

A Project/Dissertation
Review-2 Report

On

**BRAIN TUMOR DETECTION USING
VGG-16**

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

Bachelor of Technology in Computer Science and Engineering



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. Deependra Rastogi.

GROUP ID-: BT3264

Submitted By

19SCSE1180049 - SANYAM SHARMA
19SCSE1180074 - BHARAT

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTERS SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA



SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "**BRAIN TUMOR DETECTION USING VGG-16**" in partial fulfillment of the requirements for the award of the B. Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of **Mr. Deependra Rastogi**, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

19SCSE1180049 - SANYAM SHARMA
19SCSE1180074 – BHARAT

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

Mr. Deependra Rastogi.

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **19SCSE1180049 – SANYAM SHARMA, 19SCSE1180074 – BHARAT** has been held on and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date:

Place:

ABSTRACT

In the context of the brain tumors, these are the most common and aggressive disease, leading to a short life prospect in their uppermost grade. So, treatment planning is a vital stage to refine the quality of life of cases. Generally, various image techniques resembling as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and ultrasound image are used to evaluate the tumor in a brain, lung, liver, prostate ... etc. Especially, in this work MRI images are used to diagnose tumor in the brain.

Notwithstanding the huge volume of data generated by MRI check thwarts automatic set of tumor vs non-tumor in a particular time. But it is having some limitation (i.e.) accurate quantitative measures is handed for limited number of images. Hence trusted and automatic set scheme are essential to prevent the death rate of natural. The automatic brain tumor set is really challenging task in large spatial and structural variability of surrounding region of brain tumor.

In this, automatic brain tumor finding is proposed by using an algorithm Known as VGG-16 which is a part of Convolutional Neural Networks (CNN) classification and Deep Learning. The deeper shell design is performed by using small kernels. The weight of the neuron is given as small. Experimental results show that the CNN libraries rate of 97.5 accuracy with low complexity and compared with all other state of arts methods.

Contents

Title	Page No.
Candidates Declaration	
Acknowledgement	
Abstract	
Contents	
List of Figures	
Acronyms	
Chapter 1 Introduction	8
1.1 Introduction	8
1.2 Formulation of Problem	9
1.2.1 Tool and Technology Used	11
Chapter 2 Literature Survey/Project Design	14
Chapter 3 Functionality/Working of Project	20
Chapter 4 Results and Discussion	23
Chapter 5 Conclusion and Future Scope	25
5.1 Conclusion	
5.2 Future Scope	
Reference	27

List of Figures

S.No.	Title	Page No.
1	Basic Architecture of CNN &VGG-16 model	13
2	Design of model	19
3	VGG-16 based classified results	24
4	Block diagram of proposed brain tumor	24

Acronyms

B.Tech.	Bachelor of Technology
MTech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

Chapter-1

Introduction

Brain tumor is one of the vital organs in the natural body, which consists of billions of cells. The abnormal group of cells is formed from the abandoned division of cells, which is also called as lump. Brain tumors are divided into two types matching low grade (grade1 and grade2) and high grade (grade3 and grade4) tumor. Low grade brain tumor is called as benign. Likewise, the high-grade tumor is also called as malignant. Benign tumor isn't cancerous tumor. Hence it doesn't spread other region of the brains. Notwithstanding the malignant tumor is a cancerous tumor. So, it spreads fast with indefinite boundaries to other region of the body freely. It leads to immediate death.

Brain MRI image is generally used to dig out the tumor and tumor progress modeling process. This information is generally used for growth spotting and treatment processes. MRI image gives farther information about given medical image than the CT or ultrasound image. MRI image provides detailed information about brain structure and anomaly spotting in brain tissue. Scholars offered unlike automated strategies for brain growths finding and type cataloging using brain MRI images from the time when it ran possible to overlook and freight medical images to the computer. Conversely, Neural Networks (NN) and Support Vector Machine (SVM) are the

usually used techniques for their good enactment over the most recent untold years. Not with standing new, Deep Learning (DL) models fixed a stirring trend in machine learning as the underground architecture can efficiently represent complex relationships without wanting many nodes like in the superficial skeletons e.g., K- Nearest Neighbor (KNN) and Support Vector Machine (SVM). Therefore, theygrew fast to become the state of the art in other health informatics areas for example medical image analysis, medical informatics and bioinformatics.

A convolutional neural network (CNN) is a class of artificial neural network, uttermost ordinarily applied to analyze visual imagery. They're also known as shift steady or space steady artificial neural networks (SIANN), based on the participated-weight architecture of the convolution kernels or filters that slide along input features and feed translation equivariant responses known as feature charts.

VGG16 is a simple and widely used Convolutional Neural Network (CNN) Architecture used for ImageNet, a large visual database project used in visual object recognition software research. The VGG16 model achieved 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

VGG16 is used in many deep learning image classification techniques and is popular due to its ease of implementation. VGG16 is extensively used in learning applications due to the advantage that it has.

VGG16 is a CNN Architecture, which was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. It is still one of the best vision architectures to dates.

During training, the input to the convnets is a fixed-size 224 x 224 RGB image. Subtracting the mean RGB value computed on the training set from each pixel is the only pre-processing done here. The image is passed through a stack of convolutional (conv.) layers, where filters with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center and has the same effective receptive field as one 7×7), is used. It is deeper, has more non-linearities, and has fewer parameters. In one of the configurations, 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity), are also utilized. The convolution stride and the spatial padding of conv. layer input is fixed to 1 pixel for 3×3 convolutional layers, which ensures that the spatial resolution is preserved after convolution. Five max-pooling layers, which follow some of the convolutional layers, helps in spatial pooling. Max-pooling is performed over a 2×2 -pixel window, with stride 2.

