Brain Tumor Detection

**by**

# **Ankit Kumar Biswal - 21BEC1167**

A Mini-Project report submitted

to

# Dr. Sheena Christabel Pravin

**SCHOOL OF ELECTRONICS ENGINEERING**

in partial fulfilment of the requirements for the course of

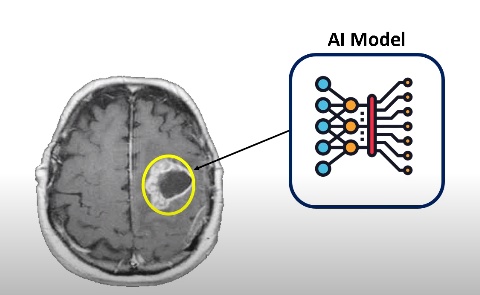
**BECE202L – Signals and Systems**

**in**

**B.TECH - Electronics and Communications Engineering**

**Vandalur – Kelambakkam Road, Chennai – 600127**

**OCTOBER 2022**

****

**ABSTRACT**

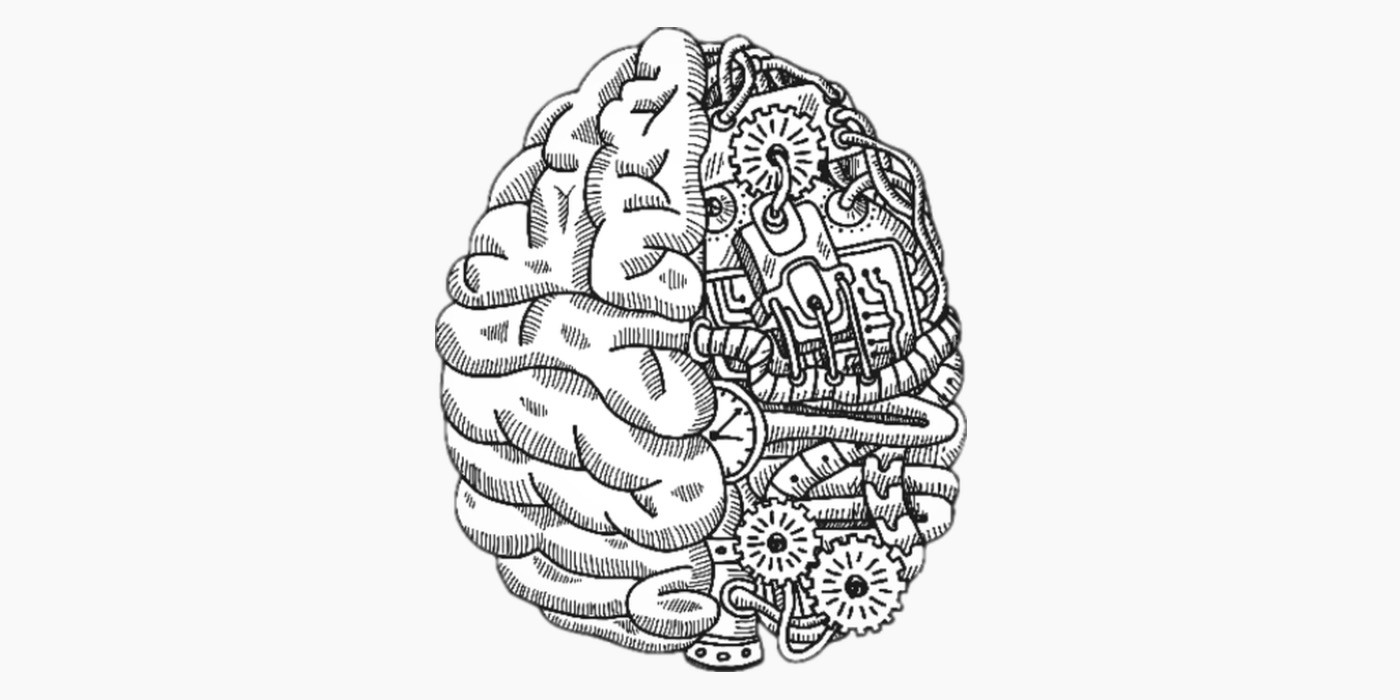
A Brain Tumor is considered as one of the aggressive diseases, among children and adults. Brain Tumors account for 85 to 90 percent of all primary Central Nervous System(CNS) Tumors. Every year, around 11,700 people are diagnosed with a brain tumor. The 5-year survival rate for people with a cancerous brain or CNS Tumor is approximately 34 percent for men and36 percent for women. Brain Tumors are classified as: Benign Tumor, Malignant Tumor, Pituitary Tumor, etc. Proper treatment, planning, and accurate diagnostics should be implemented to improve the life expectancy of the patients. The best technique to detect brain Tumors is Magnetic Resonance Imaging (MRI). A huge amount of image data is generated through the scans. These images are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain Tumors and their properties.  
Application of automated classification techniques using Machine Learning(ML) and Artificial Intelligence(AI)has consistently shown higher accuracy than manual classification. Hence, proposing a system performing detection and classification by using Deep Learning Algorithms using Convolution Neural Network (CNN), Artificial Neural Network (ANN), and Transfer Learning (TL) would be helpful to doctors all around the world.

**Context:-**

Brain Tumors are complex. There are a lot of abnormalities in the sizes and location of the brain Tumor(s). This makes it really difficult for complete understanding of the nature of the Tumor. Also, a professional Neurosurgeon is required for MRI analysis. Often times in developing countries the lack of skilful doctors and lack of knowledge about Tumors makes it really challenging and time-consuming to generate reports from MRI’. So an automated system on Cloud can solve this problem

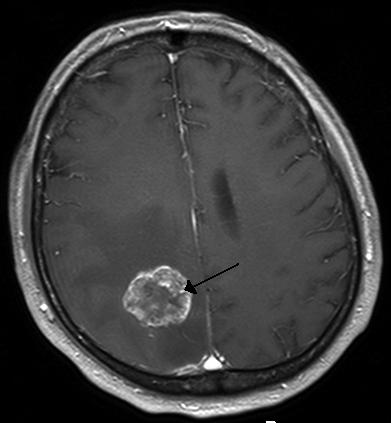
**Introduction:-**

In this notebook, I've used CNN to perform Image Classification on the Brain Tumor dataset.  
Since this dataset is small, if we train a neural network to it, it won't really give us a good result.  
Therefore, I'm going to use the concept of Transfer Learning to train the model to get really accurate results.



**What is Brain Tumor?**

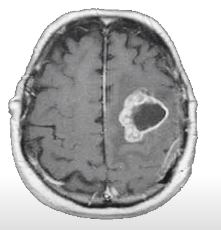
A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain metastasis tumors. All types of brain tumors may produce symptoms that vary depending on the part of the brain involved. These symptoms may include headaches, seizures, problems with vision, vomiting and mental changes. The headache is classically worse in the morning and goes away with vomiting. Other symptoms may include difficulty walking, speaking or with sensations. As the disease progresses, unconsciousness may occur.



Brain metastasis in the right cerebral hemisphere from lung cancer, shown on magnetic resonance imaging.

**Why only Brain Tumor Detection?**

* Brain Tumour is the accumulation, or mass or growth of abnormal cells in the brain.
* There are basically two types of brain tumours malignant and benign Malignant brain tumors are relatively rare ,accounting for only 1-2%of all types of cancer in adults but having lower survival rate
* If not treated at an initial phase, it may lead to death.
* According to research studies it is found that ,the incidence of most malignant brain tumors is significantly lower in East Asia, Southeast Asia ,and India.
* The highest incidences have been found in Europe ,Canada, the United States ,and Australia.

****

**In this project we will build multi-class classification based CNN model for classifying 3 different types of brain tumours and normal cases i.e.,**

**no tumour**

* Glioma
* Meningioma
* Pituitary
* No Tumour

**For building the deep learning model we have used the Brain Tumour MRI Dataset available on Kaggle**

**The distribution of images in training data for each class are as follows:**

* Glioma(1321)
* Meningioma(1339)
* Pituitary(1457)
* No Tumour(1595)

**For validating the model we will test on Testing data(unseen):**

* Glioma(300)
* Meningioma(306)
* Pituitary(300)
* No Tumour(405)

**What is Glioma:-**

According to John Hopkins Medicine ,Glioma is a common type of Tumor

originating in the brain and about 33 percent of all brain Tumors are

gliomas ,which originate in the gluey supportive cells(glial cells)that

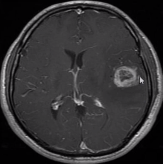
surround and support neurons in the brain.

A glioma can affect your brain function and be life-threatening depending

on its location and rate of growth.

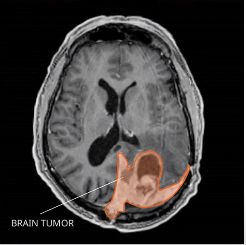
Three types of glial cells can produce Tumors.

* **Astrocytomas** ,including astrocytoma , anaplastic astrocytoma and glioblastoma
* **Ependymomas**, including anaplastic ependymoma, myxopapillary ependymoma and subependymoma
* **Oligodendrogliomas**, including oligodendroglioma, anaplastic oligodendroglioma and anaplastic oligoastrocytoma



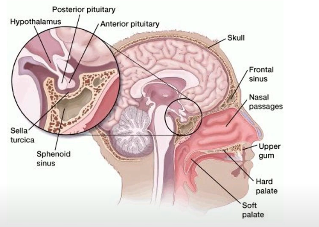
**What is Meningioma:-**

* A meningioma is a primary central nervous system(CNS) tumor.This means it begins in the brain or spinal cord.
* Specifically ,the tumor forms on the three layers of membranes that are called meninges.
* These tumors are often slow-growing. As many as 90%are benign(not cancerous).
* Often ,meningiomas cause no symptoms and require no immediate treatment.
* But the growth of benign meningiomas can cause serious problems .In some cases, such growth can be fatal.



