Brain Tumor Classification Using Deep Learning(CNN)

A PROJECT REPORT

Submitted by

Dhilipkumar M
18MIS1112

in partial fulfillment for the award of the degree of

Master of Technology

in

Software Engineering (5 Year Integrated Programme)

School of Computer Science and Engineering
Vellore Institute of Technology
Vandalur - Kelambakkam Road, Chennai - 600 127
November - 2022

Abstract

The main regulator of the humanoid architecture is the human brain. A mind cancer results from the unusual growth and division of cells in the cerebrum, and cerebrum sickness is brought on by the continued development of mind growths. PC Vision has a significant role in the area of human well-being, which lessens the human judgement that produces precise results. X-ray, X-Beam, and CT scan examinations.The imaging method used to determine whether a person has a brain growth infection is attractive reverberation imaging (X-ray). In order to determine where a growth may one day become hazardous, early detection of cerebrum growths is a crucial task in therapeutic practise.A useful and effective method for picture characterization is deep learning. The use of profound learning is widespread, particularly in the disciplines of clinical imaging,in light of the fact that its application doesn't need the dependability of a specialist in the connected field, however requires how much information and various information to deliver great grouping results.For the purposes of this evaluation, we pre-handled the commotions that were present in an MR image using the reciprocal channel (BF). For a reliable identification of the cancer district, it followed by thresholding techniques and convolutional brain organization (CNN) into two sequences. Datasets for preparation, testing, and approval are used.In this study, we examined two model CNNs to determine the most effective model CNN for ordering malignancies in Mind X-ray Picture and toward the end, we have prepared CNN and gotten a forecast exactness of up to 91%.

List of Tables

List of Figures

Introduction

medical imaging is a technology and a procedure used to provide visual representations of a bodys interior structures for use in clinical analysis medical intervention and to show how certain organs or tissues work the two main purposes of medical imaging are to identify and cure disease as well as to disclose internal structures that are covered by the skin and bones medical imaging also creates a database of typical anatomy and physiology in order to make it simpler to spot anomalies.Computer-based picture manipulation is referred to as "medical imaging processing." This processing includes several methods and strategies, including as image capture, archiving, display, and communication. This process seeks to regulate accuracy and diagnose disorders. By creating a database on the typical structure and operation of the organs, this approach makes it simpler to spot irregularities. Image processing is the process of altering digital images using a computer. Numerous benefits, such as adaptability, flexibility, data storage, and communication, are provided by this technology. With the invention of many picture scaling techniques, images may be maintained effectively.Numerous sets of criteria must be followed simultaneously in the photos while using this approach. Multiple dimensions can be handled for both 2D and 3D images. Magnetic resonance imaging (MRI), produces no ionising radiation, is one of the most frequently utilised medical imaging methods for brain tumours. Many research have investigated the potential of identifying brain tumours by combining Fuzzy C-Means and SVM algorithms.. This paper proposes the CNN technique as a solution to the data complexity problem. The input data may be processed using cnn without the geographical information being lost data in two dimensions is analysed by a machine learning technique called cnn cnn employs two techniques feedforward classification and backpropagation learning.

1.1 Background:

Brain Tumor is portrayed as amazing advancement of tissues in the frontal cortex. Nowadays the absence of capacity of cancers is creating quick. In 2016, a conventional 23,800 grown-ups in the US was associated with the hazardous malignant growths of cerebrum likewise as spinal code. Evaluation of frontal cortex growths is somewhat sketchy as the moved shape, size, cancer area and the closeness and nearness of growth at the top of the need list. It's challenging to see frontal cortex growths in beginning stage considering the way that the specific assessment of cancer can't be found. Incidentally, when the frontal cortex cancer is seen at without a doubt the beginning stage, the best drugs should be conceivable and it very well may be reparable. As of now, visual depiction of inside the body is arranged using clinical imaging method for clinical examination and clinical investigates. X-point of support is awesome and exhaustively used system for mind cancer area. Current finding procedures are performed using the ordinary frameworks reliant upon human experience and this fosters the opportunity of sham zone while seeing frontal cortex cancers. Present device and structures to destroy cancers and their lead have gotten effectively powerful. Picture getting ready strategy can be used to see frontal cortex growths. Picture getting ready strategies changes over pictures into forefront what's more, do structure on them, to offer hints of progress and updated pictures. This assessment will think instructions to see frontal cortex cancers using picture managing techniques

1.2 Statement:

healthcare industry is very distinct from other industries. people demand the best treatment and services available in this high-priority area regardless of cost deep learning is already giving unique solutions with excellent accuracy for medical imaging and is a key technology for prospective applications in the health business as a result of its success in other practical applications all body activities are regulated by the brain an organ. It can be difficult to recognise an automated brain tumour in an MRI because of the intricacy of size and location fluctuations. the automatic identification and categorization of brain tumours is the main topic of our study MRI or CT scans are routinely used to investigate the anatomy of the brain. This study aims to enhance brain MR imaging's tumour identification and accuracy. The main objective of brain tumour detection is to support clinical diagnosis. The objective is to develop a method for reliably identifying tumours in MR brain images by merging several approaches into an algorithm that guarantees the existence of a tumour. Techniques for delineating tumours, such as edge detection, are utilised, along with filtering, erosion, dilation, and thresholding. In this work, thresholding procedures, morphological analysis, and statistical analysis are recommended for MRI image analysis in order to detect tumours from brain MRI images.Using a feed-forward backprop neural network, the tumours in the picture will be categorised based on their performance. This project's objective is to eliminate tumours from MR brain pictures and represent them in a form that is clear to everyone. The purpose of this initiative is to provide users, especially healthcare professionals who are treating patients, with some important information in a more user-friendly manner. The objective of this study is to develop an algorithm that will extract a tumour image from an MR brain image.

the final picture will be able to provide information about the tumour including its size shape and location, as well as details about the tumour that may be useful in certain situations. This will provide the staff a stronger basis for choosing the treatment method. Finally, we identify the presence of a tumour in an MR brain picture using a convolution neural network. Accuracy will increase as a result of this method while iterations will be fewer.

1.3 Motivation:

A brain tumour can be identified by abnormal cell proliferation inside the brain or central spinal canal. Malignant tumours must be identified and treated right away since some of them are People may be afflicted with brain tumours without being aware of it since both the specific set of symptoms and the true aetiology of brain tumours are unclear. Malignant primary brain tumours are those that contain cancer cells, while benign primary brain tumours do not. Cells started to proliferate and expand erratically, and a brain tumour developed.

When observed with diagnostic medical imaging tools, it seems to be a solid mass. Primary brain tumours and metastatic brain tumours are the two different forms of brain tumours. Primary brain tumours develop in the brain and often stay there, in contrast to metastatic brain tumours, which start elsewhere in the body and move to the brain. The location, size, and kind of the tumour all affect whether or not brain tumour symptoms are present. It occurs when the tumour applies pressure on the surrounding cells before releasing it. Additionally, it happens when the tumour impedes the flow of fluid throughout the brain. Headaches, nausea, and vomiting are among the most typical symptoms, along with balance and walking issues. Brain tumours can be found using diagnostic imaging techniques like CT scanning and MRI.

Both modalities are advantageous in the detecting and depending on the setting and the intended plan of the test. The most popular method for seeing and locating brain tumours is magnetic resonance imaging (MRI). In addition to alternative approaches, the traditional method for classifying and finding tumour cells in ct and mr images continues to be validated by human evaluation. Because they are non-destructive and non-ionizing, mri images are typically used.High-definition pictures produced by MR imaging are frequently utilised to find brain tumours. T1-weighted, T2-weighted, and flair pictures are examples of MRI imaging methods. Pre-processing of image and segmentation of image and enhancement of image data and feature extraction, and classifiers are a few of the picture processing techniques. Since MRI images are simple to examine and accurately locate calcification and foreign masses, we prefer to utilise them in this research.

1.4 Challenge:

A MRI images of the brain is one of the most frequent ways in medical research for detecting a brain tumour and its progression. As a result, the process of scanning brain images from the human brain's internal structure provides information regarding the growth of brain tumours. Manual detection of brain tumours from MRI is a difficult work in the medical research sector because tumours produce significant changes in the internal and external structure of the brain. Brain tumour identification is significantly hampered by the variations in tumour size, shape, and location. this studys main goal is to give researchers a thorough literature assessment on brain tumour detection using magnetic resonance imaging It is proposed to study the detection of brain tumours from MRI images using hybrid computerised techniques for this aim. As a result, the performance and analysis of brain tumour growth are detailed in order to generalise symptoms and direct diagnosis toward a treatment plan. Several methodologies for the MRI segmentation process are reviewed in previous works, and brain tumour detection can be concluded.