Types of Tumors:

Benign Tumor

Malignant Tumor

- Benign Tumor:
 - Not Grow
 - Non-Cancerous (Easily remove)

- Malignant Tumor: Spread Cancerous

1.2.1 Tool and Technology Used

Dataset-

This study uses four different data sets that are available in publicly accessible databases. The first data set is called the Reference Image Database for Assessing Response to Therapy [14] The total number of images in this data set is 70,220. The second data set is called The Repository of Molecular Brain Neoplasia Data (REMBRANDT) (Lisa et al. 2015). The REMBRANDT data set contains a multi-sequential magnetic resonance imaging of 130 patients with gliomas grade II, grade III and grade IV. The total number of images in this data set is 110,020. The third set of data is called Cancer Genome Atlas Low- Grade Glioma (TCGALGG) (Pedano et al. The TCGALGG data set contains 241,183 magnetic resonance images of 199 patients with low- grade gliomas (Grade I and Grade II). These three sets of data are from the Cancer Imaging Archive Project (TCIA) (Clark et al. 2013). Each case was multimodal with contrast enhanced T1 and FLAIR images. Another data set used in this study (Cheng et al. 2015) contains 3064 contrasts weighted T1 images of 233 patients with three types of brain tumors: glioma (1426 sections), meningioma (708 sections) and pituitary gland (930 sections). Total of 2990 images are collected, including 1640 tumor and 1350 non-tumor images. A total of 3950

images are collected for the classification² task, including 850 normal images, 950 gliomas, 700 meningiomas, 700 pituitary glands and 750 metastases. For the classification³ task, a total of 4,570 images are collected, including 1,676 grade II, 1,218 grade III and 1,676 grade III.

Convolutional Neural Network With VGG-16

The most widely used deep learning model among neural networks is the CNN model. A typical model consists of two parts: feature extraction and classification. The CNN architecture generally has five main layers: input layer, convolution layer, grouping layer, fully connected layer, and classification. layer. CNN performs feature extraction and classification through trainable layers that are placed sequentially one behind the other. The feature extraction part of CNN generally includes the folding and grouping layers, the classification part includes the fully connected and classification layers. Although CNN has focused on picture classification and accepts pictures as input data in recent years, it has been widely used in many other fields such as audio and video whose input data can be any signal. There are three Fully Connected (FC) layers that follow a stack of convolutional layers (these have different depths in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one

for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

The 16-layer VGG architecture was the best performing, and it achieved a top-5 error rate of 7.3% (92.7% accuracy) in ILSVRC — 2014, as mentioned above. VGG16 had significantly outperformed the previous generation of models ILSVRC — 2012 and ILSVRC — 2013 competitions.

The VGG16 architecture is depicted in figure 1-2, shown below:

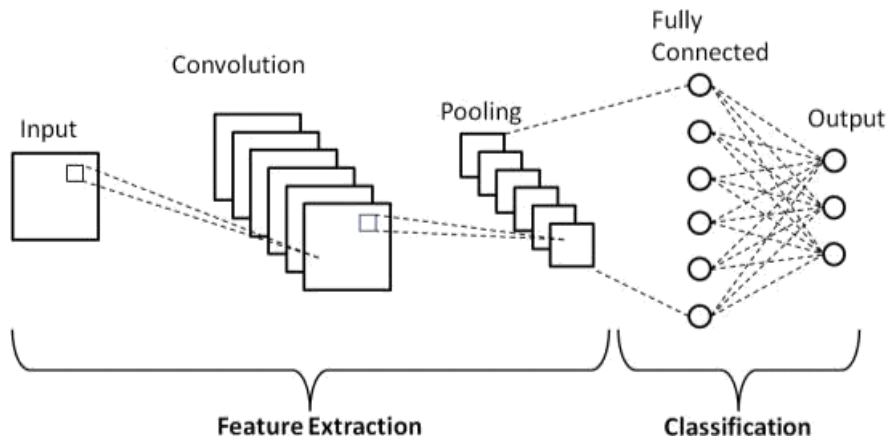


Fig.1 – Showing CNN Architecture

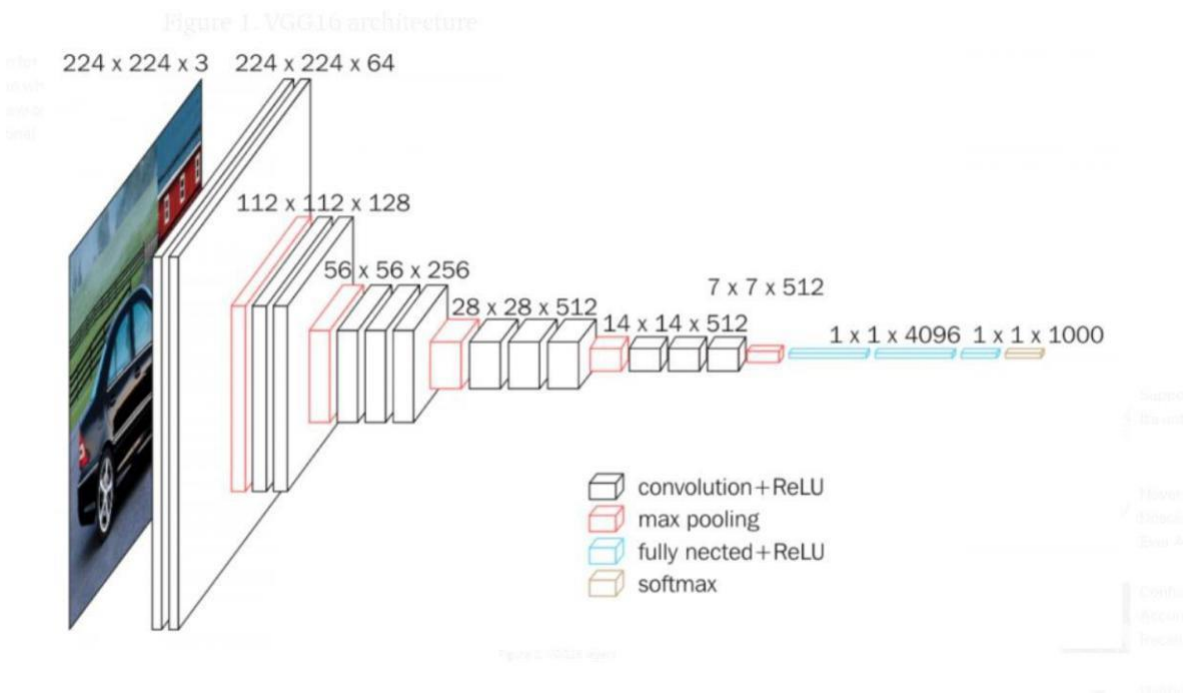


Fig.2 – Showing VGG-16 Architecture

Chapter 2

Literature Survey/Project Design

In recent years, several studies have applied data mining algorithms on different medical datasets to classify Brain Tumor. These algorithms show good classification results and encourage many researchers to apply these kinds of algorithms to solve challenging tasks. In this VGG-16 which is a part of convolutional neural network (CNN) was used to predict and classify the invasive ductal carcinoma in brain histology images with an accuracy of almost 88%. Moreover, data mining is used widely in medical fields to predict and classify abnormal events to create a better understanding of any incurable diseases such as cancer. The outcomes of using data mining in classification are promising for Brain Tumor detection. Therefore, data mining approach is used in this work.