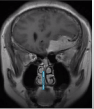
**What is Pituitary:-**

* A pituitary tumor is a tumor that forms in the pituitary gland near the brain that can cause changes in hormone levels in the body.
* Most pituitary tumors are noncancerous (benign) growths (adenomas).Adenomas remain in your pituitary gland or surrounding tissues and don't spread to other parts of your body.
* Still benign pituitary tumors can cause major health problems because they are close to the brain, may invade nearby tissues(like the skull or the sinuses)
* Pituitary cancers(called pituitary carcinomas)are very rare.



**Image Pre-Processing:-**

* Because many of the images are of different size,we have resized them to one size i.e.,(200,200) A
* Other major issue with the MRI images was they have lot of noise .So therefore, we have cropped the images so that we can only focus image for training.

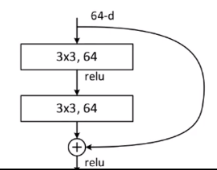
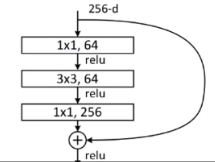
   

**Step:1 Step:2 Step:3 Step:4**

* **Step1:- Original Image**
* **Step2:- Find the Biggest Contour**
* **Step3:- Find the Extreme Points**
* **Step4:- Crop the image**

**Modelling:-**

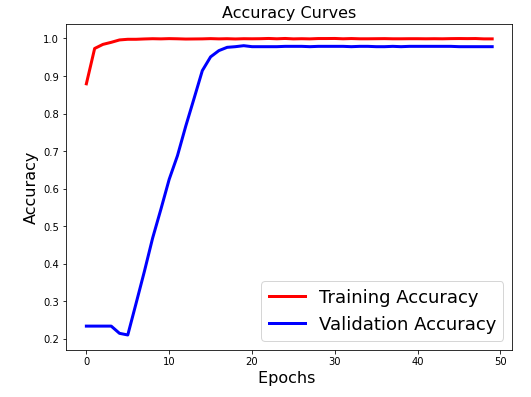
* In this project we will be using ResNet 50 pre-trained network for fine tuning brain tumour classification tasks
* When working with deep convolutional neural networks to solve a problem related to computer vision, machine learning experts engage in stacking more layers. More layers helps in learning complex patterns and improving accuracy levels but after a certain point adding more layers resulting in performance degradation resulting in over fitting. ResNet was created with the aim of tackling this exact problem.
* It has skip connections which work in two ways.
* Firstly ,they alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through.
* In addition ,they enable the model to learn an identity function. This ensures that the higher layers of the model do not perform any worse than the lower layers.

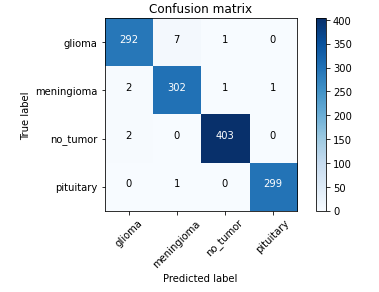
 

**Results and Discussion:-**

In this work, efficient automatic brain tumor detection is performed by using convolution neural network. Simulation is performed by using python language. The accuracy is calculated and compared with the all other state of arts methods. The training accuracy, validation accuracy and validation loss are calculated to find the efficiency of proposed brain tumor classification scheme. In the existing technique, the Support Vector Machine (SVM) based classification is performed for brain tumor detection. It needs feature extraction output. Based on feature value, the classification output is generated and accuracy is calculated. The computation time is high and accuracy is low in SVM based tumor and non-tumor detection.

In the proposed CNN based classification doesn’t require feature extraction steps separately. The feature value is taken from CNN itself. shows the classified result of Tumor and Non-tumor brain image. Hence the complexity and computation time is low and accuracy is high. The output of brain tumor classification accuracy is given in figure below Finally, the classification results as Tumor brain or non-tumor brain based on the probability score value. The normal brain image has the lowest probability score. Tumor brain has highest probability score value, when compared to normal and tumor brain.





**Code :-**

# **Brain Tumor classification using Resnet 50:-**

**import** tensorflow

**from** PIL **import** Image

**import** glob

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator,load\_img, save\_img, img\_to\_array

**from** tensorflow.keras.applications.vgg16 **import** VGG16, preprocess\_input

**from** tensorflow.keras.preprocessing **import** image

**from** tensorflow.keras **import** backend **as** K

**from** tensorflow.keras.models **import** Model, Sequential

**from** tensorflow.keras.layers **import** Input, Dense, Flatten, Dropout, BatchNormalization,Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation,GlobalAveragePooling2D

**from** tensorflow.keras.optimizers **import** Adam

**from** sklearn.model\_selection **import** train\_test\_split

**from** tensorflow.keras.callbacks **import** ModelCheckpoint, ReduceLROnPlateau,EarlyStopping

**from** tensorflow.keras.applications.imagenet\_utils **import** preprocess\_input

**from** sklearn.metrics **import** classification\_report,accuracy\_score

**from** sklearn.metrics **import** confusion\_matrix

**import** matplotlib.pyplot **as** plt

**from** sklearn.utils **import** shuffle

**import** numpy **as** np

**from** tqdm **import** tqdm

**from** sklearn.utils **import** shuffle

**import** cv2

**import** os

**import** shutil

**import** itertools

**import** imutils

**from** sklearn.model\_selection **import** StratifiedKFold

**import** random

**from** tensorflow.keras **import** layers

**from** google.colab **import** drive

drive**.**mount('/content/drive')

**!**ls drive

# **Sample Images:-**

data\_dir **=** ('/content/drive/MyDrive/brain\_tumour/Training')

categories **=** ['glioma', 'meningioma', 'notumor', 'pituitary']

plt**.**figure(figsize**=**(20, 16))

images\_path **=** ['/glioma/Tr-gl\_0010.jpg', '/meningioma/Tr-meTr\_0000.jpg', '/notumor/Tr-noTr\_0000.jpg', '/pituitary/Tr-piTr\_0000.jpg']

**for** i **in** range(4):

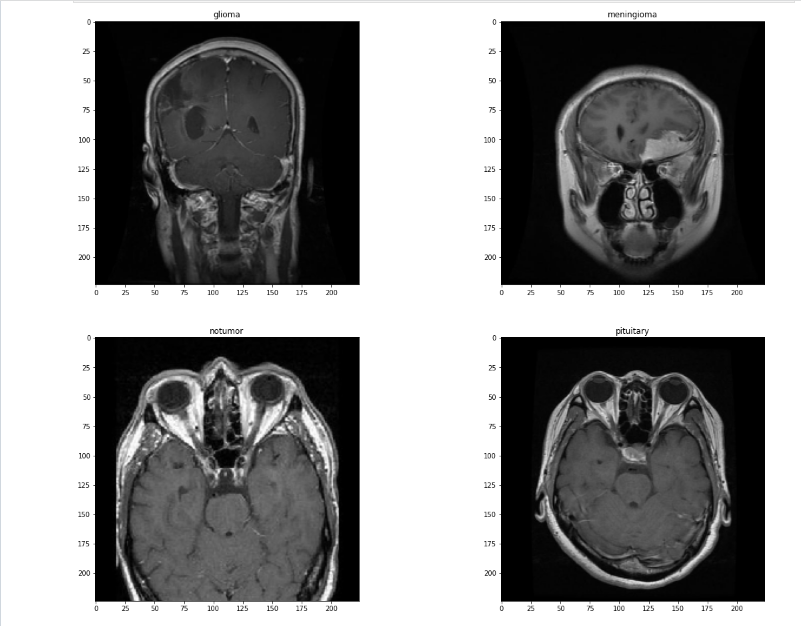
ax **=** plt**.**subplot(2, 2, i **+** 1)

img **=** cv2**.**imread(data\_dir **+** images\_path[i])

img **=** cv2**.**resize(img, (224, 224))

plt**.**imshow(img)

plt**.**title(categories[i])



# **Cropping Images Demo:-**

**def** crop\_img(img):

"""

Finds the extreme points on the image and crops the rectangular out of them

"""

gray **=** cv2**.**cvtColor(img, cv2**.**COLOR\_RGB2GRAY)

gray **=** cv2**.**GaussianBlur(gray, (3, 3), 0)

*# threshold the image, then perform a series of erosions +*

*# dilations to remove any small regions of noise*

thresh **=** cv2**.**threshold(gray, 45, 255, cv2**.**THRESH\_BINARY)[1]

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

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

*# find contours in thresholded image, then grab the largest one*

cnts **=** cv2**.**findContours(thresh**.**copy(), cv2**.**RETR\_EXTERNAL, cv2**.**CHAIN\_APPROX\_SIMPLE)

cnts **=** imutils**.**grab\_contours(cnts)

c **=** max(cnts, key**=**cv2**.**contourArea)

*# find the extreme points*

extLeft **=** tuple(c[c[:, :, 0]**.**argmin()][0])

extRight **=** tuple(c[c[:, :, 0]**.**argmax()][0])

extTop **=** tuple(c[c[:, :, 1]**.**argmin()][0])

extBot **=** tuple(c[c[:, :, 1]**.**argmax()][0])

ADD\_PIXELS **=** 0

new\_img **=** img[extTop[1]**-**ADD\_PIXELS:extBot[1]**+**ADD\_PIXELS, extLeft[0]**-**ADD\_PIXELS:extRight[0]**+**ADD\_PIXELS]**.**copy()

**return** new\_img

img **=** cv2**.**imread('/content/drive/My Drive/brain\_tumour/Training/meningioma/Tr-meTr\_0000.jpg')

img **=** cv2**.**resize(

img,

dsize**=**(224,224),

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()

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)

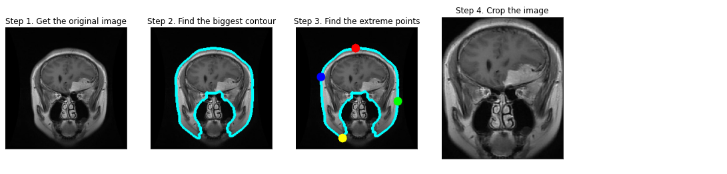
plt**.**imshow(new\_img)

plt**.**xticks([])

plt**.**yticks([])

plt**.**title('Step 4. Crop the image')

plt**.**show()



labels **=** ['glioma', 'meningioma', 'notumor', 'pituitary']

x\_train **=** [] *# training images.*

y\_train **=** [] *# training labels.*

x\_test **=** [] *# testing images.*

y\_test **=** [] *# testing labels.*

image\_size **=** 200

**for** label **in** labels:

trainPath **=** os**.**path**.**join('/content/drive/My Drive/brain\_tumour/cropped/Training',label)

