Fourier and later the spatial technique. Edge detection is a segmentation technique that uses border recognition of strictly linked objects or regions. This technique identifies the discontinuity of the objects. This technique is used mainly in image study and to recognize the parts of the image where a huge variation in intensity arises.

Module 3: Image Segmentation Using Binary Threshold

Image segmentation is a technique of segregating the image into many parts. The basic aim of this segregation is to make the images easy to analyze and interpret while preserving the quality. This technique is also used to trace the objects' borders within the images. This technique labels the pixels according to their intensity and characteristics. Those parts represent the entire original image and acquire its characteristics such as intensity and similarity. The image segmentation technique is used to create contours of the body for clinical purposes. Segmentation is used in machine perception, malignant disease analysis, tissue volumes, anatomical and functional analyses, virtual reality visualization, and anomaly analysis, and object definition and detection. Segmentation methods have the ability to detect or identify the abnormal portion from the image which is useful for analyzing the size, volume, location, texture and shape of the extracted image. MR image segmentation with the aid of preserving the threshold information, which is convenient to identify the broken regions extra precisely. It was a trendy surmise that the objects that are placed in close propinquity might be sharing similar houses and characteristics.

Module 4: Brain Tumor Image Classification Using Convolutional Neural Network

Classification is the best approach for identification of images like any kind of medical imaging. All classification algorithms are based on the prediction of an image, where one or more features and that each of these features belongs to one of several classes. An automatic and reliable classification method Convolutional Neural Network (CNN) will be used since it is robust in structure which helps in identifying every minute details. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNet can learn these filters/characteristics. A ConvNet can successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the ConvEnt is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. 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.
- Convolution2D is used to make the convolutional network that deals with the images.
- MaxPooling2D layer is used to add the pooling layers.
- Flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer.
- Dense adds the fully connected layer to the neural network

EVALUATION METRICS

True Positive (TP) is the HGG class predicted in the presence of the LGG class of the glioma. True Negative (TN) is the LGG class predicted in the absence of the HGG class of glioma. False Positive (FP) is prediction of HGG class in the absence of LGG class. False Negative (FN) is prediction of LGG class in the absence of HGG class.

Accuracy is the most intuitive performance measure. Accuracy is the amount of correct prediction made by the total number of predictions made.

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

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

$$Precision = \frac{TP}{TP + FP}$$

Recall is also known as sensitivity. It is the fraction of the total number of relative relevant instances that were retrieved.

$$Recall = \frac{TP}{TP + FN}$$

When we have an unbalanced dataset F 1 Score favored over accuracy because it takes both false positives and false negatives into account. F-measures are used to balance the ratio of false negatives using a weighting parameter (beta) it is given as

$$F = P * \ R\frac{(1 + \beta\ )}{(P + R\ )\beta}$$

Other performance metrics used are: sensitivity, specificity and error rate. Sensitivity represents the probability of predicting actual HGG class. Specificity value defines prediction of LGG class. They allow us to determine potential of over- or under segmentations of the tumor sub-regions. The error rate (ERR) is the amount of predicted classes that have been incorrectly classified by a decision model. The overall classification is also provided by the Area under the Curve (AUC) that represents better classification if the area under the curve is more. All these performances' metric is evaluated for FLAIR sequences.

The DSC(dice similarity coefficient) measures the overlap between the manual delineated brain tumor regions and the segmentation results of our fully automatic method that is. Mathematically, dice score/DSC is the number of false positives divided by the number of positives added with the number of false positives.

$$DSC = \frac{2 * TP}{TP + FP + FN}.$$

SYSTEM CONFIGURATION

1. SOFTWARE REQUIREMENTS

Python: Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

PIP: It is the package management system used to install and manage software packages written in Python.

NumPy: NumPy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas: Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code that is purely written in C or Python. We can analyze data in pandas with

1. Series

2. Data frames

Anaconda: 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).

Colab: 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: Tensor flow 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. It is used for both research and production at Google.

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.

OpenCV: OpenCV (Open source computer vision) 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 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 Open Vision Capsules. which is a portable format, compatible with all other formats.

2. HARDWARE REQUIREMENTS

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

DATASET

The data set used during the experiments was obtained from Kaggle. This data set consists of two classes. There is no tumor in first-class data. The data in the second class are images of the patient with tumor