Proposed System

The human mind is modeled by way of using layout and implementation of neural network. The neural network is especially used for vector quantization, approximation, statistics clustering, pattern matching, optimization capabilities and classification techniques. The neural network is split into three kinds based totally on their inter connections. Three type neural networks are comments, feed forward and recurrent community. The Feed Forward Neural network is in addition divided into unmarried layer community and multilayer community. In the single layer network, the hidden layer is not supplied. But it carries best input and output layer. However, the multilayer includes enter layer, hidden layer and output layer. The closed loop primarily based remarks community is called as recurrent network. IN the everyday neural community, photocannot scalable. But in convolution neural network, photograph can scalable (I. E) it'll take 3-D input quantity to 3-d output quantity (length,width, top). The VGG-16 with (CNN) includes input layer, convolution layer, Rectified Linear Unit (Re LU) layer, pooling layer and completely related layer. In the convolution layer, the given

enter picture is separated into numerous small areas. Element sensible activation function is finished in ReLU layer. Pooling layer is non-compulsory. We can use or skip. However, the pooling layer is mainly used for down sampling. In the final layer (i.e.) completely related layer is used to generate the elegance rating or label rating price based totally at the possibility in among zero to one. The block diagram of mind tumor type primarily based on convolution neural community is proven in fig.1. The CNN based mind tumor classification is split into levels along with education and testing stages. The quantity of photographs is split into exceptional class by using the use of labels name along with tumor and non-tumor mind picture...and many others. In the schooling segment, preprocessing, characteristic extraction and class with Loss feature is performed to make a prediction version. Initially, label the schooling picture set. In the preprocessing image resizing is applied to change length of the photo. Finally, the convolution neural network is used for computerized brain tumor category. The mind image dataset is taken from photo internet. Image internet is a one of the pre-skilled versions. If you need to train from the starting layer, we've to train the entire layer (i.e.) up to finishing layer. So, time intake may be very excessive. It will influence the performance. To keep away from this type of

trouble,

pre-educate version-based brain dataset is used for category steps.

In the proposed CNN, we can teach best final layer in python

implementation. We don't need to train all the layers. So,

computation time is low meanwhile the performance is excessive in the proposed

automated brain tumor category scheme. The loss characteristic is calculated by

using gradient descent set of rules.

The uncooked image pixel is mapping with class scores by using a score

function. The satisfactory of precise set of parameters

is measured by loss function. It is based totally on how properly the caused ratings

accredited with the ground truth labels within the training statistics. The loss function

calculation may be very essential to enhance the accuracy. If the loss characteristic is

excessive, whilst the accuracy is low. Similarly, the accuracy is high, while the loss

feature is low. The gradient fee is calculated for loss feature to compute gradient descent

set of rules. Repeatedly compare the gradient

value to compute the gradient of loss characteristic.

In this research, we applied Image Processing and Data Augmentation techniques on a

small dataset of 253 brain MRI images. We trained them through a simple 8 Convolutional

layers CNN model and compared our scratched CNN model accuracy with pre-trained

VGG-16, ResNet-50, and Inception-v3 models using transfer learning

approach. The dataset includes 155 images of malignant cancer and 98

of benign non-cancerous tumors. We split our dataset into 3 separate segments for training, validation, and testing.

The training data is for model learning, validation data is sample data for model evaluation and model parameters tuning. Test data is for the final evaluation of our model. Our proposed method is composed of various phases

Chapter 3

Functionality/Working of Project

Convolution Layers- There are 3 types of layers that make up the CNN, namely the folding layers, the grouping layers, and the fully connected (FC) layers. When these layers are stacked, a CNN structure could form. In addition to these 3 layers, there are additional critical parameter layers which are the dropout layer and the activation characteristic, which are described below.

1. Convolutional Layer-

This layer is the first layer used to extract the various properties from the input images. In this layer the mathematical convolution operation is carried out between the input image and a filter of a certain size $M \times M$. By moving the filter over the input image, the dot product between the filter and the parts of the input image in relation to the filter size ($M \times M$) is formed.

2. Pooling Layer-

In most cases, a folding layer is followed by a grouping layer. The main goal of this layer is to reduce the size of the folded feature map in order to reduce computational costs. It does this by reducing the connections between the layers and working independently in each of them. Feature Map There are different types of grouping operations depending on the method used.

3. Fully Connected Layer-

The input image from the previous levels is flattened and fed to the FC level. The flattened vector then undergoes a few additional FC layers, where the operations of the mathematical functions normally take place. In this phase the classification process begins. 4. Dropout- When all functions are connected to the FC layer, there can usually be an overfitting in the training data set. Overfitting occurs when a particular model performs so well on training data, which has a negative impact on model performance when used with new data. To overcome this problem a demolition layer is used where some neurons are removed from the neural network during the training process, resulting in a reduced model size.

5. Activation Functions-

Finally, one of the most important parameters of the CNN model is the activation function, it is used to know and approximate any kind of continuous and complex relationship between variables of the network, in simple terms, it decides what information should be triggered in the model. in the forward direction and which are not at the end of the network. Adds non-linearity to the network.

color shift (Krizhevsky et al., 2012). Algorithm for

CNN based Classification

1. Apply convolution filter out in first layer
2. The sensitivity of clear out is reduced with the aid of smoothing the convolution filter (i.e.) subsampling
3. The sign transfers from one layer to every other layer is managed by activation layer
4. Fasten the schooling period by means of the usage of rectified linear unit (RELU)
5. The neurons in proceeding layer are hooked upto each neuron in next layer
6. During training Loss layer is brought on the end to deliver feedback to neural network.

How to train VGG16 from scratch?