**for** file **in** tqdm(os**.**listdir(trainPath)):

image **=** cv2**.**imread(os**.**path**.**join(trainPath, file),0) *# load images in gray.*

image **=** cv2**.**bilateralFilter(image, 2, 50, 50) *# remove images noise.*

image **=** cv2**.**applyColorMap(image, cv2**.**COLORMAP\_BONE) *# produce a pseudocolored image.*

image **=** cv2**.**resize(image, (image\_size, image\_size)) *# resize images into 150\*150.*

x\_train**.**append(image)

y\_train**.**append(labels**.**index(label))

testPath **=** os**.**path**.**join('/content/drive/My Drive/brain\_tumour/cropped/Testing',label)

**for** file **in** tqdm(os**.**listdir(testPath)):

image **=** cv2**.**imread(os**.**path**.**join(testPath, file),0)

image **=** cv2**.**bilateralFilter(image, 2, 50, 50)

image **=** cv2**.**applyColorMap(image, cv2**.**COLORMAP\_BONE)

image **=** cv2**.**resize(image, (image\_size, image\_size))

x\_test**.**append(image)

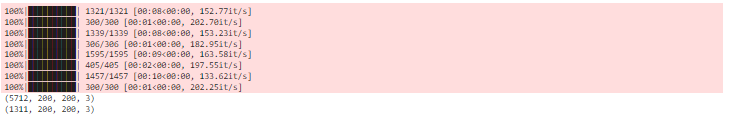
y\_test**.**append(labels**.**index(label))

x\_train **=** np**.**array(x\_train) **/** 255.0 *# normalize Images into range 0 to 1.*

x\_test **=** np**.**array(x\_test) **/** 255.0

print(x\_train**.**shape)

print(x\_test**.**shape)



images **=** [x\_train[i] **for** i **in** range(15)]

fig, axes **=** plt**.**subplots(3, 5, figsize **=** (10, 10))

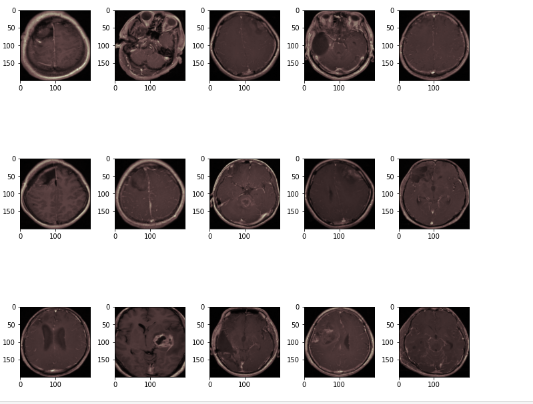
axes **=** axes**.**flatten()

**for** img, ax **in** zip(images, axes):

ax**.**imshow(img)

plt**.**tight\_layout()

plt**.**show()



x\_train, y\_train **=** shuffle(x\_train,y\_train, random\_state**=**42)

y\_train **=** tensorflow**.**keras**.**utils**.**to\_categorical(y\_train) *#One Hot Encoding on the labels*

y\_test **=** tensorflow**.**keras**.**utils**.**to\_categorical(y\_test)

x\_train, x\_val, y\_train, y\_val **=** train\_test\_split(x\_train, y\_train, test\_size**=**0.2, random\_state**=**42) *#Dividing the dataset into Training and Validation sets.*

print(x\_val**.**shape)



# **Image Augmentation :-**

*# 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,

zoom\_range**=**0.2,

shear\_range**=**0.05,

brightness\_range**=**[0.1, 1.5],

horizontal\_flip**=True**,

vertical\_flip**=True**

)

os**.**mkdir('preview\_2')

x **=** x\_train[0]

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

i **=** 0

**for** batch **in** demo\_datagen**.**flow(x, batch\_size**=**1, save\_to\_dir**=**'preview\_2', save\_prefix**=**'aug\_img', save\_format**=**'jpg'):

i **+=** 1

**if** i **>** 20:

**break**

plt**.**imshow(x[0])

plt**.**xticks([])

plt**.**yticks([])

plt**.**title('Original Image')

plt**.**show()

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

i **=** 1

**for** img **in** os**.**listdir('preview\_2/'):

img **=** cv2**.**cv2**.**imread('preview\_2/' **+** img)

img **=** cv2**.**cvtColor(img, cv2**.**COLOR\_BGR2RGB)

plt**.**subplot(3,7,i)

plt**.**imshow(img)

plt**.**xticks([])

plt**.**yticks([])

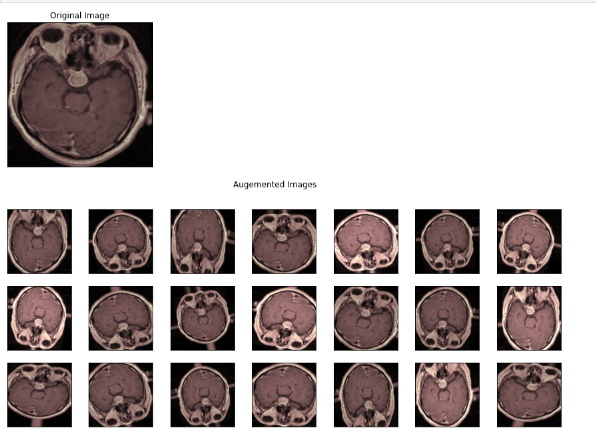
i **+=** 1

**if** i **>** 3**\***7:

**break**

plt**.**suptitle('Augemented Images')

plt**.**show()



*# ImageDataGenerator transforms each image in the batch by a series of random translations, rotations, etc.*

datagen **=** ImageDataGenerator(

rotation\_range**=**10,

width\_shift\_range**=**0.05,

height\_shift\_range**=**0.05,

horizontal\_flip**=True**)

*# After you have created and configured your ImageDataGenerator, you must fit it on your data.*

datagen**.**fit(x\_train)

**from** tensorflow.keras.applications.resnet **import** ResNet50

IMG\_SIZE**=**(200,200)

conv\_base **=** ResNet50(

include\_top**=False**,

input\_shape**=**IMG\_SIZE **+** (3,),

weights**=**'imagenet')

**for** layer **in** conv\_base**.**layers:

layer**.**trainable **=** **True**



model **=** conv\_base**.**output

model **=** GlobalAveragePooling2D()(model)

model **=** Dropout(0.4)(model)

model **=** Dense(4, activation**=**"softmax")(model)

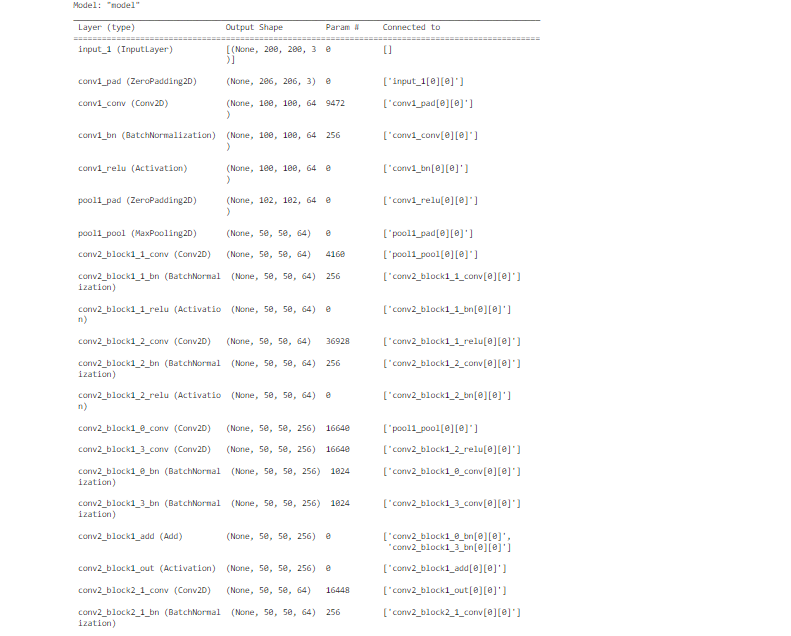
model **=** Model(inputs**=** conv\_base**.**input, outputs**=** model)