Planning and Requirements Specification

2.1 System Planning:

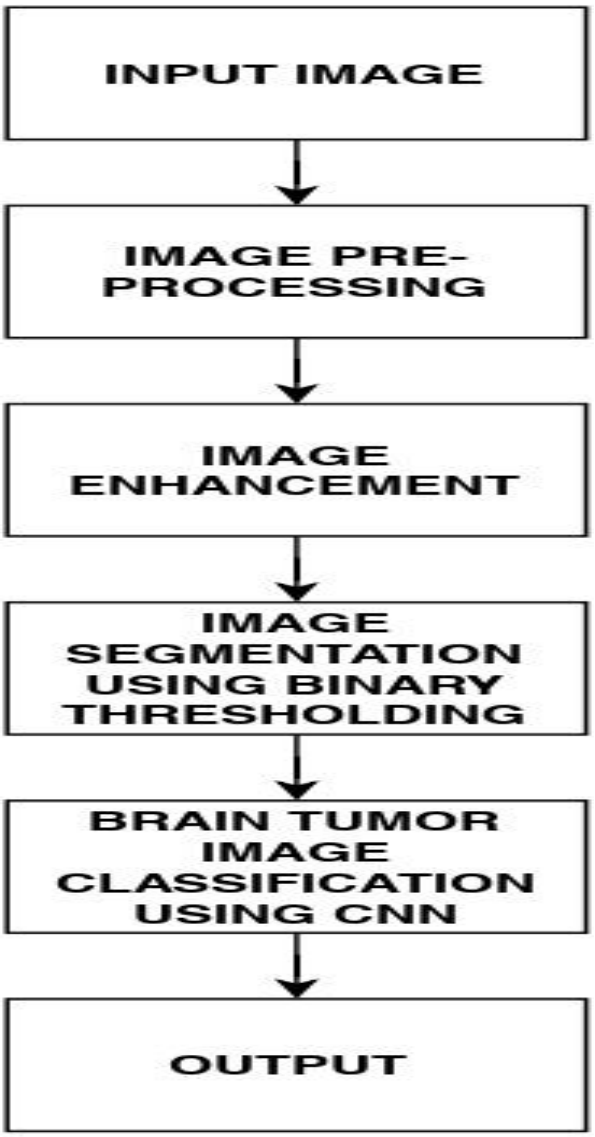


Figure 1: System Planning

The architecture for the system we will build is provided by this. The approach involves six steps: picture extraction from data acquisition image preprocessing image enhancement image segmentation using binary thresholds and convolutional neural network classification of brain tumors. After all of the aforementioned steps have been finished, the output is then checked. Each module stands out in its own unique manner. Every action has importance. Additionally, data gathering for training and testing is included in this approach. The data set utilised included roughly 256 images that were pulled from Kaggle were included in the data set that was used to train and test the algorithm. The supplied picture is first pre-processed using the noise filter a series of eroding The image is then subjected to morphological procedures after being segmented using binary thresholding. Finally, a Convolutional Neural Network is used to classify the images and determine whether the tumour is present or not.

MODULE1:IMAGE PREPROCESSING AND IMAGE ENHANCEMENT

IMAGE PREPROCESSING:

Kaggle was used to gather the dataset of brain MRI images. About 256 MRI pictures, comprising normal, benign, and malignant images, are included in the MRI dataset. These MRI pictures provide the input for the initial phase. Pre-processing is a crucial initial step in raising a brain MRI image's quality. Scaling images and removing impulsive noises are essential pre-processing steps. The brain MRI picture is first converted to its corresponding gray-scale image. The adaptive bilateral filtering method reduces unwanted noise by eliminating distorted sounds from the brain picture. This raises classification accuracy while also improving diagnosis.

IMAGE ACQUISITION FROM DATASET:

Image acquisition in image processing involves retrieving an image from a dataset and processing. Due to the fact that no processing is possible without a picture, it is the first step in the workflow sequence. The photographed image is entirely raw. In this stage, the file path from the local device is used to process the picture.

CONVERT THE IMAGE FROM ONE COLOR SPACE TO ANOTHER COLOR:

CV has above more than one fifty colour-space conversion techniques. And we use the function cv2.cvtColor(image, cv2.COLOR BGR2GRAY) to convert colours.

Use the cv2. cvtColor() method to change the colour space of an image. More than 150 color-space conversion process are available in Open-CV.

cv2.color bgr2gray

We uses the code COLOR BGR2GRAY to convert our original image from the BGR colour system to grayscale. To display images, we simply need to use the cv2 module's imshow method.

cv2.GaussianBlur

The Gaussian filter is a lower-pass filter remove high-frequency components. The Gaussianblur() method of the imgproc class can be used to execute this process on an image. This method's syntax is as follows.

FILTERS:

Filters are like used to employed in image processing to reduce the very high frequencies in the images or picture.

Median filters: It is a non-linear filtering method for reducing picture noise. It is done by numerically ordering all of the window's pixel values, after which the pixel under consideration is swapped out for the median value. through the on and off of pixels by white and dark patches this filter eliminates salt and pepper noise and speckle noise

Bilateral filters: it also serves as a noise-reduction non-linear smoothing filter for images. Every pixel's force is changed to a weighted normal of power values from nearby pixels.The Gaussian distribution provides the foundation for this weight.Using a nonlinear collection of adjacent picture pixels, bilateral filtering blurs images while keeping an eye on their boundaries. This filtering method is straightforward, focused, and local. It chooses near to vales to distant characteristics in both reach and space and coordinates a grey level based on their resemblance and symmetrical closeness.

IMAGE ENHANCEMENT:

The input pixel values for the district control local activities. Spatial and transform domain approaches are the two categories of picture improving methods. The transform technique acts on Fourier and subsequently on the spatial method, whereas the spatial techniques operate directly at the pixel level.

A segmentation method called edge detection uses border identification to isolate tightly related items or areas. This method allows for the identification of the items' discontinuity. This method is mostly employed in image analysis to locate regions of the picture with a wide range of intensity.

MODULE 2: IMAGE SEGMENTATION USING BINARYTHRESHOLD

The process of segmenting an image allows it to be broken up into several fragments. This division's major purpose is to retain image quality while making it simpler to look at and understand. Tracing the edges of things in pictures is another usage for this method. According to their intensity and characteristics, this approach recognises pixels. These areas take on characteristics like intensity and similarity while still representing the entire original image. In healthcare contexts, the photo or image segmentation technique which is used to create body shape or forms. Segmentation is a strategy used in. the measurement of tissue volumes, anatomical and functional studies, the visualisation of virtual reality, the analysis of anomalies, and the definition and detection of objects. The aberrant component of a picture may be located or identified using segmentation techniques, which is helpful for analysing the extracted image's size, volume, location, texture, and form.

ability to a more precisely identify broken sections using MRi image have the segmentation with the help of keeping the threshold records. It was a popular assumption that objects placed in close proximity would have similar properties and qualities.

THRESHOLDING:

The basic technique for segmenting images is thresholding. The method or process of converting a greyscale image to a (binary image) it's a simply while and black is a non-linear process that involves giving each pixel a level that corresponds to its position in relation to the threshold value. Using Open CV's cv2.threshold() method,

we threshold the picture before erasing any minor noise with erosions and dilations.

The cv2.THRESH BINARY INV technique is employed, and it specifies that output values be set for pixels with values p smaller than T. (the third argument). The second value, V, is the value of the second argument, and the first value, T, is the threshold value, returned by the cv2. threshold function.

cv2.erode(thresh, None, iterations=2)

To perform erosion on cv2, use the erode() on the images.

The edges of foreground objects are eroded by erosion in its most basic form, which is similar to soil erosions it is often applied to binary image

cv2.dilate(thresh, None, iterations=2)

Dilation is a technique for expanding an image. It increases the number of pixels around the edges of objects in an image. It is under the control of the structuring element.

1.src -variety of sources (single-channel, 8-bit or 32-bit floating point). This is an original image picture must be grayscale.

2. **Thresh'** - the pixel values' classification is based on the threshold value..

3.type - type of thresholding

4.Locate the Contours

Use the findContours() method to find contours in an image. Use the findContours() function of the cv2 package to locate all boundary lines. The findContours() function of the cv2 library is used to locate each x, y boundary point of an object in an image. Use the import command to import the cv2 library before using it. All continuous points (along the boundary) that are the same colour or intensity are connected by a curve to form contours.

function imutils.grab contours()

A collection of OpenCV convenience methods for Python 2.7 and Python 3 that simplify common image processing tasks including contour sorting, edge detection, skeletonization, Matplotlib image presentation, and more..Then we find the images' extreme points and crop a new image from the extreme points.In addition, cropping the imageThere is no specific cropping function in OpenCV; instead, NumPy array slicing is used. Every image read in is stored in a 2D array (for each colour channel). Simply enter the height and width (in pixels) of the cropped area. And it's finished!