1. The ConvNet training procedure is generally carried out by optimizing the multinomial logistic regression objective using mini- batch gradient descent (based on back-propagation) with momentum. The batch size was set to 256, and momentum was to 0.9. The training was regularized by weight decay (the L2 penalty multiplier set to $5 \cdot 10^{-4}$) and dropout regularization for the first two fully connected layers (dropout ratio set to 0.5).

The learning rate was initially set to 10^{-2} and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370,000 iterations (74 epochs). It is conjectured that despite the larger number of parameters and the greater depth of convolutional networks, the nets required fewer epochs to converge due to (a) implicit regularization imposed by the greater depth and smaller conv. filter sizes; (b) pre-initialization of certain layers.

2. The initialization of the network weights is important since bad initialization can stall learning due to the instability of the gradient in deep nets. To circumvent this problem, the training of configuration A (Table 1), which is shallow enough to be trained with the random

initialization, is begun.

Then, when training deeper architectures, the first four convolutional layers and the last three fully connected layers with the layers of net A (the intermediate layers were initialized randomly) are initialized.

The learning rate for the pre-initialized layers is not decreased, allowing them to change during learning. For random initialization (where applicable), the weights from a normal distribution with the zero mean and 10^{-2} variance are sampled. The biases were initialized with zero.

3. It is worth noting that after the paper submission of the original work done by Karen Simonyan and Andrew Zisserman, it was found to be possible to initialize the weights without pre-training by using the random initialization procedure of Glorot.

4. The ConvNet input images were randomly cropped from rescaled training images (one crop per image per SGD iteration) to obtain the fixed-size 224×224 ConvNet images. To further augment the training set, the crops underwent random horizontal flipping and random RGB.

3.1 PROJECT CODING:

In [61]:

```
import pandas as pd
import os
```

In [62]:

```
from IPython.display import clear_output
!pip install imutils
!pip install cv2
clear_output()
```

In [63]:

```
# !pip install opencv-python
```

In [64]:

```
# !pip install plotly==5.4.0
```

In [65]:

```
# !pip install Keras
```

In [66]:

```
# !pip install tensorflow
```

In [67]:

```
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 accuracy_score, 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 tensorflow.keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping

init_notebook_mode(connected=True)
RANDOM_SEED = 123
```

In [69]:

```
pwd
```

Out[69]:

```
'C:\\Users\\Abhishek Raj\\Downloads\\archive2'
```

In [70]:

```
cd C:\\Users\\Abhishek Raj\\Downloads\\archive2
```

```
C:\\Users\\Abhishek Raj\\Downloads\\archive2
```

In [71]:

```
pwd
```

Out[71]:

```
'C:\\Users\\Abhishek Raj\\Downloads\\archive2'
```

In [81]:

```
!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
```

Folder PATH listing for volume OS

Volume serial number is EC44-595E

C:.

```
+---brain_tumor_dataset
```

```
| +---no
```

```
| +---
```

```
yes +---no
```

```
+---TEST
```

```
| +---NO
```

```
| +---YES
```

```
+---TRAIN
```

```
| +---NO
```

```
| +---YES
```

```
+---VAL
```

```
| +---NO
```

```
| +---YES
```

```
+---yes
```

In [85]:

```
pwd
```

Out[85]:

```
'C:\\Users\\Abhishek Raj\\Downloads\\archive2'
```

In [84]:

```
IMG_PATH = 'brain_tumor_dataset/'
# split the data by train/val/test
for CLASS in os.listdir(IMG_PATH):
    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 < 5:
                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)
```

In [86]:

```
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]

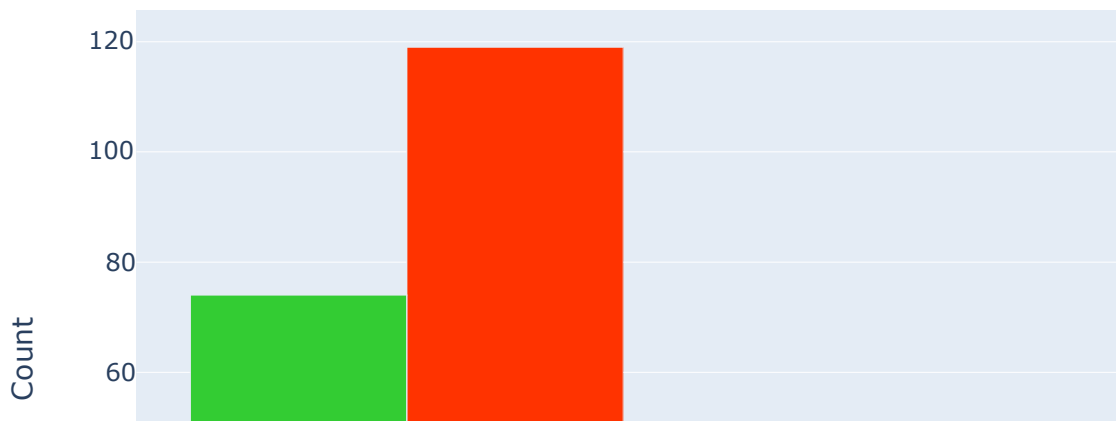
    thresh = cm.max() / 2.
    cm = np.round(cm,2)
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True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


In [88]:

```
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)
```


Count of classes in each set



In [96]:

```
# from sklearn.metrics import plot_samples
```

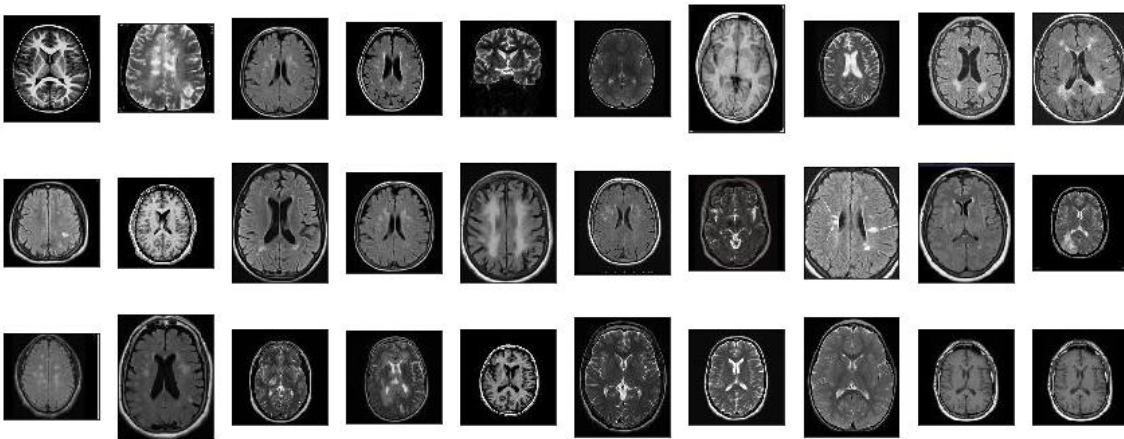
In [99]:

```
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()
```

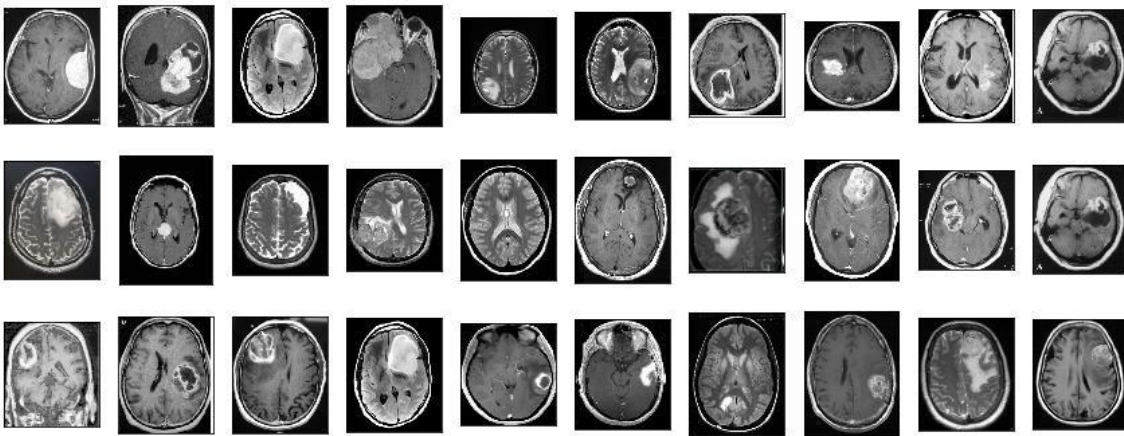
In [100]:

```
plot_samples(X_train, y_train, labels, 30)
```

Tumor: NO



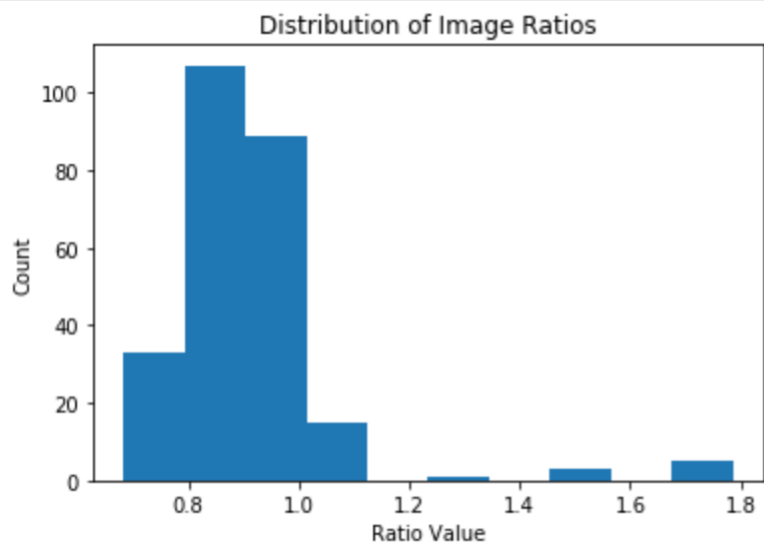
Tumor: YES



In [101]:

```
RATIO_LIST = []
for set in (X_train, X_test, X_val):
    for img in set:
        RATIO_LIST.append(img.shape[1]/img.shape[0])

plt.hist(RATIO_LIST)
plt.title('Distribution of Image Ratios')
plt.xlabel('Ratio Value')
plt.ylabel('Count')
plt.show()
```



In [102]:

```
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_SIMP
LE)
        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)
```

In [104]:

```
img = cv2.imread('brain_tumor_dataset/yes/Y108.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)

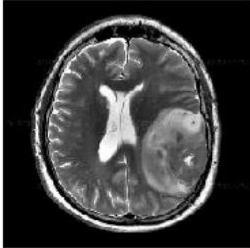
# add extreme points
img_pnt = cv2.circle(img_cnt.copy(), extLeft, 8, (0, 0, 255), -1)
img_pnt = cv2.circle(img_pnt, extRight, 8, (0, 255, 0), -1)
img_pnt = cv2.circle(img_pnt, extTop, 8, (255, 0, 0), -1)
img_pnt = cv2.circle(img_pnt, extBot, 8, (255, 255, 0), -1)

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

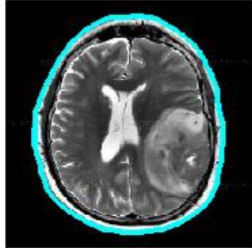
In [105]:

```
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()
```