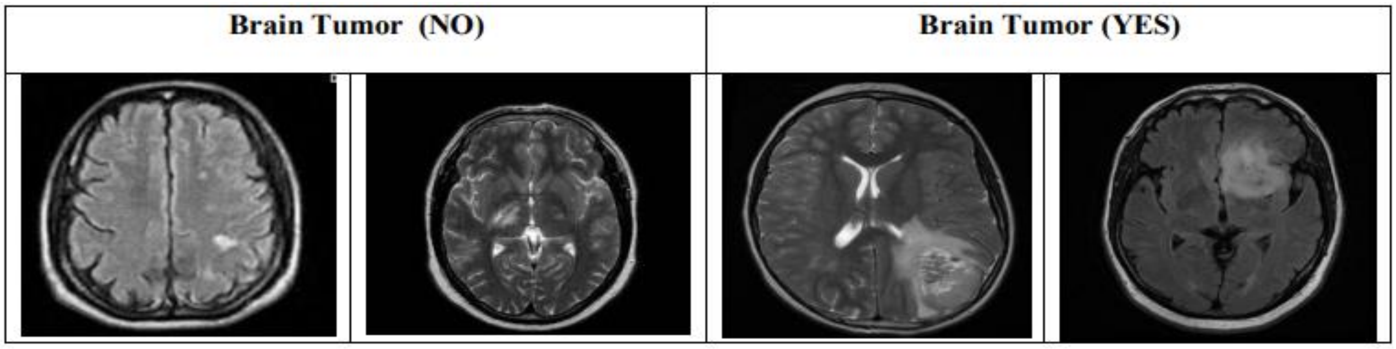


Fig 7. Sample data

CODE

Importing necessary Libraries

```
from IPython.display import clear_output
!pip install imutils
clear_output()
import numpy as np
import pandas as pd
import cv2
from PIL import Image
import scipy
import tensorflow as tf
from tensorflow.keras.applications import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.losses import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.preprocessing.image import *
from tensorflow.keras.utils import *
# import pydot
from sklearn.metrics import *
from sklearn.model_selection import *
import tensorflow.keras.backend as K
from tqdm import tqdm, tqdm_notebook
from colorama import Fore
import json
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
from skimage.io import *
%config Completer.use_jedi = False
import time
from sklearn.decomposition import PCA
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import lightgbm as lgb
import xgboost as xgb
import numpy as np
from tqdm import tqdm
import cv2
import os
import shutil
import itertools
import imutils
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16, preprocess_input
from keras import layers
from keras.models import Model, Sequential
from keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping
init_notebook_mode(connected=True)
RANDOM_SEED = 123
print("All modules have been imported")
```

Creating a required directory structure

```
!apt-get install tree
clear_output()
# create new folders
!mkdir TRAIN TEST VAL TRAIN/YES TRAIN/NO TEST/YES TEST/NO VAL/YES VAL/NO
!tree -d
```

Importing the Data

```
from google.colab import drive
drive.mount('/content/Brain_Tumor_Detection/')
IMG_PATH = "../content/Brain_Tumor_Detection/MyDrive/certificates/Brain_Tumor_Detection"

# split the data by train/val/test
```

```
ignored = {"pred"}
# split the data by train/val/test
for CLASS in os.listdir(IMG_PATH):
    if CLASS not in ignored:
        if not CLASS.startswith('.'):
            IMG_NUM = len(os.listdir(IMG_PATH +"/"+ CLASS))
            for (n, FILE_NAME) in enumerate(os.listdir(IMG_PATH +"/"+ CLASS)):
                img = IMG_PATH+ '/' +  CLASS + '/' + FILE_NAME
                if n < 300:
                    shutil.copy(img, 'TEST/' + CLASS.upper() + '/' + FILE_NAME)
                elif n < 0.8*IMG_NUM:
                    shutil.copy(img, 'TRAIN/'+ CLASS.upper() + '/' + FILE_NAME)
                else:
                    shutil.copy(img, 'VAL/'+ CLASS.upper() + '/' + FILE_NAME)
```

Function for Resizing the images

```
def load_data(dir_path, img_size=(100,100)):
    """
    Load resized images as np.arrays to workspace
    """
    X = []
    y = []
    i = 0
    labels = dict()
    for path in tqdm(sorted(os.listdir(dir_path))):
        if not path.startswith('.'):
            labels[i] = path
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i += 1
    X = np.array(X)
    y = np.array(y)
    print(f'{len(X)} images loaded from {dir_path} directory.')
    return X, y, labels
```

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize = (6,6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    cm = np.round(cm,2)
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

TRAIN_DIR = 'TRAIN/'
TEST_DIR = 'TEST/'
VAL_DIR = 'VAL/'
IMG_SIZE = (224,224)
X_train, y_train, labels = load_data(TRAIN_DIR, IMG_SIZE)
X_test, y_test, _ = load_data(TEST_DIR, IMG_SIZE)
X_val, y_val, _ = load_data(VAL_DIR, IMG_SIZE)
```

Plotting the number of samples in Training, Validation and Test sets

```
y = dict()
y[0] = []
y[1] = []
for set_name in (y_train, y_val, y_test):
    y[0].append(np.sum(set_name == 0))
    y[1].append(np.sum(set_name == 1))

trace0 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[0],
    name='No',
    marker=dict(color='#33cc33'),
    opacity=0.7
)
trace1 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[1],
    name='Yes',
    marker=dict(color='#ff3300'),
    opacity=0.7
)
data = [trace0, trace1]
layout = go.Layout(
    title='Count of classes in each set',
    xaxis={'title': 'Set'},
    yaxis={'title': 'Count'}
)
fig = go.Figure(data, layout)
iplot(fig)
```

Visualize the images we are working with

```
def plot_samples(X, y, labels_dict, n=50):
    """
    Creates a gridplot for desired number of images (n) from the specified set
    """
    for index in range(len(labels_dict)):
        imgs = X[np.argwhere(y == index)][:n]
        j = 10
        i = int(n/j)

        plt.figure(figsize=(15,6))
        c = 1
        for img in imgs:
            plt.subplot(i,j,c)
            plt.imshow(img[0])

            plt.xticks([])
            plt.yticks([])
            c += 1
        plt.suptitle('Tumor: {}'.format(labels_dict[index]))
        plt.show()
plot_samples(X_train, y_train, labels, 30)
```

Cropping the images