MODULE 3: IMAGE CLASSIFICATION OF BRAIN TUMOR USING CNN

The most effective methods for classifying pictures, classification and labelling include all forms of medical images all classification techniques are predicated on the premise that a picture will include one or more features each of which will fall under a different class. Convolutional Neural Network (CNN), whose strong structure aids in recognising even the small information, will be employ as an automated and the trustworthy of the classification method.A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning system that can analyse an image as input, prioritise distinct characteristics and objects within the picture, and distinguish between them.

A convolutional network requires substantially less pre-processing than conventional classification techniques. In contrast to hand-engineered filters used in basic approaches, ConvNet can be learning the filter and properties with the enough practise.

by employing the appropriate filters a convnet may successfully capture the spatial and temporal correlations in a picture because there are fewer factors to consider and weights may be reused the architecture produces a better fit to the image dataset in other words the network may be trained to understand the intricacy of the image more clearly the task of the convnet is to compress the images into a more easily analysed format without compromising crucial components needed to provide an accurate forecast. We must import Keras and the additional packages we'll require to create the CNN for this stage. import the subsequent files:

- The neural network is initialised using sequential.
- The convolutional network that interacts with the pictures is created using Convolution2D.
- The pooling layers are added using the MaxPooling2D layer.
- The function called "flatten" divides the pooled feature map into a single column and passes it on to fully linked layers.
- The fully connected layers is added to the convolution neural network via dense.

SEQUENTIAL:

• An object of the Sequential class is created and used to initialise the neural network.

• Sequential classifier ()

4.3.1 CONVOLUTION:

the convolution layer is added by using the add function with the classifier object and convolution2d with parameters. the number of feature detectors we wish to build is represented by the first argument feature detectors.

For CNNs, we use the 256 feature detectors. The input shape images picture , or shape of the input picture, is the following parameter. During pre-processing, the photos will be transformed into this shape. A 2D array will be created if the picture is in black and white, and a 3D array will be created if it is in colour.

We'll presume that we are dealing with coloured photos in this instance. For a coloured picture, the number of channels is 3, and the input shape tuple includes the 2d arrays dimensions for each channel. Lower dimensions should be used if a GPU is not being used to speed up processing. The activation function is the last argument. It's difficult to categorise photos linearly. So, in order to prevent computing with negative pixel values, we employ the rectifier function. We accomplish the non-linearity in this way.

Planning and Requirements Specification

POOLING:

the clustering layer is responsible for minimizing the spatial size of the built-in feature which reduces the amount of cpu power required to process the data through size reduction extracting invariant dominant features rotation and position is also useful to properly train the model two types of aggregates max composite and max composite mean which returns the maximum value of the area covered by the kernel of the image the value the average of all the image multiplier area data is returned by averaging otherwise we usually use max aggregation now we reduce the size of the feature map to maximize the sum our standard bucket size is 2x2 so we can reduce the feature map without losing any important image data classifiers

- ***classifier.add***

FLATTENING:

- All of the pooling feature mapping are now taken and combined into a one vector for use in the subsequent layers.
- Using the Flatten function, and all the feature maps are collapsed into a single, lengthy columns.

• Flatten () classifier add

FULLY CONNECTION:

The vector we collected before will now be used as the input for the neural network by utilising Keras', Dense function. The numerous of the nodes in the hidden layers is the first parameter, output.

• Through experimentation, you can identify the best number. You will require more processing power to fit the model the more dimensions there are. Selecting nodes in powers of two is a typical approach.

dense (output = 64); classifier.add

• The output layer is the next layer that has to be added. Since we anticipate a binary result in this situation, we will employ the sigmoid activation function. We would employ the SoftMax function if we anticipated more than two results.

•

Since we just anticipate the classes' expected probability, the result in this case is 1.

• classifier.add (Dense (output = 1, activation = "sigmoid"))

2.2 Requirements

2.2.1user requirements

System requirements outline what the system must do without describing how it must be accomplished. This document's requirements are comprehensive and uniform. There are two categories of users for this programme:
Patients and doctors.

1. The softwarea allows the patient determine tumor's size. The layperson can easily comprehend the tumor's size and location.
2. Doctors are employing to visualise and extract the tumour from brain MRI scan images.

2.1.2 Functional Requirements

- choosing the brain's MRI scan images.
- Extraction of the scan pictures' tumour region exclusively.
- Identifying the tumor's border.
- constructing a graphical user interface for the software.

2.2.3 Non functional requirements

• Availability: All PCs with Python installed may access the programme for extracting brain tumours from MRI scan pictures. • Reliability – This programme makes every effort to ensure suitable material, but it takes no responsibility for manipulations from other sources.

• The suggested software's CPU time ranges from 4 seconds to 6 seconds, and its PSNR value ranges from 25dB to 26dB.

System Designs

Algorithm explanation

- One technique that is frequently utilised in computer vision applications is the convnets., sometimes referred to as CNN or convnets.

- For tasks like processing pixel data and image identification, a convnets is a specific form of deep learning network design.

- Although deeplearning employs a number of network for neural designs, CNNs continue to be the favoured network architecture for object recognition and classification.They are therefore excellent candidates for computer vision (CV) jobs.

CNN working methodology :

- First, choose a dataset.
- Prepare a training dataset in step two.
- Create Training Data in step 3
- Rearrange the dataset in step four.
- Step 5: Labeling and Feature Selection
- X is normalised in step 6 and labels are converted into categorical data

- Separate X and Y for CNN in step 7.

- Step 8: Specify, compile, and train the CNN Model

- Step 9: Score and model accuracy

Hardware requirements

- Intel core i5 or higher processor required.
- Quad-core, 64-bit, 2.5 GHz minimum for each core
- 4.0gb or more of RAM
- 10.0gb or more of free disk or space on the hard drive.
- Display: Two monitors with a resolution of 1024 x 768 or higher or above.
- The operating system windows

Softwares requirement

- ❖ Windows or mac:
- ❖ Google colab
- ❖ Python
- ❖ PIP and NumPy
- ❖ Tensorflow
- ❖ keras

Python:

Python is an interpreted programming language that was created by Guido Van Rossum and originally made that available in 1991. Python's design philosophy places a strong focus on the readability of its code as much , as shown by the significant usage of whitespace. Its language elements and object-oriented methodology are designed to help programmers write the understandable, logical code for en both small- and large-scale projects. Python is dynamically typed and garbage collected. It is compatible with a number of object-oriented, and functional programming.

PIP:

Python software packages are installed and managed using this package management tool.

NumPy:

An array processing Python library is called NumPy., it. It hava also consist a number of characteristics are available in the numpy , including the following:

- Strong N-dimensional object array in class
- high-tech features
- Integration tools for the program
- The ability to use the Fourier transform, linear algebra, and random numbers is all beneficial.

Pandas:

The most popular data analysis package for Python is called Pandas. With the back-end of the source of code written completely in an flexible of C or Python, it provides performance that is highly optimised. Data analysis is possible with pandas.

Data frames

Series

Anaconda:

The Python and R programming languages are combined in Anaconda, a free and open-source distribution that makes it easier to manage and distribute packages for scientific computing.Conda, the package management system, manages package versions. The data-science packages from Anaconda are accessible on Windows, Linux, and macOS. The conda package manager and virtual environment manager are both included in the anaconda distribution, along with 1,500 PyPI items.

Jupyter Notebook:

The conda package manager and virtual environment manager are both included in the anaconda distribution, along with 1,500 PyPI items.

Implementation of System

Implementation:

1. Method

1.1. Dataset

The dataset which is used in the study was Brain MRI Images or data for Brain Tumor Detection from Kaggle website . 253 pictures total, split between 155 brain scans with tumours and 98 brain images without them, make up the collection. Figure 2 showcases images data from the dataset .