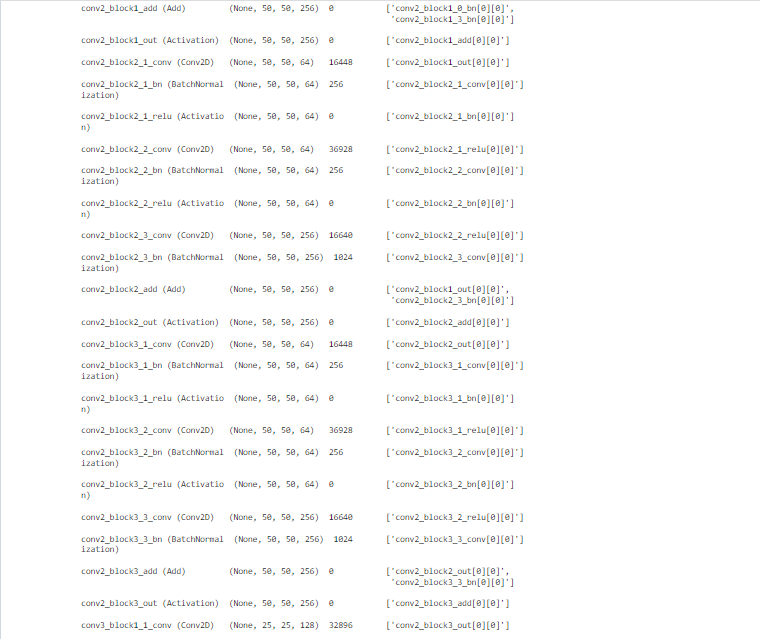
*#compile our model.*

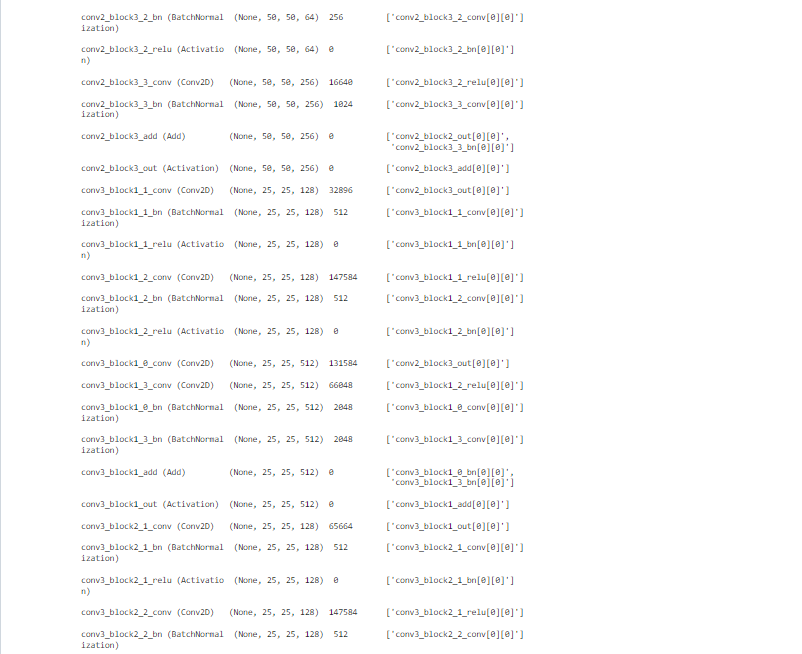
adam **=** Adam(learning\_rate**=**0.0001)

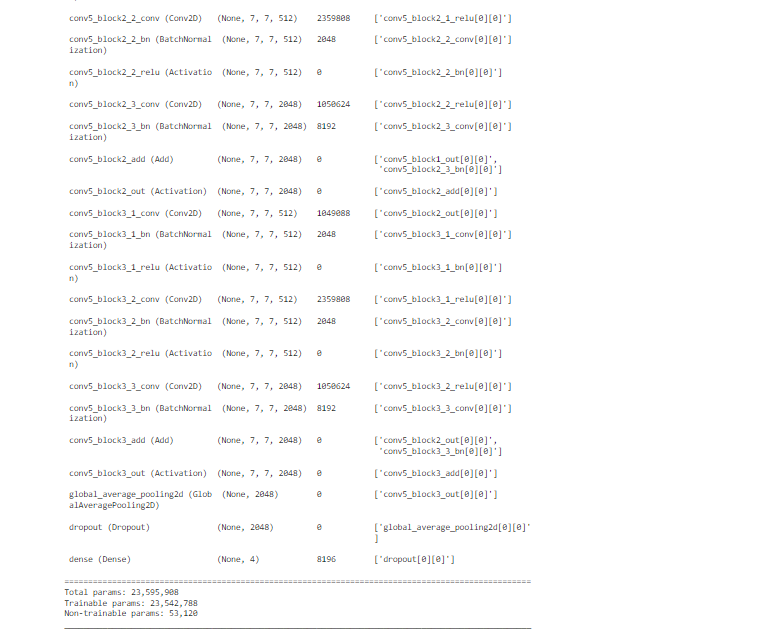
model**.**compile(optimizer**=**adam, loss **=** 'categorical\_crossentropy', metrics**=**['accuracy'])

model**.**summary()









callbacks **=** [ModelCheckpoint('.mdl\_wts.hdf5', monitor**=**'val\_loss',mode**=**'min',verbose**=**1, save\_best\_only**=True**),

ReduceLROnPlateau(monitor**=**'val\_loss', factor**=**0.3, patience**=**2, verbose**=**1, mode**=**'min', min\_lr**=**0.00000000001)]

train\_len **=** len(x\_train)

val\_len **=** len(x\_val)

print("-----------Training Data length-----------------")

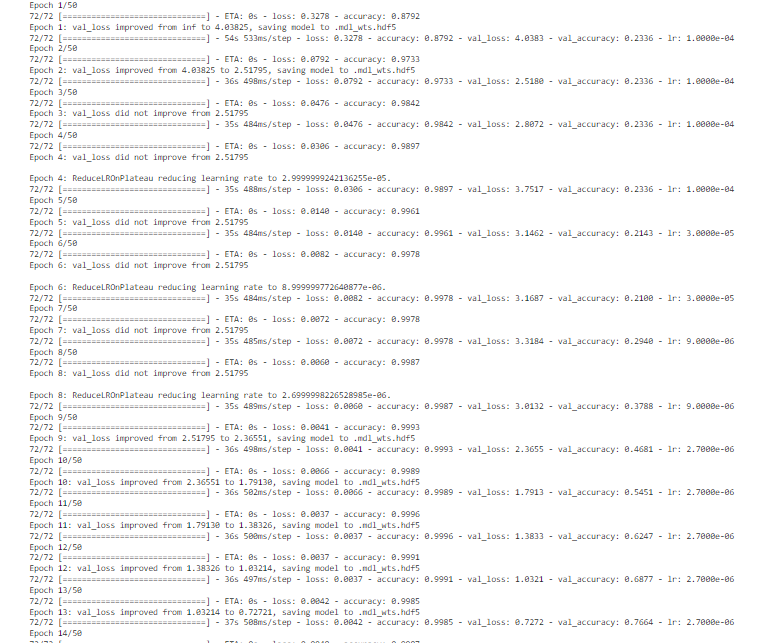
print(train\_len)

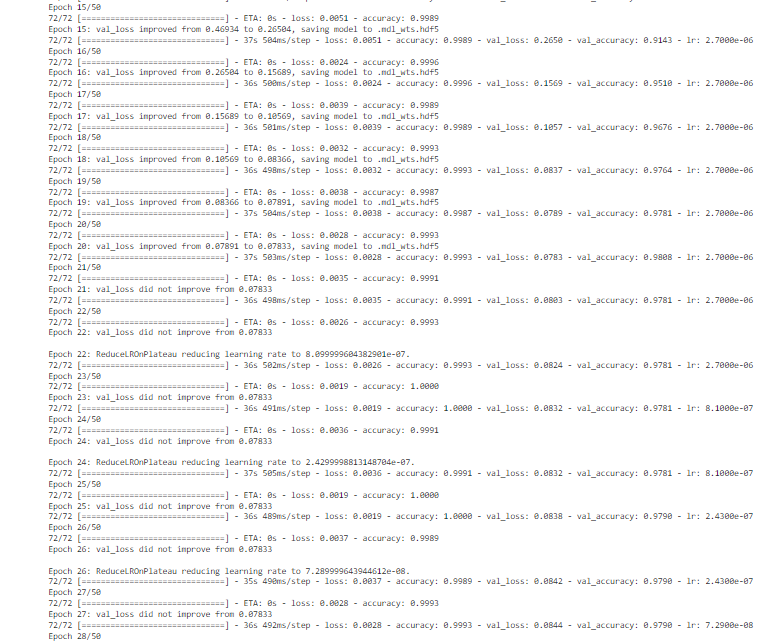
print("-----------Validation Data length-----------------")

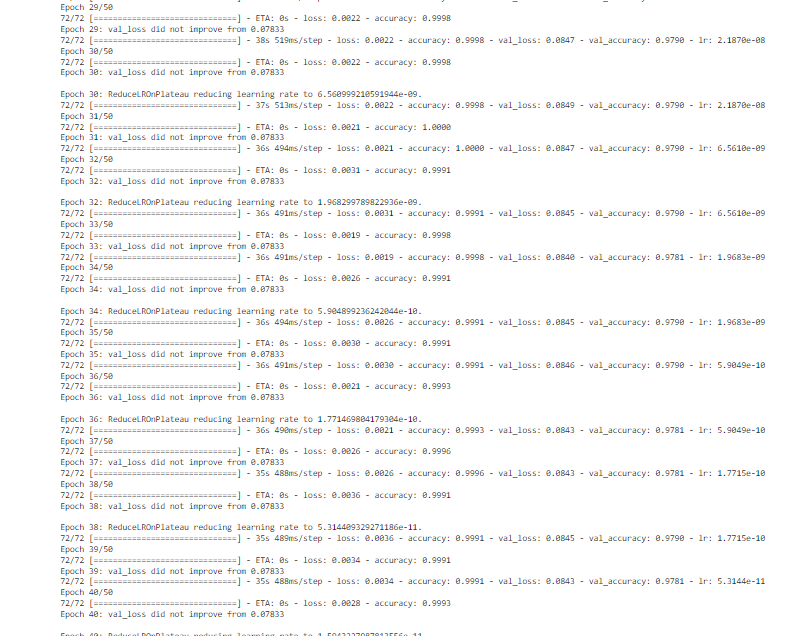
print(val\_len)

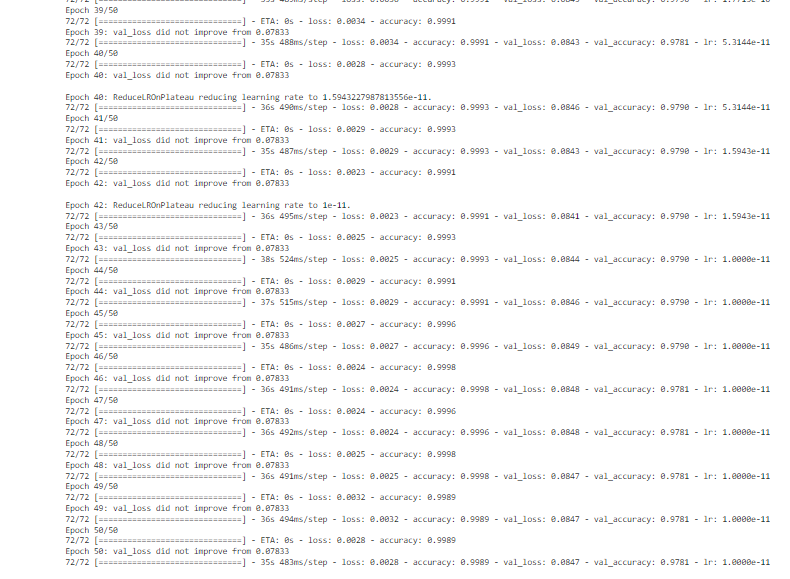


history **=** model**.**fit(datagen**.**flow(x\_train, y\_train, batch\_size**=**64),validation\_data **=** (x\_val,y\_val),epochs **=** 50,callbacks **=** callbacks)