```
def crop_imgs(set_name, add_pixels_value=0):
    """
    Finds the extreme points on the image and crops the rectangular out of them
    """
    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)
        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-
ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)

import imutils
img = cv2.imread('./VAL/NO/no150.jpg')
img = cv2.resize(
            img,
            dsize=IMG_SIZE,
            interpolation=cv2.INTER_CUBIC
        )
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
gray = cv2.GaussianBlur(gray, (5, 5), 0)

# threshold the image, then perform a series of erosions +
# dilations to remove any small regions of noise
thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
thresh = cv2.erode(thresh, None, iterations=2)
thresh = cv2.dilate(thresh, None, iterations=2)

# find contours in thresholded image, then grab the largest one
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
c = max(cnts, key=cv2.contourArea)

# find the extreme points
extLeft = tuple(c[c[:, :, 0].argmin()][0])
extRight = tuple(c[c[:, :, 0].argmax()][0])
```

```
extTop = tuple(c[c[:, :, 1].argmin()][0])
extBot = tuple(c[c[:, :, 1].argmax()][0])

# add contour on the image
img_cnt = cv2.drawContours(img.copy(), [c], -1, (0, 255, 255), 4)

# add extreme points
img_pnt = cv2.circle(img_cnt.copy(), extLeft, 8, (0, 0, 255), -1)
img_pnt = cv2.circle(img_pnt, extRight, 8, (0, 255, 0), -1)
img_pnt = cv2.circle(img_pnt, extTop, 8, (255, 0, 0), -1)
img_pnt = cv2.circle(img_pnt, extBot, 8, (255, 255, 0), -1)

# crop
ADD_PIXELS = 0
new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-
ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
```

Visualizing how the cropping works

```
plt.figure(figsize=(15,6))
plt.subplot(141)
plt.imshow(img)
plt.xticks([])
plt.yticks([])
plt.title('Step 1. Get the original image')
plt.subplot(142)
plt.imshow(img_cnt)
plt.xticks([])
plt.yticks([])
plt.title('Step 2. Find the biggest contour')
plt.subplot(143)
plt.imshow(img_pnt)
plt.xticks([])
plt.yticks([])
plt.title('Step 3. Find the extreme points')
plt.subplot(144)
plt.imshow(new_img)
plt.xticks([])
plt.yticks([])
plt.title('Step 4. Crop the image')
plt.show()

X_train_crop = crop_imgs(set_name=X_train)
X_val_crop = crop_imgs(set_name=X_val)
X_test_crop = crop_imgs(set_name=X_test)
```

Visualizing the images after being cropped

```
plot_samples(X_train_crop, y_train, labels, 30)
def save_new_images(x_set, y_set, folder_name):
    i = 0
    for (img, imclass) in zip(x_set, y_set):
        if imclass == 0:
            cv2.imwrite(folder_name+'NO/'+str(i)+'.jpg', img)
        else:
            cv2.imwrite(folder_name+'YES/'+str(i)+'.jpg', img)
        i += 1
# saving new images to the folder
!mkdir TRAIN_CROP TEST_CROP VAL_CROP TRAIN_CROP/YES TRAIN_CROP/NO TEST_CROP/YES
TEST_CROP/NO VAL_CROP/YES VAL_CROP/NO

save_new_images(X_train_crop, y_train, folder_name='TRAIN_CROP/')
save_new_images(X_val_crop, y_val, folder_name='VAL_CROP/')
save_new_images(X_test_crop, y_test, folder_name='TEST_CROP/')
```

Resizing the images

```
def preprocess_imgs(set_name, img_size):
    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
        set_new.append(preprocess_input(img))
    return np.array(set_new)
X_train_prep = preprocess_imgs(set_name=X_train_crop, img_size=IMG_SIZE)
X_test_prep = preprocess_imgs(set_name=X_test_crop, img_size=IMG_SIZE)
X_val_prep = preprocess_imgs(set_name=X_val_crop, img_size=IMG_SIZE)
plot_samples(X_train_prep, y_train, labels, 30)
```

Image Augmentation

```
demo_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.05,
    height_shift_range=0.05,
    rescale=1./255,
    shear_range=0.05,
    brightness_range=[0.1, 1.5],
    horizontal_flip=True,
    vertical_flip=True
)

os.mkdir('preview')
x = X_train_crop[0]
x = x.reshape((1,) + x.shape)

i = 0
for batch in demo_datagen.flow(x, batch_size=1, save_to_dir='preview', save_prefix='aug_img', save_format='jpg'):
    i += 1
    if i > 20:
        break

plt.imshow(X_train_crop[0])
plt.xticks([])
plt.yticks([])
plt.title('Original Image')
plt.show()

plt.figure(figsize=(15,6))
```

```
i = 1
for img in os.listdir('preview/'):
    img = cv2.cv2.imread('preview/' + img)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.subplot(3,7,i)
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])
    i += 1
    if i > 3*7:
        break
plt.suptitle('Augemented Images')
plt.show()

!rm -rf preview/
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'
RANDOM_SEED = 42
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)



train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    seed=RANDOM_SEED
)


validation_generator = test_datagen.flow_from_directory(
    VAL_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=16,
    class_mode='binary',
    seed=RANDOM_SEED
)
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'
RANDOM_SEED = 42
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)


train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    seed=RANDOM_SEED
)


validation_generator = test_datagen.flow_from_directory(
    VAL_DIR,
    color_mode='rgb',
    target_size=IMG_SIZE,
    batch_size=16,
    class_mode='binary',
    seed=RANDOM_SEED
)
```

Creating the model