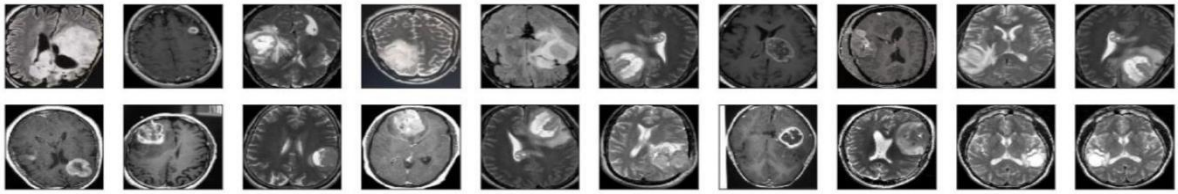


Figure ii. samples of the datasets images

1.2. Proposed Method

In this study, CNN is used to automatically detect brain cancers. In order to discriminate between tissues that contain tumours and those that do not, this study employs input photos from the raw data that are labelled (yes/no) as input images. 2065 example photos, 1085 of which had tumours and 980 of which did not, were used to train CNN. As a result, the strategy suggested in this study is represented in Figure 3.

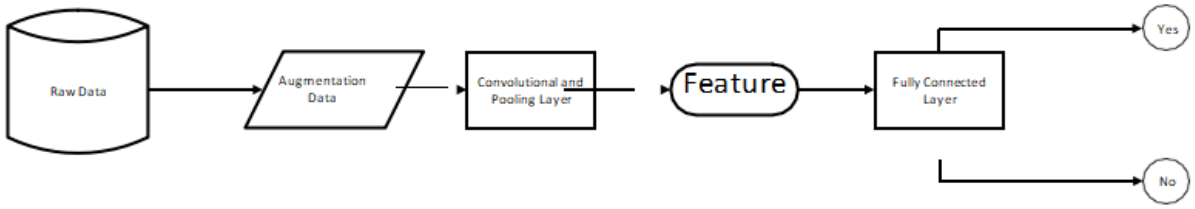


Figure iii. step propose

Yes Represent **tumors**
No represent **non-tumorous**

1.3. Data Augmentation

The dataset does not include enough data to be used as CNN training datas. In orders to address the imbalance of problems, the augmentation method is applied. An algorithm known as augmentation can use statistical data to build an integrated model. This programme is capable of creating several two-dimensional images in a range of poses and dimensions. CNN segmentation accuracy can be increased by using augmentation to create picture variants. In this study, each image that has a tumour is divided into six images, and each image that doesn't contain a tumour is divided into nine images. 1085 samples with tumours (53%) and 980 samples without tumours (47%), totaling 2065 pictures, make up the dataset after data augmentation.

1.4. Image Pre-processing

Because images come in a variety of intensities, contrasts, and sizes, pre-processing helps to provide smooth training .Wrapping and resizing are the initial pre-processes that will be done to the supplied image. The input image is compared to the edge of the main subject of the image while wrapping. So that the item in the image is retained during cropping, the maximum edge of the image is computed depending on its edge.

Implementation of System

Given that the dataset contains a variety of picture sizes, and then cropping, resize the image to fit (240=width, 240=height, 3=channel). Use normalisation to scale pixel values to the 0–1 range.

3.5 CNN model

Convolution, pooling, flattening, dropout, and thick layers are just a few of the many layers that make up the CNN model that was employed in this study. This work contains a rule-based activation function in the more to the layers used in the CNN approach. For this study's comparison material, two CNN models were developed. Tables 1 and 2 show how the CNN model is laid up. A number-shaped image with an interwoven first convolution and a 240x240 pixel resolution. According to the channel of the image data and filters, 3x3 kernels with a thickness of 3 are used. The model will take care of the activation and data pooling tasks after obtaining the operation's outcomes. The size of the feature map is decreased via the pooling layer technique. A feature map is created by the convolution process, which is subsequently utilised in other convolution operations. In order to do a totally fully-connected layer approach to achieve picture classification, the final step is to construct a vectorized fallen feature map.

Implementation of System

CNN MODEL

Table 1. First CNN model.

Layer (type)	Output Shape	Param #
CONV2D (CONV2D)	(None, 240, 240, 32)	896
MAX_POOLING2D (MAXPOOLING2D)	(None, 60, 60, 32)	0
MAX_POOLING2D_1 (MAXPOOLING2D)	(None, 30, 30, 32)	0
MAX_POOLING2D_2 (MAXPOOLING2D)	(None, 15, 15, 32)	0
FLATTEN (FLATTEN)	(None, 7200)	0
DENSE (DENSE)	(None, 256)	1843456
DENSE_1 (DENSE)	(None, 1)	257
TOTAL PARAMS: 1,844,609 TRAINABLE PARAMS: 1,844,609 NON-TRAINABLE PARAMS: 0		

Implementation of System

SECOND CNN MODEL

Table 2. Second CNN model.

Layer (type)	Output Shape	Param #
CONV0 (CONV2D)	(None, 240, 240, 32)	896
MAX_POOLING2D (MAXPOOLING2D)	(None, 60, 60, 32)	0
DROPOUT (DROPOUT)	(None, 60, 60, 32)	0
CONV2D_1 (CONV2D)	(None, 60, 60, 32)	9248
MAX_POOLING2D_1 (MAXPOOLING2D)	(None, 15, 15, 32)	0
DROPOUT_1 (DROPOUT)	(None, 15, 15, 32)	0
FLATTEN (FLATTEN)	(None, 7200)	0
DENSE (DENSE)	(None, 256)	1843456
DROPOUT_3 (DROPOUT)	(None, 256)	0
DENSE_1 (DENSE)	(None, 1)	257
TOTAL PARAMS: 1,863,105 TRAINABLE PARAMS: 1,863,105 NON-TRAINABLE PARAMS: 0		

IMPORTING THE NECESSARY MODULES

Import Necessary Modules

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import cv2
import imutils
import numpy as np
import matplotlib.pyplot as plt
import time
from os import listdir

%matplotlib inline
```

Tensorflow

The recognised photographs are saved in a specific folder and have a special capacity to identify pictures thanks to TensorFlow. It will be straightforward to apply this logic for security reasons using very comparable photos.. The relevant images are contained in the dataset image and must be loaded.

Keras

For building neural networks, Google created the high-level deep learning API called Keras. It was made in Python and is employed to make the neural network implementation process simpler. Additionally, several backend neural network calculations are made possible..

Conv2D

The quantity of filters that convolutional layers will learn from is specified by the Conv2D parameter. It also regulates the quantity of convolution output filters, and it is an integer variable.

ZeroPaddling2D

This layer can add rows and columns of zeros to an image tensor's top, bottom, left, and right sides. padding: Int, tuple of two ints, or tuple of two tuples of two ints. The same symmetric padding is given to height and width if int.

BatchNormalization

The normalisation process are called as batch Normalization is carried out inside the layers of a neural network layer of data rather than on the raw data input, it is done in mini-batches. Learning is simplified since instruction is accelerated and quicker learning rates are used.

Activation

Simply put, the activation argument to the Conv2D class carried you to specify a string that contains the name of the function activation you wish to employ after convolution. Nothing is activated if nothing is given.

Maxpooling2D

The maximum pooling procedure is used for 2D spatial data. Obtains the greatest value of channel over an input window to downsample the input along its spatial dimensions (height and breadth) (of size determined by pool size). The window's dimensions are changed in strides.

Flatten

Multi-dimensional arrays can be flattened into 1-dimensional arrays using this method. When feeding the 1-D array data to the classification model in deep learning, it is frequently employed.

Implementation of System

Dense

Based on inputs from the convolutional layer, pictures are identified using the dense layer. The neurons in one of the layer of the neural network count the weighted median of their input, which is then processed by a non-linear function called a "activation function."

Sklearn

scikit-image is a Python image processing package that interacts with NumPy arrays, which are a collection of image processing techniques.

Cv2

The image processing and computer vision software OpenCV is excellent. It is an open-source library that may be used for several applications, including object tracking, face identification, and landmark detection.

Numpy

NumPy was used to convert the image to a grayscale image. We can do this by taking the weighted mean of the image's RGB values.

Matplotlib

The matplotlib function imshow() generates a picture from a two-dimensional numpy array. Each array element will be represented by one square in the image.

Matplotlib

The colour of each square is determined by the value of the relevant array member and the colour map provided by imshow().

Time

You can manipulate time in Python by using the time module. It offers capabilities like getting the current time, pausing the operation of the software, and others. We must thus import this module before we can use it.

from os import listdir

The listdir() method is supplied by the os module, and we can use it to produce a list of the names of all the files in the specified directory. If we wish to print a list of files in the current working directory, we can use the listdir() method (where the programme is present).

%Matplotlib inline

%matplotlib inline enables "inline plotting," which causes plot visuals to show in your notebook. This has significant ramifications for interactivity: when using inline plotting, directives in cells below the cell that outputs a plot have no effect on the plot.

Data Preparation & Preprocessing

In order to crop the part that contains only the brain of the image, The extreme top, bottom, left, and right locations of the brain were located using a cropping approach.

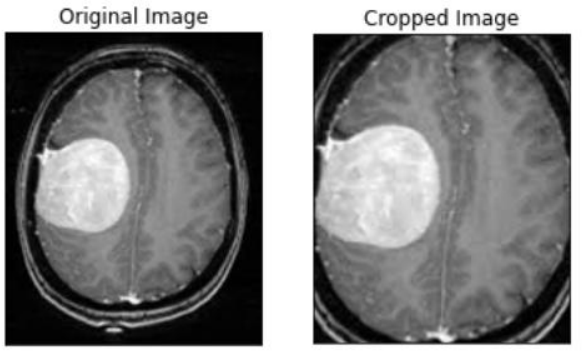
```
def crop_brain_contour(image, plot=False):