# **Learning Curves :-**

*#Plot the Loss Curves*

plt**.**figure(figsize**=**[8,6])

plt**.**plot(history**.**history['loss'],'r',linewidth**=**3.0)

plt**.**plot(history**.**history['val\_loss'],'b',linewidth**=**3.0)

plt**.**legend(['Training loss', 'Validation Loss'],fontsize**=**18)

plt**.**xlabel('Epochs ',fontsize**=**16)

plt**.**ylabel('Loss',fontsize**=**16)

plt**.**title('Loss Curves',fontsize**=**16)

plt**.**show()

*#Plot the Accuracy Curves*

plt**.**figure(figsize**=**[8,6])

plt**.**plot(history**.**history['accuracy'],'r',linewidth**=**3.0)

plt**.**plot(history**.**history['val\_accuracy'],'b',linewidth**=**3.0)

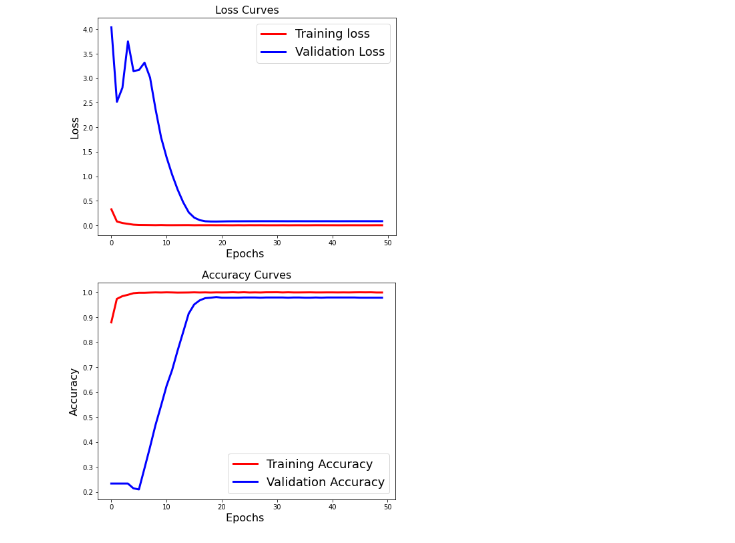
plt**.**legend(['Training Accuracy', 'Validation Accuracy'],fontsize**=**18)

plt**.**xlabel('Epochs ',fontsize**=**16)

plt**.**ylabel('Accuracy',fontsize**=**16)

plt**.**title('Accuracy Curves',fontsize**=**16)

plt**.**show()



# **Loading Model :-**

**from** tensorflow.keras.models **import** load\_model

model **=** load\_model('.mdl\_wts.hdf5')

model**.**save('/content/drive/My Drive/brain\_tumour/modelres50.h5')

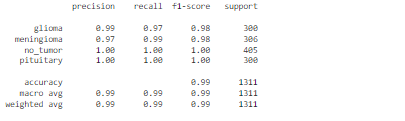
model **=** load\_model('/content/drive/My Drive/brain\_tumour/modelres50.h5')

# **Validation on Test set :-**

**import** seaborn **as** sns

predicted\_classes **=** np**.**argmax(model**.**predict(x\_test), axis **=** 1)

print(classification\_report(np**.**argmax(y\_test,axis**=**1), predicted\_classes,target\_names**=**['glioma','meningioma','no\_tumor','pituitary']))



**import** itertools

pred\_Y **=** model**.**predict(x\_test, batch\_size **=** 8, verbose **=** **True**)

**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**.**imshow(cm, interpolation**=**'nearest', cmap**=**cmap)

plt**.**title(title)

plt**.**colorbar()

tick\_marks **=** np**.**arange(len(classes))

plt**.**xticks(tick\_marks, classes, rotation**=**45)

plt**.**yticks(tick\_marks, classes)

target\_names**=**['glioma','meningioma','no\_tumor','pituitary']

**if** target\_names **is** **not** **None**:

tick\_marks **=** np**.**arange(len(target\_names))

plt**.**xticks(tick\_marks, target\_names, rotation**=**45)

plt**.**yticks(tick\_marks, target\_names)

**if** normalize:

cm **=** cm**.**astype('float') **/** cm**.**sum(axis**=**1)[:, np**.**newaxis]

thresh **=** cm**.**max() **/** 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')

*# Predict the values from the validation dataset*

Y\_pred **=** model**.**predict(x\_test, batch\_size**=**8)

*# Convert predictions classes to one hot vectors*

Y\_pred\_classes **=** np**.**argmax(pred\_Y,axis **=** 1)

*# Convert validation observations to one hot vectors*

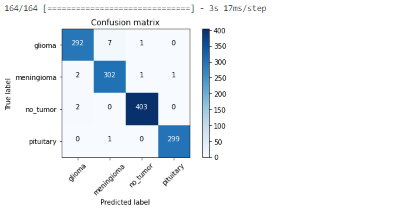
*# compute the confusion matrix*

rounded\_labels**=**np**.**argmax(y\_test, axis**=**1)

confusion\_mtx **=** confusion\_matrix(rounded\_labels, Y\_pred\_classes)

*# plot the confusion matrix*

plot\_confusion\_matrix(confusion\_mtx, classes **=** range(4))



**import** seaborn **as** sns

**import** pandas **as** pd

**from** sklearn.datasets **import** make\_classification

**from** sklearn.preprocessing **import** label\_binarize

**from** scipy **import** interp

**from** itertools **import** cycle

**import** pandas **as** pd

**%matplotlib** inline

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** sklearn.metrics **import** roc\_curve, auc

y\_test **=** np**.**array(y\_test)

n\_classes **=** 4

pred\_Y **=** model**.**predict(x\_test, batch\_size **=** 16, verbose **=** **True**)

*# Plot linewidth.*

lw **=** 2

*# Compute ROC curve and ROC area for each class*

*# Compute ROC curve and ROC area for each class*

fpr **=** dict()

tpr **=** dict()

roc\_auc **=** dict()

**for** i **in** range(n\_classes):

fpr[i], tpr[i], \_ **=** roc\_curve(y\_test[:, i], pred\_Y[:, i])

roc\_auc[i] **=** auc(fpr[i], tpr[i])

*# Compute micro-average ROC curve and ROC area*

fpr["micro"], tpr["micro"], \_ **=** roc\_curve(y\_test**.**ravel(), pred\_Y**.**ravel())

roc\_auc["micro"] **=** auc(fpr["micro"], tpr["micro"])

*# Plot of a ROC curve for a specific class*

**for** i **in** range(n\_classes):

plt**.**figure()

plt**.**plot(fpr[i], tpr[i], label**=**'ROC curve (area = %0.2f)' **%** roc\_auc[i])

plt**.**plot([0, 1], [0, 1], 'k--')

plt**.**xlim([0.0, 1.0])

plt**.**ylim([0.0, 1.05])

plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**title('Receiver operating characteristic example')

plt**.**legend(loc**=**"lower right")

plt**.**show()

*# First aggregate all false positive rates*

all\_fpr **=** np**.**unique(np**.**concatenate([fpr[i] **for** i **in** range(n\_classes)]))

*# Then interpolate all ROC curves at this points*

mean\_tpr **=** np**.**zeros\_like(all\_fpr)

**for** i **in** range(n\_classes):

mean\_tpr **+=** np**.**interp(all\_fpr, fpr[i], tpr[i])

*# Finally average it and compute AUC*

mean\_tpr **/=** n\_classes

fpr["macro"] **=** all\_fpr

tpr["macro"] **=** mean\_tpr

roc\_auc["macro"] **=** auc(fpr["macro"], tpr["macro"])

*# Plot all ROC curves*

fig **=** plt**.**figure(figsize**=**(12, 8))

plt**.**plot(fpr["micro"], tpr["micro"],

label**=**'micro-average ROC curve (area = {0:0.2f})'

''**.**format(roc\_auc["micro"]),

color**=**'deeppink', linestyle**=**':', linewidth**=**4)

plt**.**plot(fpr["macro"], tpr["macro"],

label**=**'macro-average ROC curve (area = {0:0.2f})'

''**.**format(roc\_auc["macro"]),

color**=**'navy', linestyle**=**':', linewidth**=**4)

colors **=** cycle(['aqua', 'darkorange', 'cornflowerblue'])

**for** i, color **in** zip(range(n\_classes), colors):

plt**.**plot(fpr[i], tpr[i], color**=**color, lw**=**lw,

label**=**'ROC curve of class {0} (area = {1:0.2f})'

''**.**format(i, roc\_auc[i]))

plt**.**plot([0, 1], [0, 1], 'k--', lw**=**lw)

plt**.**xlim([0.0, 1.0])

plt**.**ylim([0.0, 1.05])

plt**.**xlabel('False Positive Rate')

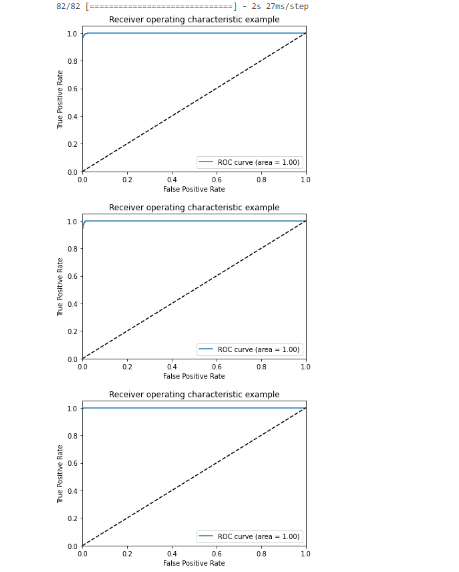
plt**.**ylabel('True Positive Rate')