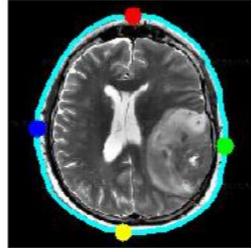
Step 1. Get the original image



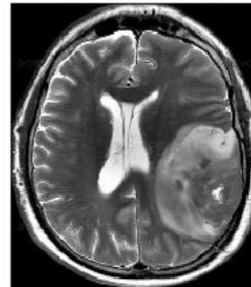
Step 2. Find the biggest contour



Step 3. Find the extreme points



Step 4. Crop the image



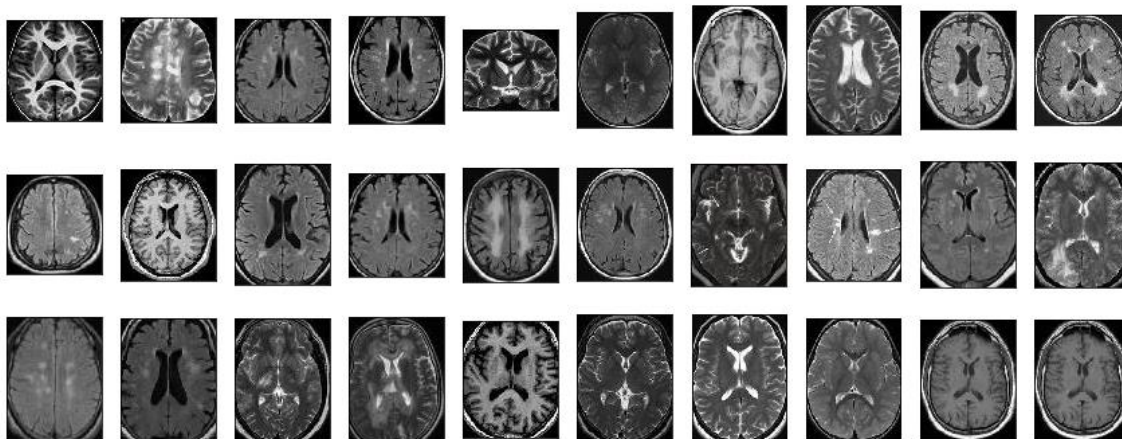
In [106]:

```
# apply this for each set
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)
```

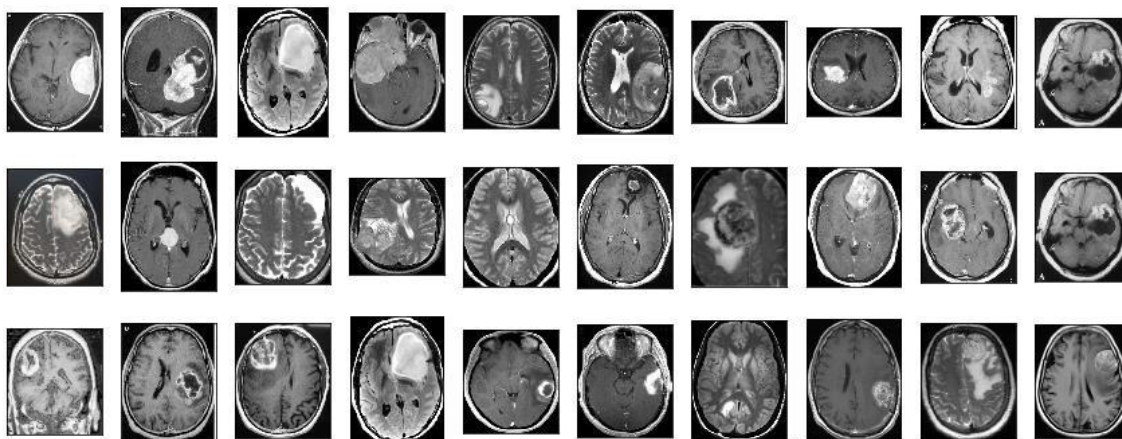
In [107]:

```
plot_samples(X_train_crop, y_train, labels, 30)
```

Tumor: NO



Tumor: YES



In [108]:

```
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
```

In [110]:

```
# 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/')
```

In [111]:

```
def preprocess_imgs(set_name, img_size):
    """
    Resize and apply VGG-16 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)
```

In [112]:

```
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)
```

In [113]:

```
# set the paramters we want to change randomly
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
)
```


In [114]:

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

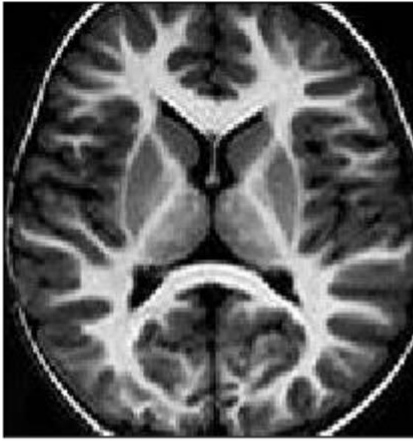
i = 0
for batch in demo_datagen.flow(x, batch_size=1, save_to_dir='preview',
save_prefix='aug_img', save_format='jpg'):
    i += 1
    if i > 20:
        break
```

In [115]:

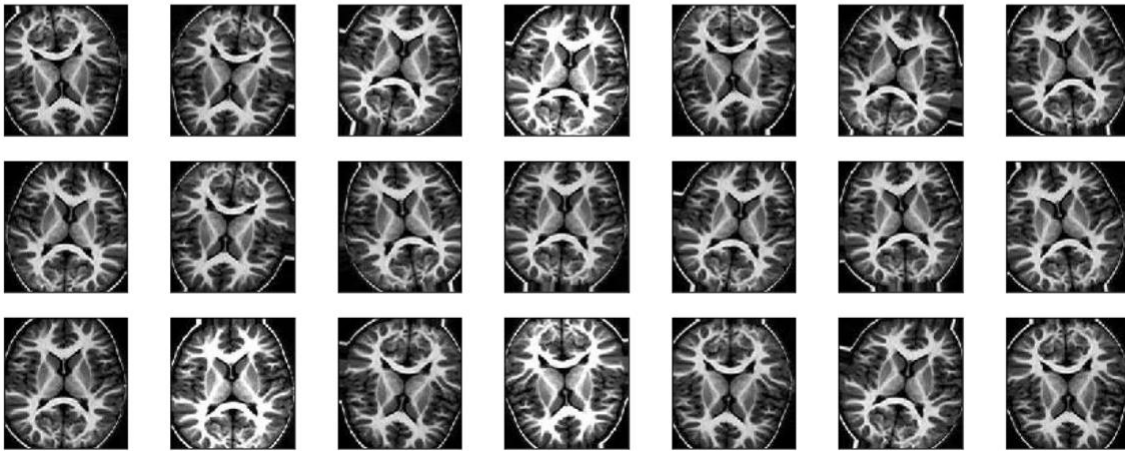
```
plt.imshow(X_train_crop[0])
plt.xticks([])
plt.yticks([])
plt.title('Original Image')
plt.show()

plt.figure(figsize=(15,6))
i = 1
for img in os.listdir('preview/'):
    img = cv2.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('Augmented Images')
plt.show()
```

Original Image



Augmented Images



In [117]:

```
TRAIN_DIR = 'TRAIN_CROP/'
VAL_DIR = 'VAL_CROP/'