```
base_Neural_Net= ResNet50(input_shape=(224,224,3), weights='imagenet', include_top=False)
model=Sequential()
model.add(base_Neural_Net)
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(256,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))

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


model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy' , 'AUC']
)

model.summary()
```

TRAINING

```
EPOCHS = 15
es = EarlyStopping(
    monitor='val_acc',
    mode='max',
    patience=6
)
history = model.fit_generator(
    train_generator,
    steps_per_epoch=15,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
    callbacks=[es]
)
```

Validating with the training set

```
predictions = model.predict(X_train_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_train, predictions)
print('Train Accuracy = %.2f' % accuracy)

confusion_mtx = confusion_matrix(y_train, predictions)
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), normalize=False)
```

Validating with the Validation set

```
predictions = model.predict(X_val_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_val, predictions)
print('Val Accuracy = %.2f' % accuracy)

confusion_mtx = confusion_matrix(y_val, predictions)
```

```
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), normalize=False)
```

Validating with the Test set

```
# validate on test set
predictions = model.predict(X_test_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_test, predictions)
print('Test Accuracy = %.2f' % accuracy)

confusion_mtx = confusion_matrix(y_test, predictions)
cm = plot_confusion_matrix(confusion_mtx, classes = list(labels.items()), normalize=False)
prob_pred = model.predict_proba(X_test_prep)
```

Other performance metrics on the test set

```
from sklearn import metrics
print('Accuracy score is :', np.round(metrics.accuracy_score(y_test, predictions),4))
print('Precision score is :', np.round(metrics.precision_score(y_test, predictions, average='weighted'),4))
print('Recall score is :', np.round(metrics.recall_score(y_test, predictions, average='weighted'),4))
print('F1 Score is :', np.round(metrics.f1_score(y_test, predictions, average='weighted'),4))
print('ROC AUC Score is :', np.round(metrics.roc_auc_score(y_test, prob_pred,multi_class='ovo',
average='weighted'),4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test, predictions),4))

print('\t\tClassification Report:\n', metrics.classification_report(y_test, predictions))
```

Saving the model

```
model.save('RestNet50model.h5')
```

STREAMLIT CODE FOR HOSTING

```
import streamlit as st
import numpy as np
import cv2
import imutils
```

```
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import urllib.request

urllib.request.urlretrieve(
        'https://github.com/siddhsuresh/CSE3505-MRI-Classifier/raw/main/app/model.h5', 'model.h5')
model = load_model('model.h5')

st.set_page_config(
    page_title="CSE3505 MRI Classifier",
    page_icon=":brain:",
    initial_sidebar_state="expanded",
    menu_items={
        'About': "CSE3505 J Component Final Review - MRI Brain Tumor Detection by Siddharth Suresh, Harsh Deshwal,
Kanishka Ghosh",

    }
)

def crop_imgs(set_name, add_pixels_value=0):
    """
    Finds the extreme points on the image and crops the rectangular out of them
    """
    set_new = []
    for img in set_name:
        #conver image to cv2
        print(img.shape)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-
ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)

def preprocess_imgs(set_name, img_size):
    """
    Resize and apply VGG-15 preprocessing
    """
    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
        set_new.append(preprocess_input(img))
    return np.array(set_new)

def load_image():
    uploaded_file = st.file_uploader(label='Pick an image to test')
    if uploaded_file is not None:
        image_data = uploaded_file.getvalue()
        st.image(image_data)
        return image_data
```

```
def main():
    st.header('CSE3505 J Component Final Review')
    st.title('MRI Brain Tumor Detection')
    st.subheader('Presented by')
    st.markdown('''
    - Harsh Deswal 20BPS1145
    - Siddharth Suresh 20BPS1042
    - Kanishka Ghosh 20BPS1125
    ''')
    st.header('Upload the scan to test')
    image_data = load_image()
    if image_data is not None:
        # get image
        img = cv2.imdecode(np.frombuffer(image_data, np.uint8), -1)
        # make the size 224x224
        # crop the image
        img = cv2.resize(img, (224, 224),interpolation=cv2.INTER_CUBIC)
        print(img.shape)
        img = crop_imgs(np.array([img]), add_pixels_value=0)[0]
        print(img.shape)
        img = cv2.resize(img, (224, 224),interpolation=cv2.INTER_CUBIC)
        print(img.shape)
        #preprocess
        img = preprocess_input(img)
        #predict
        prediction = model.predict(np.array([img]))
        #get the class
        class_ = np.argmax(prediction)
        #get the probability
        prob = prediction[0][class_]
        #display the result
        st.markdown(f'''
          ### Accuracy of Model is  <span style="color:#2FA4FF">**92%**</span>
        ''',unsafe_allow_html=True)
        if prob > 0.5:
            st.markdown(f'''
            ##### The model predicts that the image <span style="color:#F24C4C">**has a tumor**</span> with a
probability of **{prob}**
```