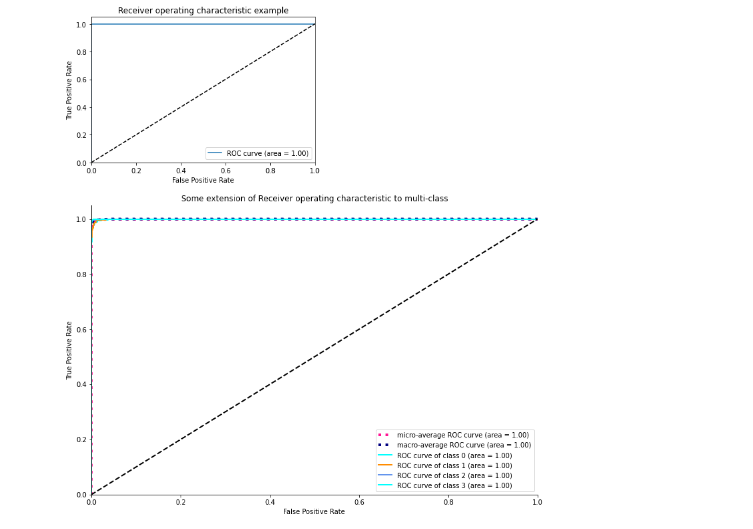
plt**.**title('Some extension of Receiver operating characteristic to multi-class')

plt**.**legend(loc**=**"lower right")

sns**.**despine()

plt**.**show()





# **Plotting sample predictions :-**

y\_hat **=** model**.**predict(x\_test)

*# define text labels*

target\_labels **=** ['glioma','meningioma','no\_tumor','pituitary']

*# plot a random sample of test images, their predicted labels, and ground truth*

fig **=** plt**.**figure(figsize**=**(20, 8))

**for** i, idx **in** enumerate(np**.**random**.**choice(x\_test**.**shape[0], size**=**12, replace**=False**)):

ax **=** fig**.**add\_subplot(4,4, i**+**1, xticks**=**[], yticks**=**[])

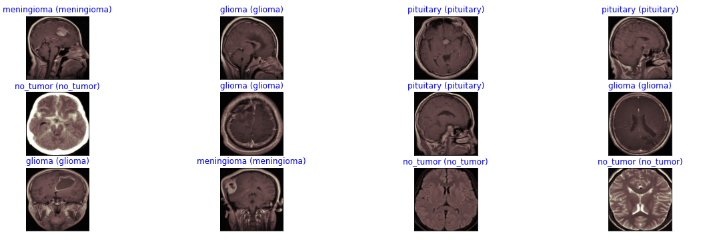
ax**.**imshow(np**.**squeeze(x\_test[idx]))

pred\_idx **=** np**.**argmax(y\_hat[idx])

true\_idx **=** np**.**argmax(y\_test[idx])

ax**.**set\_title("{} ({})"**.**format(target\_labels[pred\_idx], target\_labels[true\_idx]),

color**=**("blue" **if** pred\_idx **==** true\_idx **else** "orange"))



**Conclusion and Future Work:-**

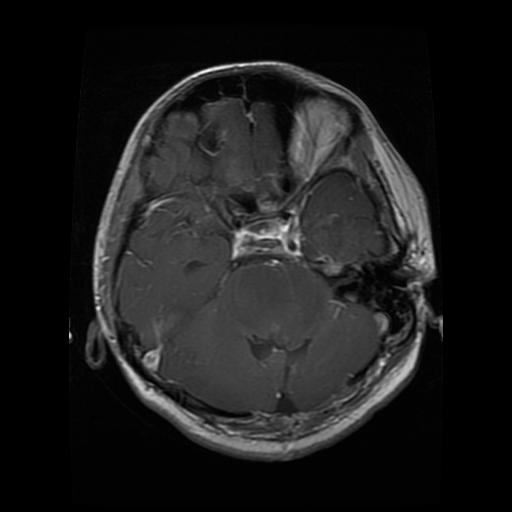
The main goal of this research work is to design efficient automatic brain tumor classification with high accuracy, performance and low complexity. In the conventional brain tumor classification is performed by using Fuzzy C Means (FCM) based segmentation, texture and shape feature extraction and SVM and DNN based classification are carried out. The complexity is low. But the computation time is high meanwhile accuracy is low. Further to improve the accuracy and to reduce the computation time, a convolution neural network based classification is introduced in the proposed scheme. Also the classification results are given as tumor or normal brain images. CNN is one of the deep learning methods, which contains sequence of feed forward layers. Also python language is used for implementation. Image net database is used for classification. It is one of the pre-trained models. So the training is performed for only final layer. Also raw pixel value with depth, width and height feature value are extracted from CNN. Finally, the Gradient decent based loss function is applied to achieve high accuracy. The training accuracy, validation accuracy and validation loss are calculated. The training accuracy is 97.5%. Similarly, the validation accuracy is high and validation loss is very low.

**Working Model Brain Tumor Detection Web app:-**

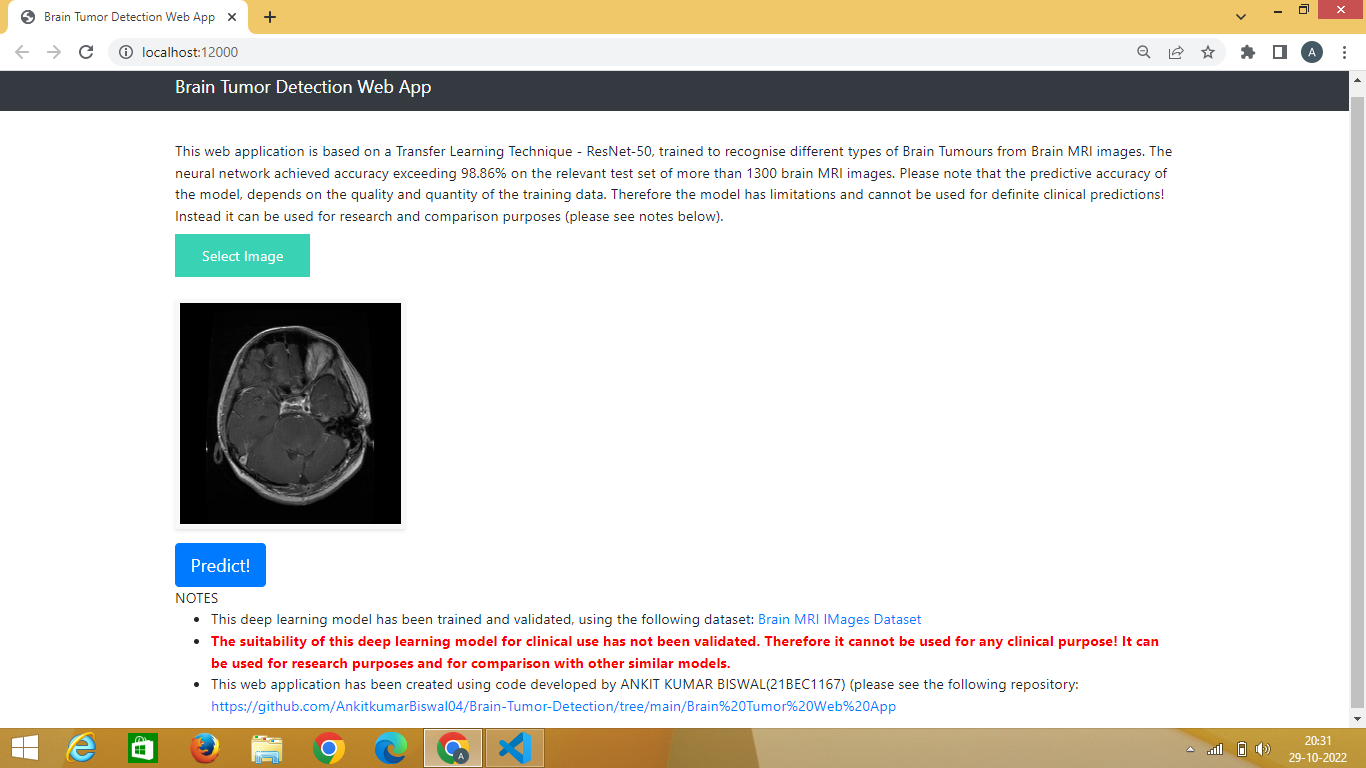
This web application is based on a Transfer Learning Technique - ResNet-50, trained to recognise different types of Brain Tumours from Brain MRI images. The neural network achieved accuracy exceeding 98.86% on the relevant test set of more than 1300 brain MRI images. Please note that the predictive accuracy of the model, depends on the quality and quantity of the training data. Therefore the model has limitations and cannot be used for definite clinical predictions! Instead it can be used for research and comparison purposes.