    #import imutils
    #import cv2
    #from matplotlib import pyplot as plt

    # Convert the image to grayscale, and blur it slightly
    gray = cv2.cvtColor(image, 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])
```

```
new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

if plot:
    plt.figure()

    plt.subplot(1, 2, 1)
    plt.imshow(image)

    plt.tick_params(axis='both', which='both',
                    top=False, bottom=False, left=False, right=False,
                    labelbottom=False, labeltop=False, labelleft=False, labelright=False)

    plt.title('Original Image')

    plt.subplot(1, 2, 2)
    plt.imshow(new_image)

    plt.tick_params(axis='both', which='both',
                    top=False, bottom=False, left=False, right=False,
                    labelbottom=False, labeltop=False, labelleft=False, labelright=False)

    plt.title('Cropped Image')

    plt.show()

return new_image
```

- Grab one image from the dataset and use this cropping method to examine the results so you can see what it does better
- Grab a picture from the collection and use this cropping function to view the outcome in order to better understand what it is doing.

Output:

```
ex_img = cv2.imread('/content/drive/MyDrive/Brain-Tumor-Detection-master/yes/Y1.jpg')
ex_new_img = crop_brain_contour(ex_img, True)
```



Load up the data:

The following code accepts two arguments: the image size and a list of directory paths for the folders "yes" and "no," which contain the image data. For each picture in both directories, it performs the following actions:

Read the image, first.

2. Crop the area of the picture that only shows the brain.

3. Resize the image because the dataset contains images of various sizes (i.e., width, height, and number of channels). So, in order to feed it as an input to the neural network, we want all of our photos to be (240, 240, 3).

4. Use normalisation since we want the range of pixel values to be from 0 to 1.

5. Include the picture with X and its to y.

6. Next, shuffle X and Y since the data is ordered and we don't want the beginning half of the arrays to belong to one class and the second part to another.

Return X and Y lastly.

```
def load_data(dir_list, image_size):
    """
    Read images, resize and normalize them.
    Arguments:
        dir_list: list of strings representing file directories.
    Returns:
        X: A numpy array with shape = (#_examples, image_width, image_height, #_channels)
        y: A numpy array with shape = (#_examples, 1)
    """
```

```
    # load all images in a directory
    X = []
    y = []
    image_width, image_height = image_size

    for directory in dir_list:
        for filename in listdir(directory):
            # load the image
            image = cv2.imread(directory + '//' + filename)
            # crop the brain and ignore the unnecessary rest part of the image
            image = crop_brain_contour(image, plot=False)
            # resize image
            image = cv2.resize(image, dsize=(image_width, image_height), interpolation=cv2.INTER_CUBIC)
            # normalize values
            image = image / 255.
            # convert image to numpy array and append it to X
            X.append(image)
            # append a value of 1 to the target array if the image
            # is in the folder named 'yes', otherwise append 0.
            if directory[-3:] == 'yes':
                y.append([1])
            else:
                y.append([0])

    X = np.array(X)
    y = np.array(y)

    # Shuffle the data
    X, y = shuffle(X, y)

    print(f'Number of examples is: {len(X)}')
```

Implementation of System

Load up the data that we augmented earlier in the Data Augmentation notebook.

Note:

the augmented data directory contains not only the new generated images but also the original images.

Here the Data agumentation with output:

```
augmented_path = '/content/drive/MyDrive/Brain-Tumor-Detection-master/augmenteddata/'

# augmented data (yes and no) contains both the original and the new generated examples
augmented_yes = augmented_path + 'yes'
augmented_no = augmented_path + 'no'

IMG_WIDTH, IMG_HEIGHT = (240, 240)

X, y = load_data([augmented_yes, augmented_no], (IMG_WIDTH, IMG_HEIGHT))
```

```
Number of examples is: 2065
X shape is: (2065, 240, 240, 3)
y shape is: (2065, 1)
```

As we see, we have 2065 images. Each images has a shape of **(240, 240, 3)= (image_width, image_height, number_of_channels)**

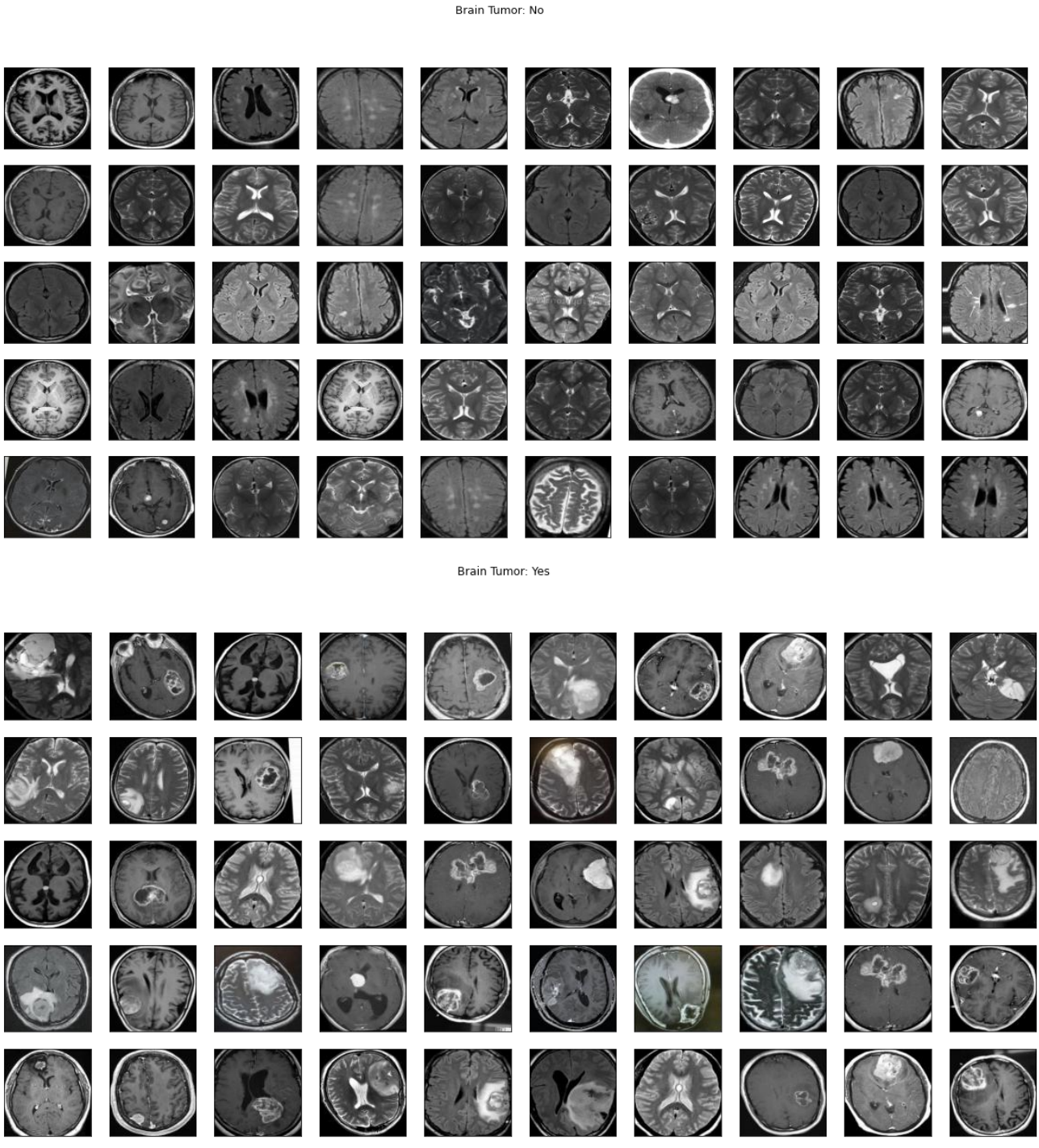
SAMPLE IMAGES PLOTS

```
def plot_sample_images(X, y, n=50):
    """
    Plots n sample images for both values of y (labels).
    Arguments:
        X: A numpy array with shape = (#_examples, image_width, image_height, #_channels)
        y: A numpy array with shape = (#_examples, 1)
    """

    for label in [0,1]:
        # grab the first n images with the corresponding y values equal to label
        images = X[np.argwhere(y == label)]
        n_images = images[:n]

        columns_n = 10
        rows_n = int(n/ columns_n)

        plt.figure(figsize=(20, 10))

        i = 1 # current plot
        for image in n_images:
            plt.subplot(rows_n, columns_n, i)
            plt.imshow(image[0])

            # remove ticks
            plt.tick_params(axis='both', which='both',
                            top=False, bottom=False, left=False, right=False,
                           labelbottom=False, labeltop=False, labelleft=False, labelright=False)

            i += 1
```

```
            label_to_str = lambda label: "Yes" if label == 1 else "No"
            plt.suptitle(f"Brain Tumor: {label_to_str(label)}")
            plt.show()
```

```
[ ]  plot_sample_images(X, y)
```

Output:



division of the data

Dividing X and Y into training, validation (development), and validation sets.