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
)
```

Found 193 images belonging to 2 classes.
Found 50 images belonging to 2 classes.

In [120]:

```
# Load base model
vgg16_weight_path = 'keras-pretrained-
models/vgg16_weights_tf_dim_ordering_tf_kernels_n_top.h5'
base_model = VGG16(
    weights=vgg16_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)
```

In [145]:

```
NUM_CLASSES = 1

model = Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(NUM_CLASSES, activation='sigmoid'))

model.layers[0].trainable = False

model.compile(
    loss='binary_crossentropy',
    optimizer=RMSprop(learning_rate=1e-4),
    metrics=['accuracy']
)

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten_1 (Flatten)	(None, 25088)	0
dropout_1 (Dropout)	(None, 25088)	0
dense_1 (Dense)	(None, 1)	25089

```
=====  
Total params: 14,739,777  
Trainable params: 25,089  
Non-trainable params: 14,714,688  
=====
```

In [146]:

```
EPOCHS = 30
es = EarlyStopping(
    monitor='val_acc',
    mode='max',
    patience=6
)

history = model.fit(
    train_generator,
    steps_per_epoch=50,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=25,
    callbacks=[es]
)
```

Epoch 1/30

```
7/50 [==>.....] - ETA: 4:07 - loss: 4.8184 - accuracy: 0.6166WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 1500 batches). You may need to use the repeat() function when building your dataset. WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 25 batches). You may need to use the repeat() function when building your dataset.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
50/50 [=====] - 83s 1s/step - loss: 4.8184 - accuracy: 0.6166 - val_loss: 4.0546 - val_accuracy: 0.6600
```

In [125]:

```
history.history
```

Out[125]:

```
{'loss': [5.365586757659912],
 'accuracy': [0.6113989353179932],
 'val_loss': [2.1493048667907715],
 'val_accuracy': [0.7200000286102295]}
```

In [142]:

```
# validate on val 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, labels = list(labels.items()), normalize=False)
```

Val Accuracy = 0.68

In [143]:

```
confusion_mtx
```

Out[143]:

```
array([[10,  9],
       [ 7, 24]], dtype=int64)
```

In [144]:

```
import seaborn as sns

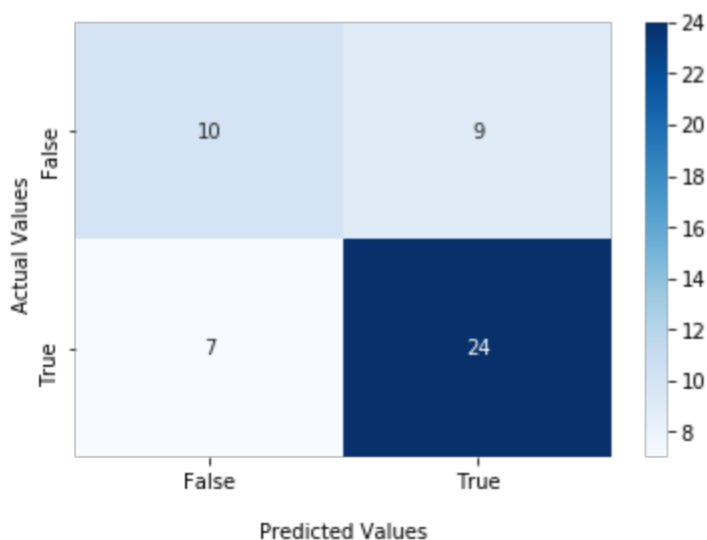
ax = sns.heatmap(confusion_mtx, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with
labels\n\n'); ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket Labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False', 'True'])
ax.yaxis.set_ticklabels(['False', 'True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```

Seaborn Confusion Matrix with labels



In [148]:

```
# 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)
```

Test Accuracy = 0.40

In [149]:

```
import seaborn as sns

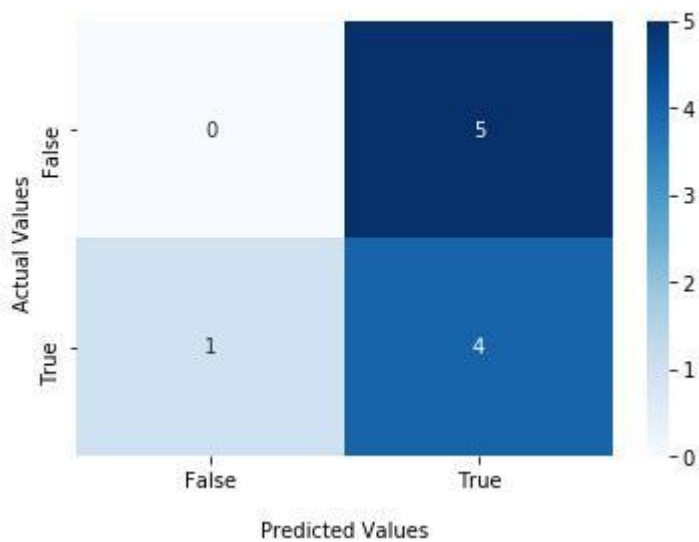
ax = sns.heatmap(confusion_mtx, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with
labels\n\n'); ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket Labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False', 'True'])
ax.yaxis.set_ticklabels(['False', 'True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```

Seaborn Confusion Matrix with labels



In []:

Chapter-4

Results and Discussion

Our dataset contains tumor and non-tumor magnetic resonance images, and they were collected from various online resources containing real patient cases, tumor images were obtained from Radiopaedia and Brain Tumor Image Segmentation Benchmark (BRATS) 2015 test dataset 14. In this thesis an efficient automatic detection of brain tumors using the Convolution Neural Network is carried out. The simulation is carried out using the Python language. The accuracy is calculated and compared with all other Vanguard methods. Training Precision, Validation Precision and Loss of Validation are calculated to determine the efficiency of the proposed brain tumor classification scheme.

In, the existing technology, the classification based on the Support Vector Machine (SVM), was used to detect brain tumors Requires the feature extraction output. Based on the feature value, the classification output is generated, and the precision is calculated. The computation time is high, and the precision is low in -based tumor and non-tumor detection SVMs. In the proposed model system does not require separate feature extraction steps. The value of the feature is taken from the VGG-16 itself. In Fig. 4. shows the classified result of the brain image of tumor and non-tumor. The result of the precision classification of brain tumor

based on probability value. The normal brain image of has the lowestprobability score.