```
            ''',unsafe_allow_html=True)
            # Select random number from 80 to 100 and assign it to a severity variable
            severity = np.random.randint(80, 100)
            st.markdown(f'''
            ##### The tumor has a <span style="color:#F24C4C">severity of **{severity}%**</span>
            ''',unsafe_allow_html=True)
        else:
            st.markdown(f'''
            ##### The model predicts that the image <span style="color:#5FBDB0">**does not have a tumor**</span> with
a probability of **{1-prob}**
            ''',unsafe_allow_html=True)

if __name__ == '__main__':
    main()
```

RESULT

Deep learning architectures were used for the classification of brain tumor images. There are certain criteria that express the performance of the models in deep learning . All these performance metrics were calculated using Confusion Matrix .

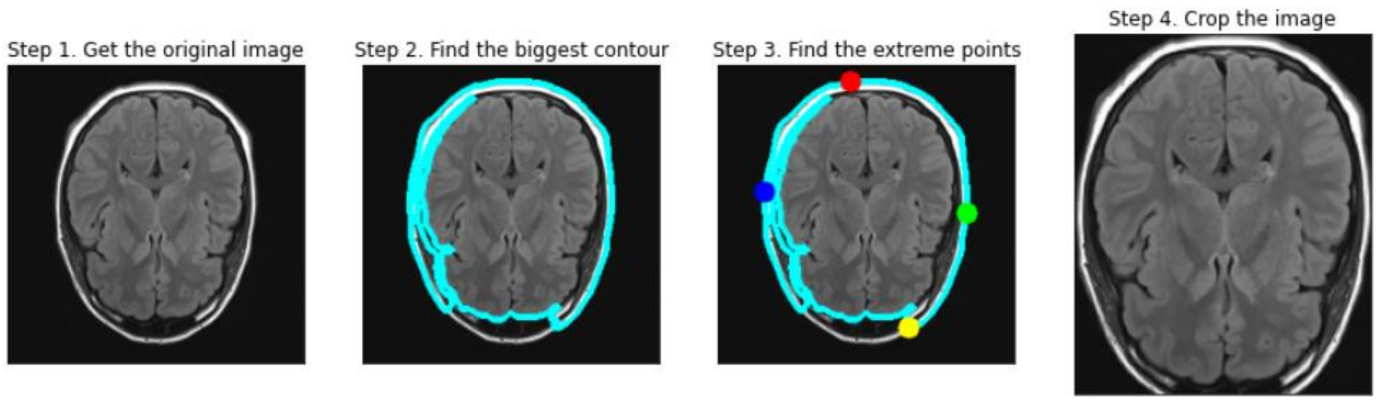

Step 1. Get the original image
Step 2. Find the biggest contour
Step 3. Find the extreme points
Step 4. Crop the image

Fig 8. Cropping of image workflow

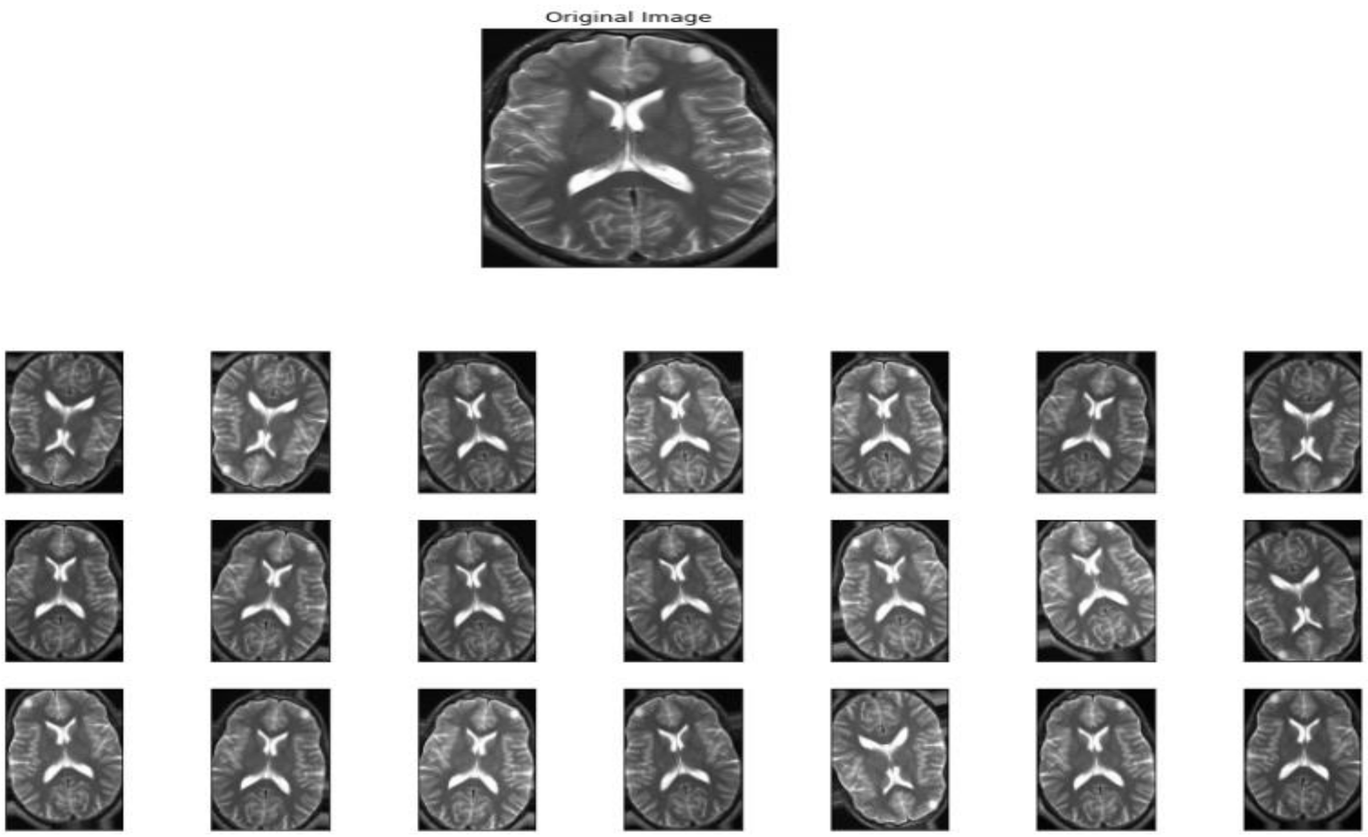


Fig 9. Augmented Images

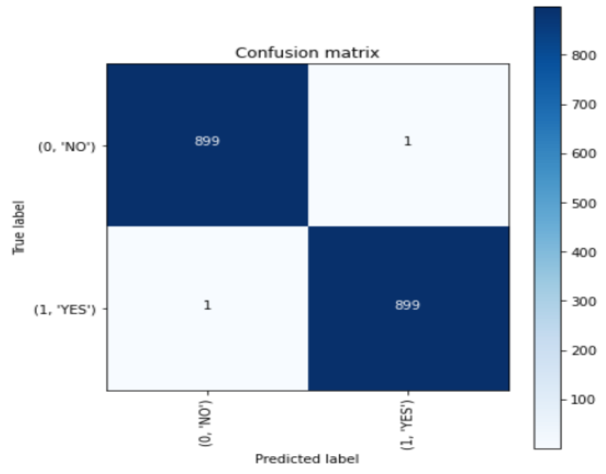


Fig 10. Confusion matrix for Training data

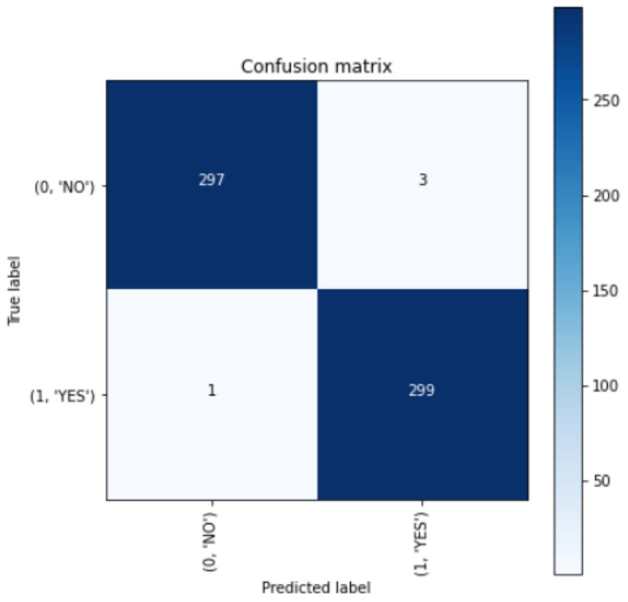


Fig 11. Confusion matrix for Test data

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

$Sensitivity = \frac{TP}{TP+FN}$

$Specificity = \frac{TN}{TN+FP}$

$Precision = \frac{TP}{TP+FP}$

$Negative\ Predictive\ Value(NPV) = TN / (TN + FN)$

$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$