```
def split_data(X, y, test_size=0.2):

    """
    Splits data into training, development and test sets.
    Arguments:
        X: A numpy array with shape = (#_examples, image_width, image_height, #_channels)
        y: A numpy array with shape = (#_examples, 1)
    Returns:
        X_train: A numpy array with shape = (#_train_examples, image_width, image_height, #_channels)
        y_train: A numpy array with shape = (#_train_examples, 1)
        X_val: A numpy array with shape = (#_val_examples, image_width, image_height, #_channels)
        y_val: A numpy array with shape = (#_val_examples, 1)
        X_test: A numpy array with shape = (#_test_examples, image_width, image_height, #_channels)
        y_test: A numpy array with shape = (#_test_examples, 1)
    """

    X_train, X_test_val, y_train, y_test_val = train_test_split(X, y, test_size=test_size)
    X_test, X_val, y_test, y_val = train_test_split(X_test_val, y_test_val, test_size=0.5)

    return X_train, y_train, X_val, y_val, X_test, y_test
```

Let's separate in the manner shown below:

1.70% of the data were used for training.

2.15% of the data are used for validation.

3.testing using 15% of the data.

```
X_train, y_train, X_val, y_val, X_test, y_test = split_data(X, y, test_size=0.3)
```

```
print ("number of training examples = " + str(X_train.shape[0]))
print ("number of development examples = " + str(X_val.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
print ("Y_train shape: " + str(y_train.shape))
print ("X_val (dev) shape: " + str(X_val.shape))
print ("Y_val (dev) shape: " + str(y_val.shape))
print ("X_test shape: " + str(X_test.shape))
print ("Y_test shape: " + str(y_test.shape))
```

Output:

```
number of training examples = 1445
number of development examples = 310
number of test examples = 310
X_train shape: (1445, 240, 240, 3)
Y_train shape: (1445, 1)
X_val (dev) shape: (310, 240, 240, 3)
Y_val (dev) shape: (310, 1)
X_test shape: (310, 240, 240, 3)
Y_test shape: (310, 1)
```

Some helper function:

```
# Nicely formatted time string
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"
```

```
def compute_f1_score(y_true, prob):
    # convert the vector of probabilities to a target vector
    y_pred = np.where(prob > 0.5, 1, 0)

    score = f1_score(y_true, y_pred)

    return score
```

Build the Model:

```
def build_model(input_shape):
    """
    Arugments:
        input_shape: A tuple representing the shape of the input of the model. shape=(image_width, image_height, #_channels)
    Returns:
        model: A Model object.
    """
    # Define the input placeholder as a tensor with shape input_shape.
    X_input = Input(input_shape) # shape=(?, 240, 240, 3)

    # Zero-Padding: pads the border of X_input with zeroes
    X = ZeroPadding2D((2, 2))(X_input) # shape=(?, 244, 244, 3)

    # CONV -> BN -> RELU Block applied to X
    X = Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0')(X)
    X = BatchNormalization(axis = 3, name = 'bn0')(X)
    X = Activation('relu')(X) # shape=(?, 238, 238, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool0')(X) # shape=(?, 59, 59, 32)

    # MAXPOOL
    X = MaxPooling2D((4, 4), name='max_pool1')(X) # shape=(?, 14, 14, 32)



    # FLATTEN X
    X = Flatten()(X) # shape=(?, 6272)
    # FULLYCONNECTED
    X = Dense(1, activation='sigmoid', name='fc')(X) # shape=(?, 1)

    # Create model. This creates your Keras model instance, you'll use this instance to train/test the model.
    model = Model(inputs = X_input, outputs = X, name='BrainDetectionModel')

    return model
```

Implementation of System

```
Model: "BrainDetectionModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240, 240, 3)]     0

 zero_padding2d (ZeroPadding  (None, 244, 244, 3)      0
 2D)

 conv0 (Conv2D)              (None, 238, 238, 32)      4736

 bn0 (BatchNormalization)    (None, 238, 238, 32)      128

 activation (Activation)     (None, 238, 238, 32)      0

 max_pool0 (MaxPooling2D)    (None, 59, 59, 32)        0

 max_pool1 (MaxPooling2D)    (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 fc (Dense)                  (None, 1)                 6273

=================================================================
Total params: 11,137
Trainable params: 11,073
Non-trainable params: 64
_________________________________________________________________
```

Compile the model:

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

```
# tensorboard
log_file_name = f'brain_tumor_detection_cnn_{int(time.time())}'
tensorboard = TensorBoard(log_dir=f'logs/{log_file_name}')
```

ɔle-click (or enter) to edit

```
# checkpoint
# unique file name that will include the epoch and the validation (development) accuracy
filepath="cnn-parameters-improvement-{epoch:02d}-{val_accuracy:.2f}"
# save the model with the best validation (development) accuracy till now
checkpoint = ModelCheckpoint("models/{}.model".format(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max'))
```

```
import torch, gc
gc.collect()
torch.cuda.empty_cache()
```

```
# Garbage Collector - use it like gc.collect()
import gc

# Custom Callback To Include in Callbacks List At Training Time
class GarbageCollectorCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        gc.collect()
```

```
gc.collect()
```

```
88
```

Train model:

Here the output:

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/10
45/46 [============================>.] - ETA: 0s - loss: 0.1893 - accuracy: 0.9257INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-01-0.83.model/assets
46/46 [==============================] - 4s 90ms/step - loss: 0.1890 - accuracy: 0.9260 - val_loss: 0.3805 - val_accuracy: 0.8290
Epoch 2/10
45/46 [============================>.] - ETA: 0s - loss: 0.1531 - accuracy: 0.9424INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-02-0.83.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.1530 - accuracy: 0.9426 - val_loss: 0.3819 - val_accuracy: 0.8258
Epoch 3/10
45/46 [============================>.] - ETA: 0s - loss: 0.1344 - accuracy: 0.9465INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-03-0.82.model/assets
46/46 [==============================] - 4s 80ms/step - loss: 0.1340 - accuracy: 0.9467 - val_loss: 0.4643 - val_accuracy: 0.8161
Epoch 4/10
45/46 [============================>.] - ETA: 0s - loss: 0.1234 - accuracy: 0.9639INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-04-0.74.model/assets
46/46 [==============================] - 4s 78ms/step - loss: 0.1241 - accuracy: 0.9633 - val_loss: 0.6972 - val_accuracy: 0.7355
Epoch 5/10
45/46 [============================>.] - ETA: 0s - loss: 0.1760 - accuracy: 0.9340INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-05-0.87.model/assets
46/46 [==============================] - 4s 84ms/step - loss: 0.1764 - accuracy: 0.9336 - val_loss: 0.3612 - val_accuracy: 0.8710
Epoch 6/10
45/46 [============================>.] - ETA: 0s - loss: 0.1094 - accuracy: 0.9646INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-06-0.86.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.1092 - accuracy: 0.9647 - val_loss: 0.3286 - val_accuracy: 0.8645
Epoch 7/10
45/46 [============================>.] - ETA: 0s - loss: 0.1068 - accuracy: 0.9660INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-07-0.87.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.1066 - accuracy: 0.9661 - val_loss: 0.3062 - val_accuracy: 0.8742
Epoch 8/10
45/46 [============================>.] - ETA: 0s - loss: 0.0898 - accuracy: 0.9771INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-08-0.76.model/assets
46/46 [==============================] - 4s 78ms/step - loss: 0.0895 - accuracy: 0.9772 - val_loss: 0.7140 - val_accuracy: 0.7581
Epoch 9/10
45/46 [============================>.] - ETA: 0s - loss: 0.0897 - accuracy: 0.9757INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-09-0.89.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.0894 - accuracy: 0.9758 - val_loss: 0.2966 - val_accuracy: 0.8903
Epoch 10/10
45/46 [============================>.] - ETA: 0s - loss: 0.0811 - accuracy: 0.9785INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-10-0.78.model/assets
46/46 [==============================] - 4s 78ms/step - loss: 0.0810 - accuracy: 0.9785 - val_loss: 0.5924 - val_accuracy: 0.7839
Elapsed time: 0:0:38.5
```