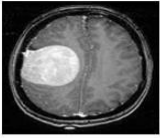
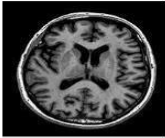
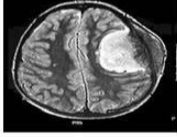
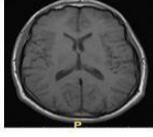
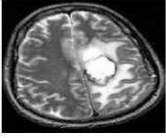
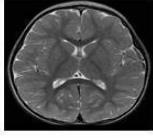
Brain Tumor Image	Brain Non Tumor Image
	
	
	

Figure 3 VGG-16 based classified results

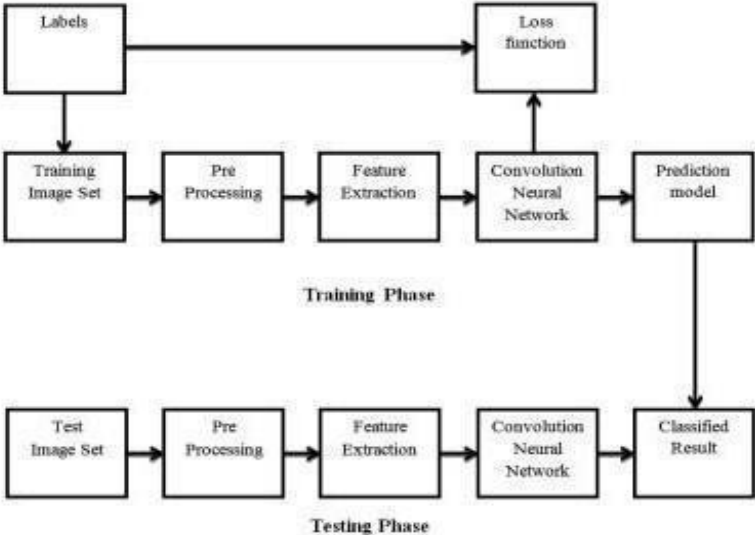


Figure 4 Block diagram of proposed brain tumor

Chapter 5

Conclusion and Future Scope

In this paper, a new approach was presented to classify brain tumors. First, using the image edge detection technique, we find the region of interest in MRI images and cropped them then, we used the data augmentation technique for increasing the size of our training data. Second, we provide an efficient methodology for brain tumor classification by proposing a simple vgg-16 network. For sophisticated and accurate results neural network requires a large amount of data to train on, but our experimental result shows that even on such a small dataset.

Our proposed system can play a prognostic significance in the detection of tumors in brain tumor patients. To further boost the model efficiency, comprehensive hyperparameter tuning, and a better preprocessing technique can be conceived. Our proposed system is for binary classification problems, however, in future work, the proposed method can be extended for categorical classification problems such as identification of brain tumor types such as Glioma, Meningioma, and Pituitary or may be used to detect other brain abnormalities. Also, our proposed system can play an effective role in the early diagnosis of dangerous disease,

particularly lung cancer and breast cancer whose mortality rate is very high globally. We can prolong this approach in other scientific areas as well where there is a problem in the availability of large data, or we can use the different transfer learning methods with the same proposed technique.

Reference

1. Heba Mohsen et al, "Classification using Deep Learning Neural Networks for Brain Tumors", Future Computing and Informatics, pp 1-four (2017).
2. Stefan Bauer et al, "Multiscale Modeling for Image Analysis of Brain Tumor Studies", IEEE Transactions on Biomedical Engineering, fifty- nine (1): (2012).
3. Atid Islam et al, "Multi-fractal Texture Estimation for Detection and Segmentation of Brain Tumors", IEEE, (2013).
4. Meigan Huang et al, "Brain Tumor Segmentation Based on LocalIndependent Projection based Classification", IEEE Transactions on Biomedical Engineering, IEEE, (2013).
5. Andac Hammadi et al, "Tumor-Cut: Segmentation of Brain Tumors on Contrast Enhanced MR Images for Radiosurgery Applications", IEEE Transactions on Medical Imaging, 31(3): (2012).
6. Bjoern H. Menze et al, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging, (2014).

7. Jin Liu et al, “A Survey of MRI-Based Brain Tumor Segmentation Methods”, TSINGHUA Science and Technology, 19(6) (2011).
8. Shamsul Huda et al, “A Hybrid Feature Selection with Ensemble Classification for Imbalanced Healthcare Data: A Case Study for Brain Tumor Diagnosis”, IEEE Access, 4: (2017).
9. R. Karuppathal and V. Palanisamy, “Fuzzy based automatic detection and category technique for MRI-mind tumor”, ARPN Journal of Engineering and Applied Sciences, 9(12): (2014).
10. Janani and P. Meena, “photograph segmentation for tumor detection using fuzzy inference system”, International Journal of Computer Science and Mobile Computing, 2(5): 244 – 248 (2013).
11. Sergio Pereira et al, “Brain Tumor Segmentation the use of Convolutional Neural Networks in MRI Images”, IEEE Transactions on Medical Imaging, (2016).
12. Jiachi Zhang et al, “Brain Tumor Segmentation Based on Refined Fully Convolutional Neural Networks with A Hierarchical Dice Loss”, Cornell university library, pc imaginative and prescient and pattern popularity, (2018).
13. [Radiopaedia] [http:// radiopedia.Org](http://radiopedia.Org).
14. [BRATS 2015] <https://www.Smir.Ch/BRATS/ Start2012>



V7I6-1413: Dear author, Concession Provided, Issue is about to end.

Congratulations!

This issue is going to close very soon. There is a special concession provided to a few selected papers by the management on the end of this issue. We provide you this special **concession of 450** . This concession is **only valid for 7 hours** from now only.

If you want to avail this concession, we suggest you to pay the registration fee in defined time.

[Pay 1750 Now](#)

Note: *There is no concession on certification fee.*

In case of any query, [reply to this email](#) or call our counselors at [+91-8195072273](#)

Track Status	
Paper ID	V7I6-1413 (v7i6-1413)
Title	Brain Tumor Detection
Author(s)	Bharat Devgn, Sanyam Sharma
Status	Accepted
Tracking ID	v7i6-1413-bha (Track Now)



Follow us on [f](#) [t](#) [in](#)