```
Accuracy score is : 0.92
Precision score is : 0.9212
Recall score is : 0.92
F1 Score is : 0.9199
ROC AUC Score is : 0.9831
Cohen Kappa Score: 0.84
                Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.89      0.92       300
           1       0.90      0.95      0.92       300

    accuracy                           0.92       600
   macro avg       0.92      0.92      0.92       600
weighted avg       0.92      0.92      0.92       600
```

CONCLUSION

In this paper, it was aimed to develop a method to diagnose a disease that changes human life completely negatively. The sooner the disease is diagnosed, the sooner the treatment process begins. In this study, deep learning methods were used to detect brain tumors from MRI images. The purpose of this classification process is to assist the doctor. The Resnet50 architecture achieved the highest accuracy rate during the experiments. The accuracy rate of the Resnet50 architecture is 92 percent. The scientific world continues to work on both diagnosing and treating brain tumors. In the future, architectures that will give higher accuracy can be developed for brain tumor diagnosis. We will try to develop a new method based on convolutional neural networks in the near future. With this model, we will try to achieve higher accuracy than any known deep learning method.

FUTURE SCOPE

It is observed on extermination that the proposed approach needs a vast training set for better accurate results; in the field of medical image processing, the gathering of medical data 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 improved through cooperating weakly trained algorithms that can identify the abnormalities with a minimum training data and self-learning algorithms would aid in enhancing the accuracy of the algorithm and reduce the computational time.

PUBLICITY

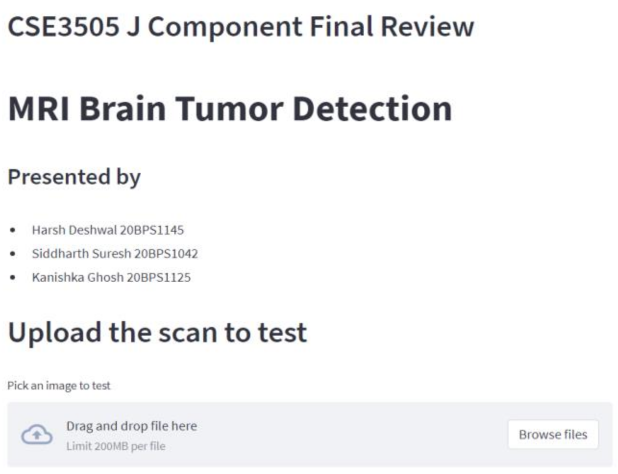
We have used Streamlit for hosting the model . In this we will be asking the user to upload an MRI scan image and our system will detect whether there is a tumor or not with the severity of the tumor.

Link : https://cse3505-mri-classifier.streamlit.app/

STREAMLIT CODE FOR HOSTING