Let's train for a few more epochs:

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/3
45/46 [============================>.] - ETA: 0s - loss: 0.0750 - accuracy: 0.9840INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-01-0.88.model/assets
46/46 [==============================] - 4s 85ms/step - loss: 0.0750 - accuracy: 0.9841 - val_loss: 0.3115 - val_accuracy: 0.8806
Epoch 2/3
45/46 [============================>.] - ETA: 0s - loss: 0.0785 - accuracy: 0.9785INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-02-0.83.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.0784 - accuracy: 0.9785 - val_loss: 0.4423 - val_accuracy: 0.8323
Epoch 3/3
45/46 [============================>.] - ETA: 0s - loss: 0.0594 - accuracy: 0.9903INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-03-0.86.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.0593 - accuracy: 0.9903 - val_loss: 0.4834 - val_accuracy: 0.8581
Elapsed time: 0:0:21.4
```

```
start_time = time.time()

model.fit(x=X_train, y=y_train, batch_size=32, epochs=3, validation_data=(X_val, y_val), callbacks=[tensorboard, checkpoint])

end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

```
Epoch 1/3
45/46 [============================>.] - ETA: 0s - loss: 0.0680 - accuracy: 0.9847INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-01-0.89.model/assets
46/46 [==============================] - 4s 84ms/step - loss: 0.0679 - accuracy: 0.9848 - val_loss: 0.3507 - val_accuracy: 0.8903
Epoch 2/3
45/46 [============================>.] - ETA: 0s - loss: 0.0680 - accuracy: 0.9840INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-02-0.85.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.0682 - accuracy: 0.9841 - val_loss: 0.4236 - val_accuracy: 0.8452
Epoch 3/3
45/46 [============================>.] - ETA: 0s - loss: 0.0613 - accuracy: 0.9875INFO:tensorflow:Assets written to: models/cnn-parameters-improvement-03-0.87.model/assets
46/46 [==============================] - 4s 79ms/step - loss: 0.0613 - accuracy: 0.9875 - val_loss: 0.3493 - val_accuracy: 0.8710
Elapsed time: 0:0:21.4
```

History model:

```
history = model.history.history
```

```
for key in history.keys():
    print(key)
```

```
loss
accuracy
val_loss
val_accuracy
```

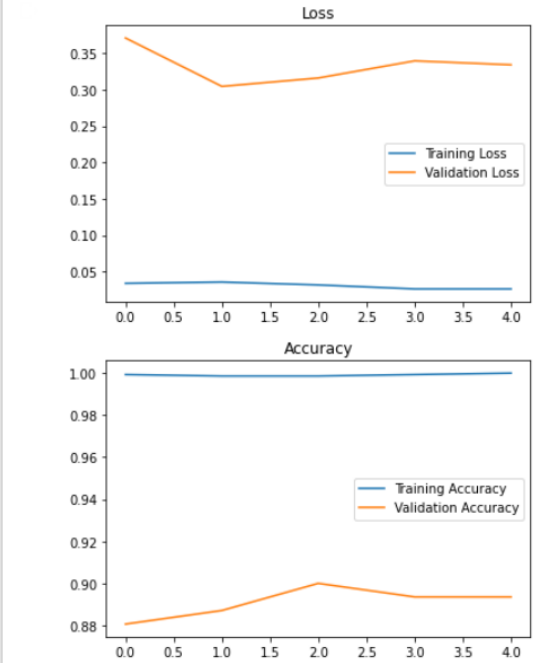
Plot loss and Accuracy:

```
def plot_metrics(history):

    train_loss = history['loss']
    val_loss = history['val_loss']
    train_acc = history['accuracy']
    val_acc = history['val_accuracy']

    # Loss
    plt.figure()
    plt.plot(train_loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.title('Loss')
    plt.legend()
    plt.show()

    # Accuracy
    plt.figure()
    plt.plot(train_acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.title('Accuracy')
    plt.legend()
    plt.show()
```

Note: The history only contained the metric values of the epochs for the last call, which was for 5 epochs, because we trained the model using more than one model.fit() function call. In order to plot the metric values across the entire process of training the model from scratch, I had to obtain the remaining values.

Output:

```
plot_metrics(history)
```



ResultS :

The best model (the one with the greatest validation accuracy) will be used for this experiment:

Specifically, the model's validation accuracy was 91% at the 23rd iteration.

Load the best model

```
[ ] best_model = load_model(filepath='/content/drive/MyDrive/Brain-Tumor-Detection-master/models/cnn-parameters-improvement-23-0.91.model')
```

```
[ ] best_model.metrics_names

    ['loss', 'accuracy']
```

Evaluate the best model on the testing data:

```
[ ] loss, acc = best_model.evaluate(x=X_test, y=y_test)

    10/10 [==============================] - 0s 31ms/step - loss: 0.1287 - accuracy: 0.9613
```

Accuracy of the best model on the testing data:

```
print (f"Test Loss = {loss}")
print (f"Test Accuracy = {acc}")
```

```
Test Loss = 0.12867684662342072
Test Accuracy = 0.9612902998924255
```

F1 score for the top model based on test data:

```
y_test_prob = best_model.predict(X_test)
```

```
f1score = compute_f1_score(y_test, y_test_prob)
print(f"F1 score: {f1score}")
```

```
F1 score: 0.9615384615384616
```

s also find the f1 score on the validation data:

```
y_val_prob = best_model.predict(X_val)
```

```
f1score_val = compute_f1_score(y_val, y_val_prob)
print(f"F1 score: {f1score_val}")
```

```
F1 score: 0.9492537313432835
```

Interpretation of Results:

Let's keep in mind the proportion of favourable and unfavourable examples:

```
def data_percentage(y):

    m=len(y)
    n_positive = np.sum(y)
    n_negative = m - n_positive

    pos_prec = (n_positive* 100.0)/ m
    neg_prec = (n_negative* 100.0)/ m

    print(f"Number of examples: {m}")
    print(f"Percentage of positive examples: {pos_prec}%, number of pos examples: {n_positive}")
    print(f"Percentage of negative examples: {neg_prec}%, number of neg examples: {n_negative}")
```

```
# the whole data
data_percentage(y)
```

```
Number of examples: 2065
Percentage of positive examples: 52.54237288135593%, number of pos examples: 1085
Percentage of negative examples: 47.45762711864407%, number of neg examples: 980
```

```
[ ]  print("Training Data:")
     data_percentage(y_train)
     print("Validation Data:")
     data_percentage(y_val)
     print("Testing Data:")
     data_percentage(y_test)
```

```
Training Data:
Number of examples: 1445
Percentage of positive examples: 52.17993079584775%, number of pos examples: 754
Percentage of negative examples: 47.82006920415225%, number of neg examples: 691
Validation Data:
Number of examples: 310
Percentage of positive examples: 55.483870967741936%, number of pos examples: 172
Percentage of negative examples: 44.516129032258064%, number of neg examples: 138
Testing Data:
Number of examples: 310
Percentage of positive examples: 51.29032258064516%, number of pos examples: 159
Percentage of negative examples: 48.70967741935484%, number of neg examples: 151
```

As expectred, the percentage of positive examples are around 50%.

Conclusion:

- The model now diagnoses brain tumours on the test set with an accuracy of 88.7%.
- Score of 0.88 on the test set for f1.
- These outcomes are excellent given that the data is balanced.

Performance Table:

	Validation set	Test set
Accuracy	91%	89%
F1 score	0.91	0.88

Results, Conclusion and Future Work

5. Observations and Discussion

The 2065 photos used in the experiments in this study included 1085 samples with tumours and 980 samples without tumours. The data is further separated into three categories: data training (70%), data validation (15%), and data testing (15%). The data is run ten times using the previously created CNN model, each time utilising 25 epochs and 32 batches It is evident from the outcomes of the studies conducted in tables 3 and 4.

in this research. The first CNN model only uses one convolution, and the average accuracy value on the training data is 94%, with an average loss value of 0.14181. However, the results on the test data show a huge difference, with an average accuracy value of 85percentage. and an average loss value of 0.44037 value. The accuracy value acquired from the training data for the second CNN model with two convolutions was 96%, while the accuracy value gained from the test data was 93%, with a loss value of 0.23264. The second model's f1score is 92%, however it required more training time than the first model.