**Working on different Tumor cases:-**

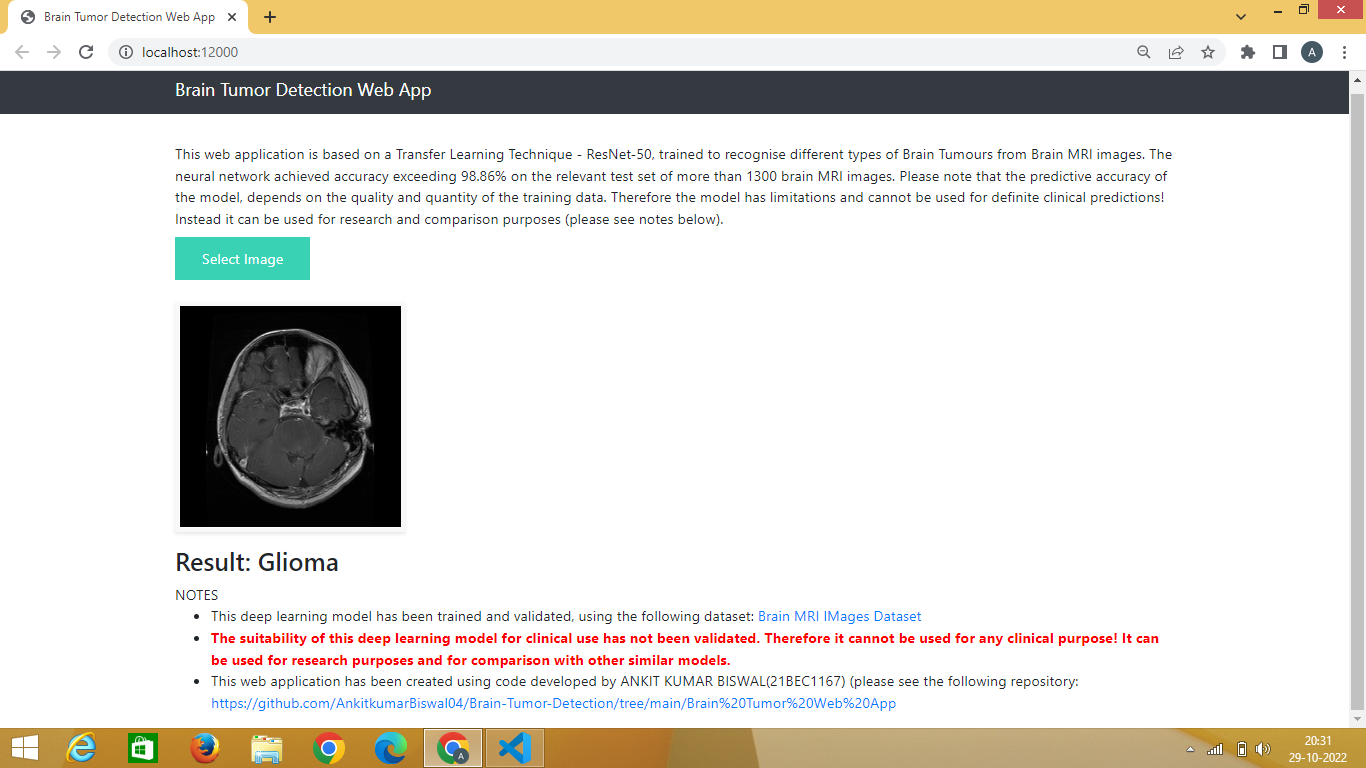
**This is for Glioma Case :-**

****

**Selecting the image :-**

****

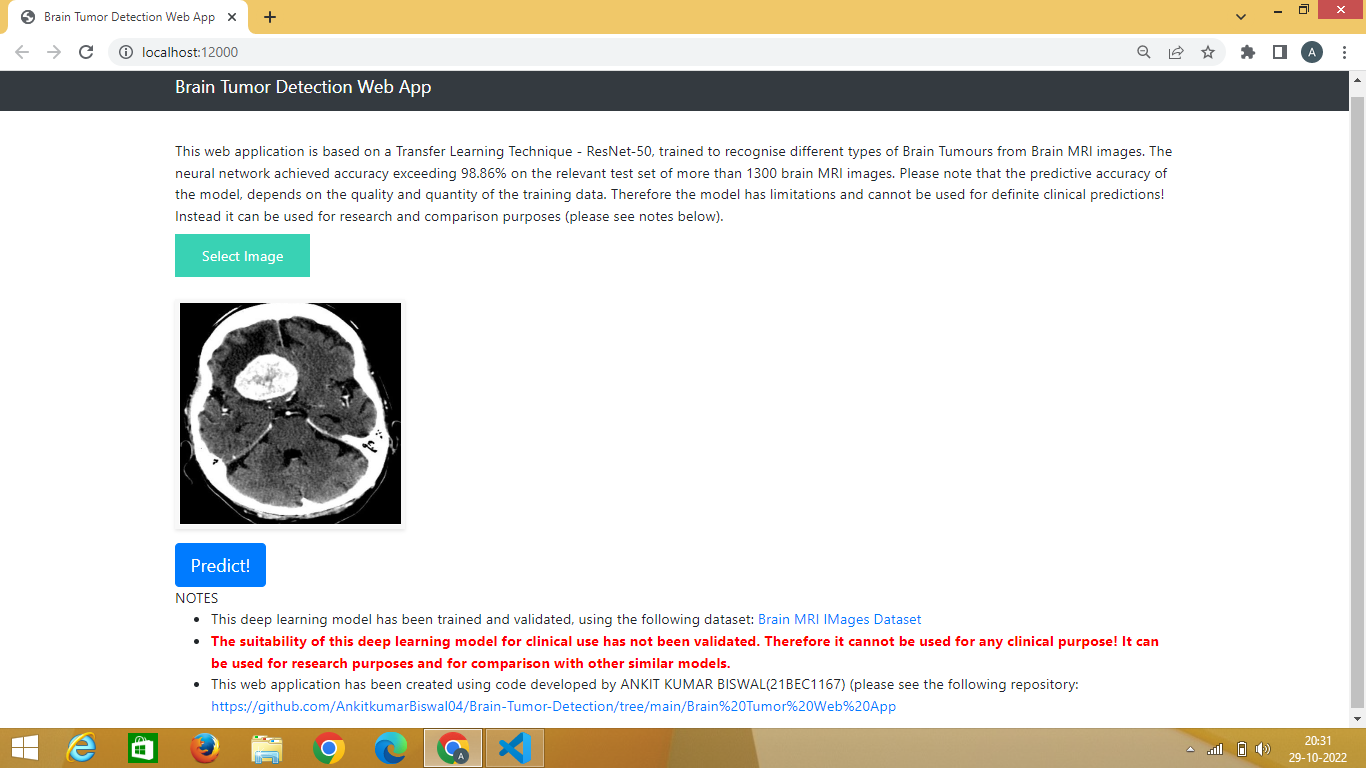
**After selecting the image the predicted result is :-**