```
import streamlit as st
import numpy as np
import cv2
import imutils
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import urllib.request

urllib.request.urlretrieve(
        'https://github.com/siddhsuresh/CSE3505-MRI-Classifier/raw/main/app/model.h5', 'model.h5')
model = load_model('model.h5')

st.set_page_config(
    page_title="CSE3505 MRI Classifier",
    page_icon=":brain:",
    initial_sidebar_state="expanded",
    menu_items={
        'About': "CSE3505 J Component Final Review - MRI Brain Tumor Detection by Siddharth Suresh, Harsh Deshwal,
Kanishka Ghosh",

    }
)

def crop_imgs(set_name, add_pixels_value=0):
    """
    Finds the extreme points on the image and crops the rectangular out of them
    """
    set_new = []
    for img in set_name:
        #conver image to cv2
        print(img.shape)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS, extLeft[0]-
ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)

def preprocess_imgs(set_name, img_size):
    """
    Resize and apply VGG-15 preprocessing
    """
    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
        set_new.append(preprocess_input(img))
    return np.array(set_new)

def load_image():
    uploaded_file = st.file_uploader(label='Pick an image to test')
    if uploaded_file is not None:
        image_data = uploaded_file.getvalue()
        st.image(image_data)
        return image_data
```

```
def main():
    st.header('CSE3505 J Component Final Review')
    st.title('MRI Brain Tumor Detection')
    st.subheader('Presented by')
    st.markdown('''
    - Harsh Deswal 20BPS1145
    - Siddharth Suresh 20BPS1042
    - Kanishka Ghosh 20BPS1125
    ''')
    st.header('Upload the scan to test')
    image_data = load_image()
    if image_data is not None:
        # get image
        img = cv2.imdecode(np.frombuffer(image_data, np.uint8), -1)
        # make the size 224x224
        # crop the image
        img = cv2.resize(img, (224, 224),interpolation=cv2.INTER_CUBIC)
        print(img.shape)
        img = crop_imgs(np.array([img]), add_pixels_value=0)[0]
        print(img.shape)
        img = cv2.resize(img, (224, 224),interpolation=cv2.INTER_CUBIC)
        print(img.shape)
        #preprocess
        img = preprocess_input(img)
        #predict
        prediction = model.predict(np.array([img]))
        #get the class
        class_ = np.argmax(prediction)
        #get the probability
        prob = prediction[0][class_]
        #display the result
        st.markdown(f'''
          ### Accuracy of Model is  <span style="color:#2FA4FF">**92%**</span>
        ''',unsafe_allow_html=True)
        if prob > 0.5:
            st.markdown(f'''
            ##### The model predicts that the image <span style="color:#F24C4C">**has a tumor**</span> with a probability of **{prob}**
            ''',unsafe_allow_html=True)
            # Select random number from 80 to 100 and assign it to a severity variable
            severity = np.random.randint(80, 100)
            st.markdown(f'''
            ##### The tumor has a <span style="color:#F24C4C">severity of **{severity}%**</span>
            ''',unsafe_allow_html=True)
        else:
            st.markdown(f'''
            ##### The model predicts that the image <span style="color:#5FBDB0">**does not have a tumor**</span> with a probability of **{1-prob}**
            ''',unsafe_allow_html=True)
if __name__ == '__main__':
    main()
```



CSE3505 J Component Final Review

MRI Brain Tumor Detection

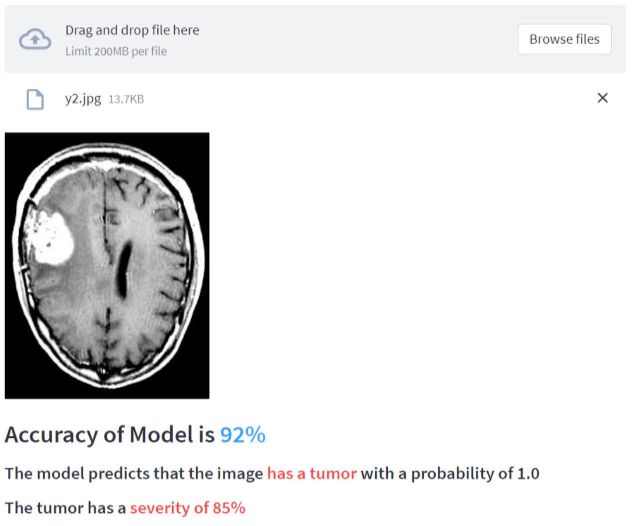
Presented by

- Harsh Deshwal 20BPS1145
- Siddharth Suresh 20BPS1042
- Kanishka Ghosh 20BPS1125

Upload the scan to test

Pick an image to test

Fig 12. Interface for Streamlit

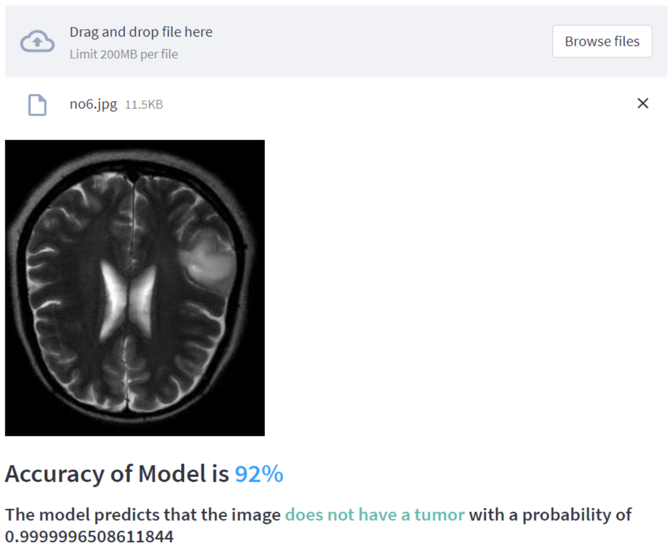


Accuracy of Model is 92%