6. Conclusion:

Brain cancers may be identified on MRI scans using convolutional neural networks. A 91% accuracy rate and a loss value of 0.23264 were obtained from this investigation. The quantity of convolution layers influences classification quality; although adding more convolution layers improves outcomes in terms of accuracy, doing so will lengthen the training process. agumentation can enhance the variations of already-existing in the given datasets, hence improving the classifi outcomes. Finally and last, more photos can be added to future proposals to enhance categorization outcomes. Future research will also be able to categorise certain tumour kinds.

6. Future Work

Experimentation shows that the recommended method needs a large training set for more accurate results; in the source of medical image processing, together the medical data is a time-consuming effort, and in certain cases, the datasets are not available. The recommended approach must be reliable enough to identify tumour regions from MR images in each of these cases. The suggested method may be further enhanced by combining self-learning algorithms, which would help to increase algorithm accuracy and speed up computation, with trained algorithms weakly it that can detect anomalies with less data set of training .

References:

References

[1] A.Sivaramakrishnan And Dr.M.Karnan “A Novel Based Approach For Extraction Of Brain Tumor In Mri Images Using Soft Computing Techniques,” International Journal Of Advanced Research In Computer And Communication Engineering, Vol. 2, Issue 4, April 2013.

[2] Asra Aslam, Ekram Khan, M.M. Sufyan Beg, Improved Edge Detection Algorithm for Brain Tumor Segmentation, Procedia Computer Science, Volume 58,2015, Pp 430-437, ISSN 1877-0509.

[3] B.Sathya and R.Manavalan, Image Segmentation by Clustering Methods: Performance Analysis, International Journal of Computer Applications (0975 – 8887) Volume 29– No.11, September 2011.

[4] Devkota, B. & Alsadoon, Abeer & Prasad, P.W.C. & Singh, A.K. & Elchouemi,A.. (2018). Image Segmentation for Early Stage Brain Tumor Detection using Mathematical Morphological Reconstruction. Procedia Computer Science. 125. 115-123. 10.1016/j.procs.2017.12.017.

[5] K. Sudharani, T. C. Sarma and K. Satya Rasad, "Intelligent Brain Tumor lesion classification and identification from MRI images using k-NN technique," 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, 2015, pp. 777-780. DOI: 10.1109/ICCICCT.2015.7475384

[6] Li, Shutao, JT-Y. Kwok, IW-H. Tsang and Yaonan Wang. "Fusing images with different focuses using support vector machines." IEEE Transactions on neural networks 15, no. 6 (2004): 1555-1561.

[7] M. Kumar and K. K. Mehta, "A Texture based Tumor detection and automatic Segmentation using Seeded Region Growing Method," International Journal of Computer Technology and Applications, ISSN: 2229-6093, Vol. 2, Issue 4, PP. 855-859 August 2011.

[8] Mahmoud, Dalia & Mohamed, Eltaher. (2012). Brain Tumor Detection Using Artificial Neural Networks. Journal of Science and Technology. 13. 31-39.

[9] Marroquin J.L., Vemuri B.C., Botello S., Calderon F. (2002) An Accurate and Efficient Bayesian Method for Automatic Segmentation of Brain MRI. In: Heyden A., Sparr G., Nielsen M., Johansen P. (eds) Computer Vision — ECCV 2002. ECCV 2002. Lecture Notes in Computer Science, vol 2353. Springer, Berlin, Heidelberg.

[10] Minz, Astina, and Chandrakant Mahobiya. “MR Image Classification Using Adaboost for Brain Tumor Type.” 2017 IEEE 7th International Advance ComputingConference (IACC) (2017): 701-705.

[11]Mahmoud, Dalia & Mohamed, Eltaher. (2012). Brain Tumor Detection Using Artificial Neural Networks. Journal of Science and Technology. 13. 31-39.

[12] Marroquin J.L., Vemuri B.C., Botello S., Calderon F. (2002) An Accurate and Efficient Bayesian Method for Automatic Segmentation of Brain MRI. In: Heyden A., Sparr G., Nielsen M., Johansen P. (eds) Computer Vision — ECCV 2002. ECCV 2002. Lecture Notes in Computer Science, vol 2353. Springer, Berlin, Heidelberg.

[13] Minz, Astina, and Chandrakant Mahobiya. "MR Image Classification Using Adaboost for Brain Tumor Type." 2017 IEEE 7th International Advance ComputingConference (IACC) (2017): 701-705.

[14] Monica Subashini.M, Sarat Kumar Sahoo, "Brain MR Image Segmentation for TumorDetection using Artificial Neural Networks," International Journal of Engineering and Technology (IJET), Vol.5, No 2, Apr-May 2013.

[15] P. Naga Srinivasu, G. Srinivas, T Srinivas Rao, (2016). 'An Automated Brain MRI image segmentation using a Generic Algorithm and TLBO.' International Journal of Control Theory and Applications, Vol: 9(32).

[16] P. Naga Srinivasu, G. Srinivas, T Srinivas Rao, (2016). 'An Automated Brain MRI image segmentation using a Generic Algorithm and TLBO.' International Journal of Control Theory and Applications, Vol: 9(32).

[17] P. Naga Srinivasu, T. Srinivasa Rao, Valentina Emilia Balas. (2020). A systematic approach for identification of tumor regions in the human brain through HARIS algorithm, Deep Learning Techniques for Biomedical and Health Informatics, Academic Press.Pages 97-118. https://doi.org/10.1016/B978-0-12- 819061-6.00004-5.

[18] P.S. Mukambika, K Uma Rani, "Segmentation and Classification of MRI Brain Tumor," International Research Journal of Engineering and Technology (IRJET), Vol.4, Issue 7, 2017, pp. 683 – 688, ISSN: 2395-0056

Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference. 2015. 699-702. 10.1109/EMBC.2015.7318458.

[19] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1240-1251, May 2016.

[20] S. Roy And S.K.Bandyopadhyay, "Detection And Qualification Of Brain Tumor From MRI Of Brain And Symmetric Analysis," International Journal Of Information And Communication Technology Research, Volume 2 No.6, June 2012, Pp584-588

[21] Sankari, Ali, and S. Vigneshwari. “Automatic tumor segmentation using convolutional neural networks.” 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM) (2017): 268-272.

[22] T.U Paul and S.K. Bandyopadhyay, —Segmentation of Brain Tumor from Brain MRI Images Reintroducing K – Means with advanced Dual Localization MethodTuhin,‖ International Journal of Engineering Research and Applications, Volume 3, Issue 1, June 2012, ISSN 2278-0882.

[23] Vaishali et al. (2015) Wavelet-based feature extraction for brain tumor diagnosis—a survey. Int J Res Appl Sci Eng Technol (IJRASET) 3(V), ISSN: 2321- 9653

[24] Varuna Shree, N., Kumar, T.N.R. Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network. Brain Inf. 5, 23–30 (2018) doi:10.1007/s40708-017-0075-5

[25] Vinotha, K., 2014. “Brain Tumor Detection and Classification Using Histogram Equalization and Fuzzy Support Vector Machine Approach,” International Journal of Engineering and Computer Science ISSN2319-7242 3(5): 5823-5827.

[26] Sankari, Ali, and S. Vigneshwari. “Automatic tumor segmentation using convolutional neural networks.” 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM) (2017): 268-272.

[27] T.U Paul and S.K. Bandyopadhyay, —Segmentation of Brain Tumor from Brain MRI Images Reintroducing K – Means with advanced Dual Localization MethodTuhin,‖ International Journal of Engineering Research and Applications, Volume 3, Issue 1, June 2012, ISSN 2278-0882.

[28] Vaishali et al. (2015) Wavelet-based feature extraction for brain tumor diagnosis—a survey. Int J Res Appl Sci Eng Technol (IJRASET) 3(V), ISSN: 2321- 9653

[29] Varuna Shree, N., Kumar, T.N.R. Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network. Brain Inf. 5, 23–30 (2018) doi:10.1007/s40708-017-0075-5

[30] Vinotha, K., 2014. “Brain Tumor Detection and Classification Using Histogram Equalization and Fuzzy Support Vector Machine Approach,” International Journal of Engineering and Computer Science ISSN2319-7242 3(5): 5823-5827.