The model predicts that the image has a tumor with a probability of 1.0

The tumor has a severity of 85%

Fig 13. Testing the Tumor MRI image as input



Accuracy of Model is 92%

The model predicts that the image does not have a tumor with a probability of 0.9999996508611844

Fig 14. Testing the Normal image as input

REFERENCES

[1]L.Guo,L.Zhao,Y.Wu,Y.Li,G.Xu, and Q.Yan,"Tumor Detection in MR images using oneclass immune feature weighted SVMs," IEEE Transactions on Magnetics, vol. 47, no. 10, pp. 3849–3852,2011.

[2]R.Kumari,"SVMclassificationanapproachondetectingabnormalityinbrainMRIimages,"InternationalJournalofEngineeringResearchandApplications,vol.3,pp.1686–1690,2013.

[3]DICOM Samples Image Sets, http://www.osirix-viewer.com/.

[4]"Brainweb: Simulated Brain Database" http://brainweb.bic.mni.mcgill.ca/cgi/brainweb1.

[5]Obtainable Online: www.cancer.ca/~/media/CCE 10/08/2015.

[6] J. C. Buckner, P. D. Brown, B. P. O'Neill, F. B. Meyer , C. J. Wetmore,J. H Uhm, "Central nervous system tumors." In Mayo Clinic Proceedings,Vol. 82, No. 10, pp. 1271- 1286, October 2007.

[7] Deepa , Singh Akansha. (2016). - Review of Brain Tumor Detection from tomography. International Conference on Computing for Sustainable Global Development (INDIACom)

[8] R. A. Novellines, M. D. - Squire's fundamentals of radiology; Six Edition; UPR, 2004.

[9] preston, D. c. (2006). Magnetic Resonance Imaging (MRI) of the Brain and Spine from Basics. casemed.case.edu .

[10] Hendrik RE. (2005) Glossary of MR Terms from American College of Radiology . .

[11].A. Demirhan, M. Toru, and I. Guler, “Segmentation of tumor and edema along with healthy tissues of brain using wavelets and neural networks,” IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 4, pp. 1451–1458, 2015.

[12]Amin, J., Sharif, M., Yasmin, M., Fernandes, S. L. (2018). Big data analysis for brain tumor detection: Deep convolutional neural networks. Future Generation Computer Systems, 87, 290-297.

[13] URL-1, https://www.acibadem.com.tr/ilgi-alani/beyin-tumorleri/, Last Accessed Date: 27.01.2021

[14] Çinar, A., Yildirim, M. (2020). Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture. Medical hypotheses, 139, 109684.

[15] URL-1, https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection, Last Accessed Date: 27.01.2021

[16] Dong, H., Yang, G., Liu, F., Mo, Y., Guo, Y. (2017). Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In annual conference on medical image understanding and analysis (pp. 506-517). Springer, Cham.

[17] Amin, J., Sharif, M., Yasmin, M., Fernandes, S. L. (2017). A distinctive approach in brain tumor detection and classification using MRI. Pattern Recognition Letters.

[18] Wu, M. N., Lin, C. C., Chang, C. C. (2007). Brain tumor detection using color-based k-means clustering segmentation. In Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007) (Vol. 2, pp. 245-250). IEEE.

[19] Chandra, G. R., Rao, K. R. H. (2016). Tumor detection in brain using genetic algorithm. Procedia Computer Science, 79, 449-457.

[20] Şeker, A., Diri, B., Balık, H. H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. Gazi Mühendislik Bilimleri Dergisi (GMBD), 3(3), 47-64.

[21] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[22] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[23] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[24] Yildirim, M., Cinar, A. (2020). A deep learning based hybrid approach for COVID-19 disease detections. Traitement du Signal, 37(3), 461-468.

[25] Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., Acharya, U. R. (2020). Automated detection of COVID19 cases using deep neural networks with X-ray images. Computers in biology and medicine, 121, 103792.

[26] Townsend, J. T. (1971). Theoretical analysis of an alphabetic confusion matrix. Perception & Psychophysics, 9(1), 40-50.