****

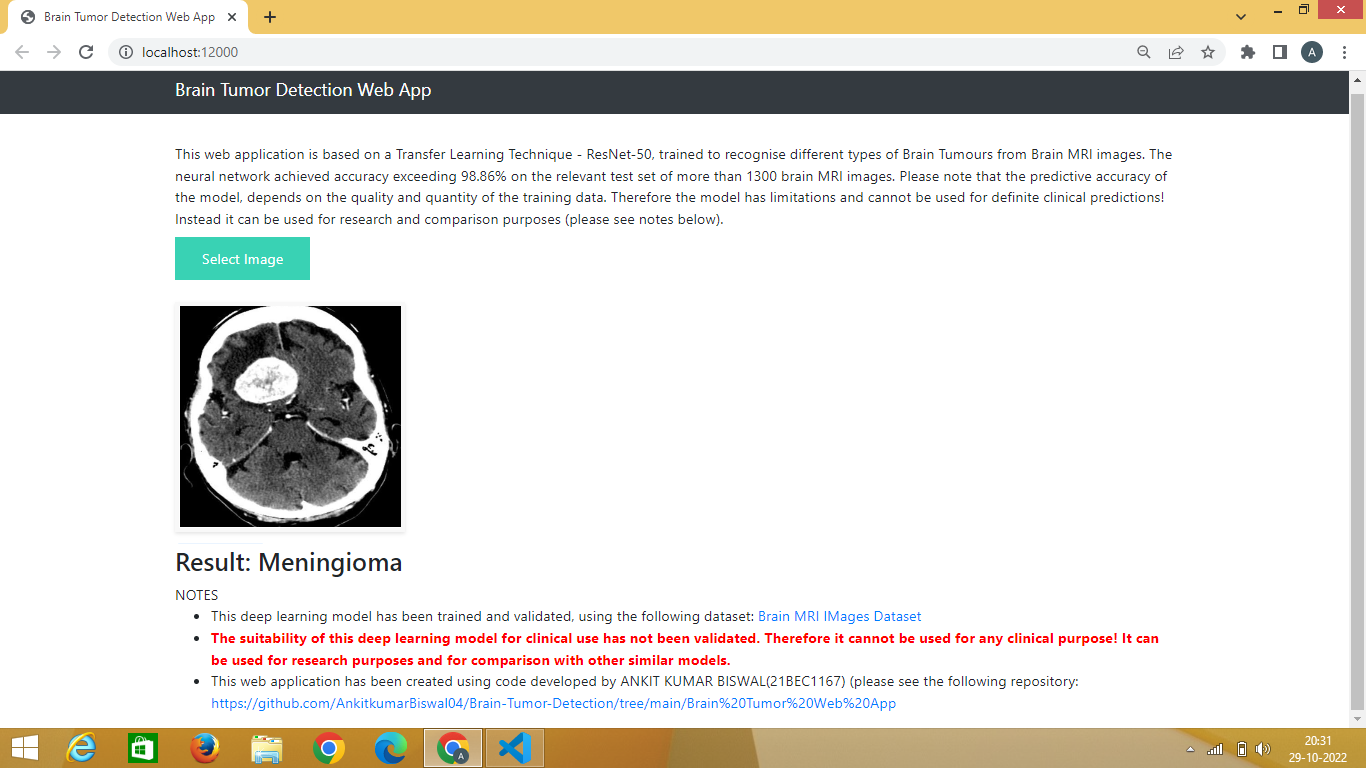
**This is for Meningioma case :-**

****

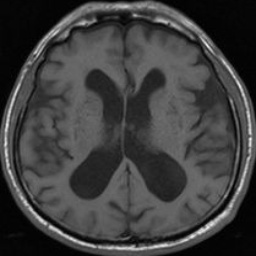
**Selecting the image :-**

****

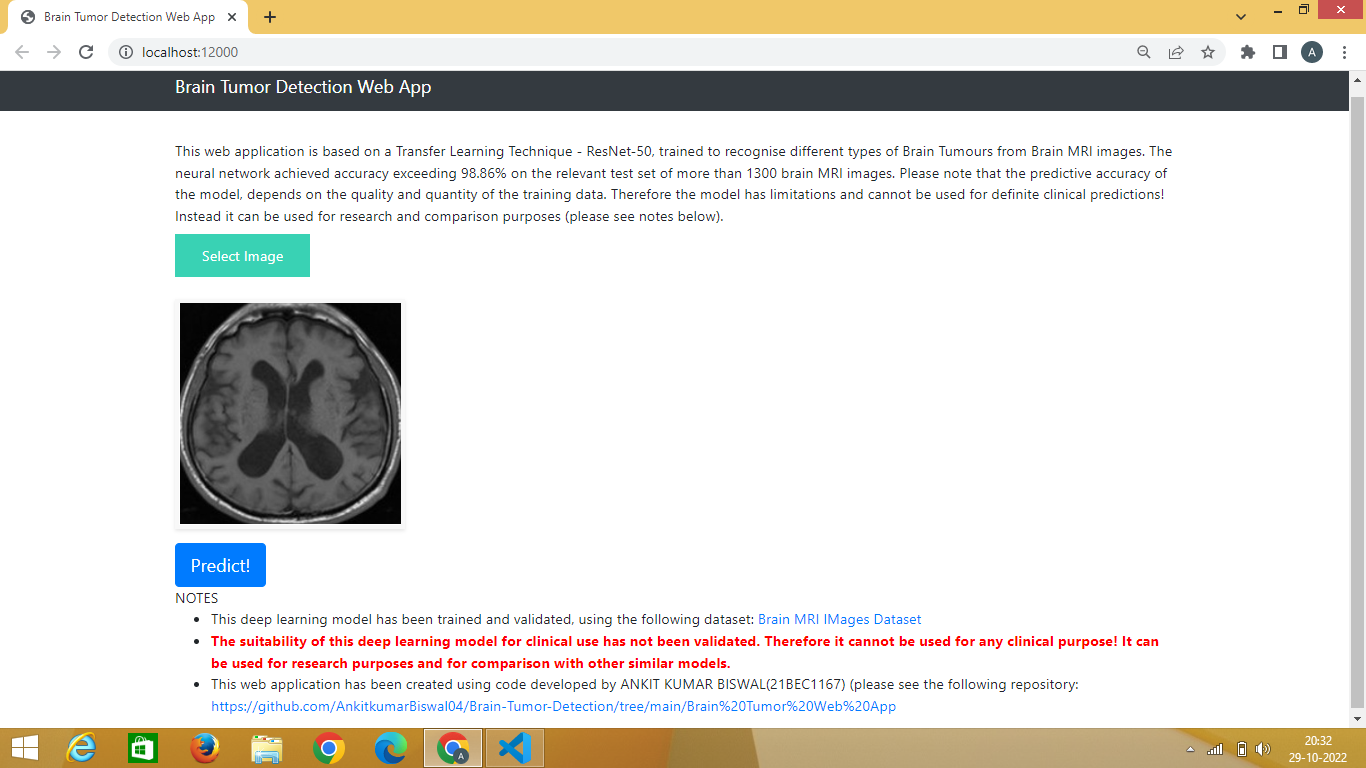
**After selecting the image the predicted result is :-**

****

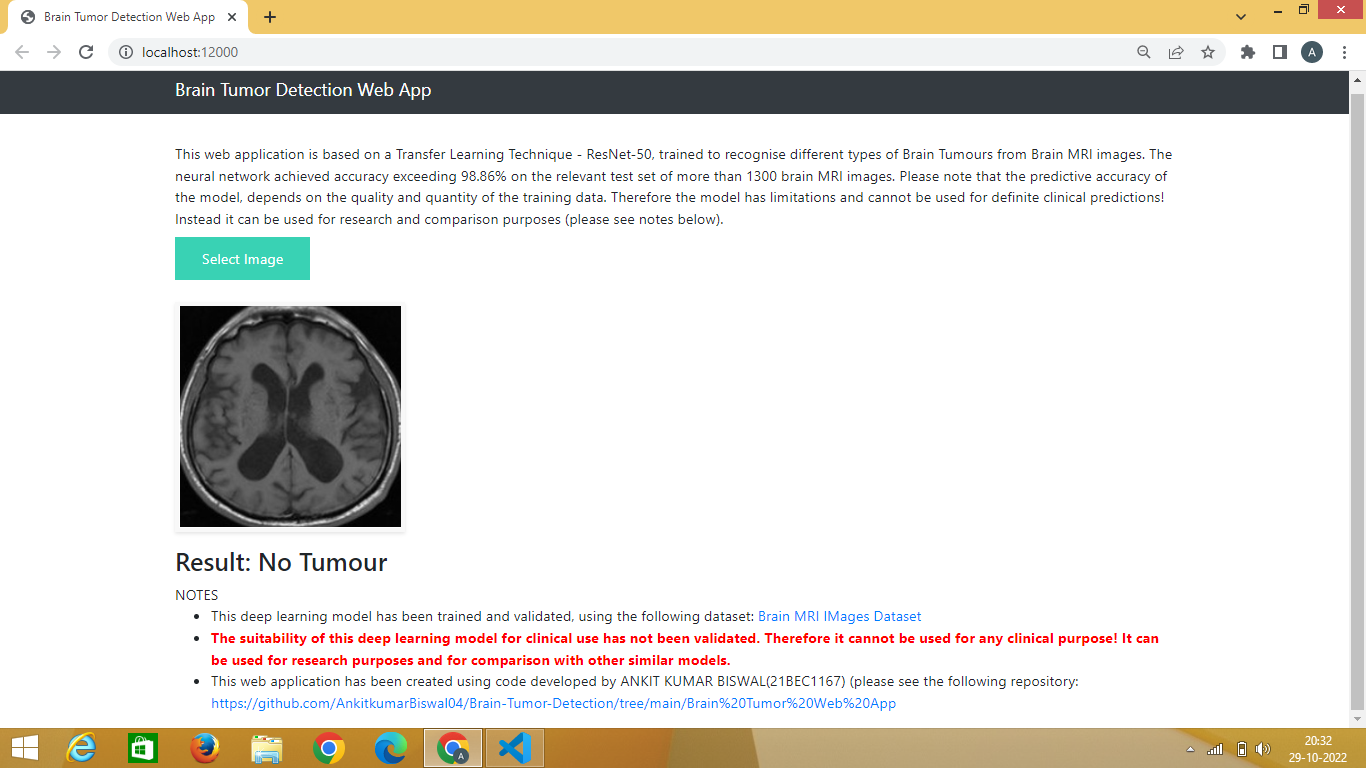
**This is for No Tumor case :-**

****

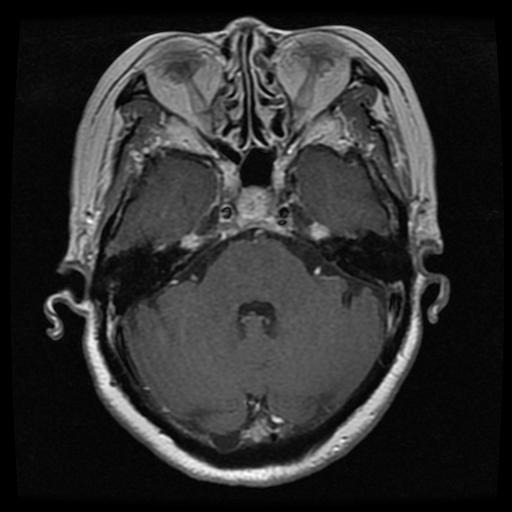
**Selecting the image :-**

****

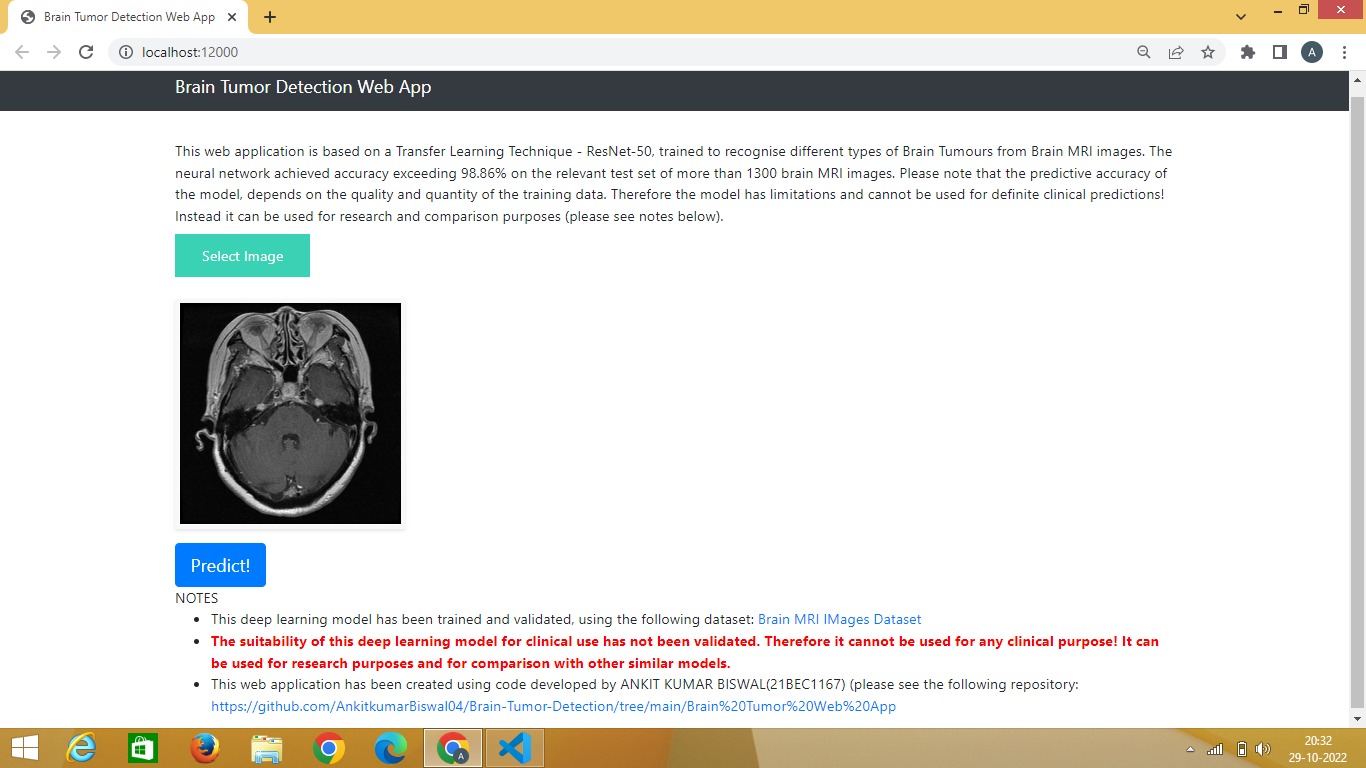
**After selecting the image the predicted result is :-**

****

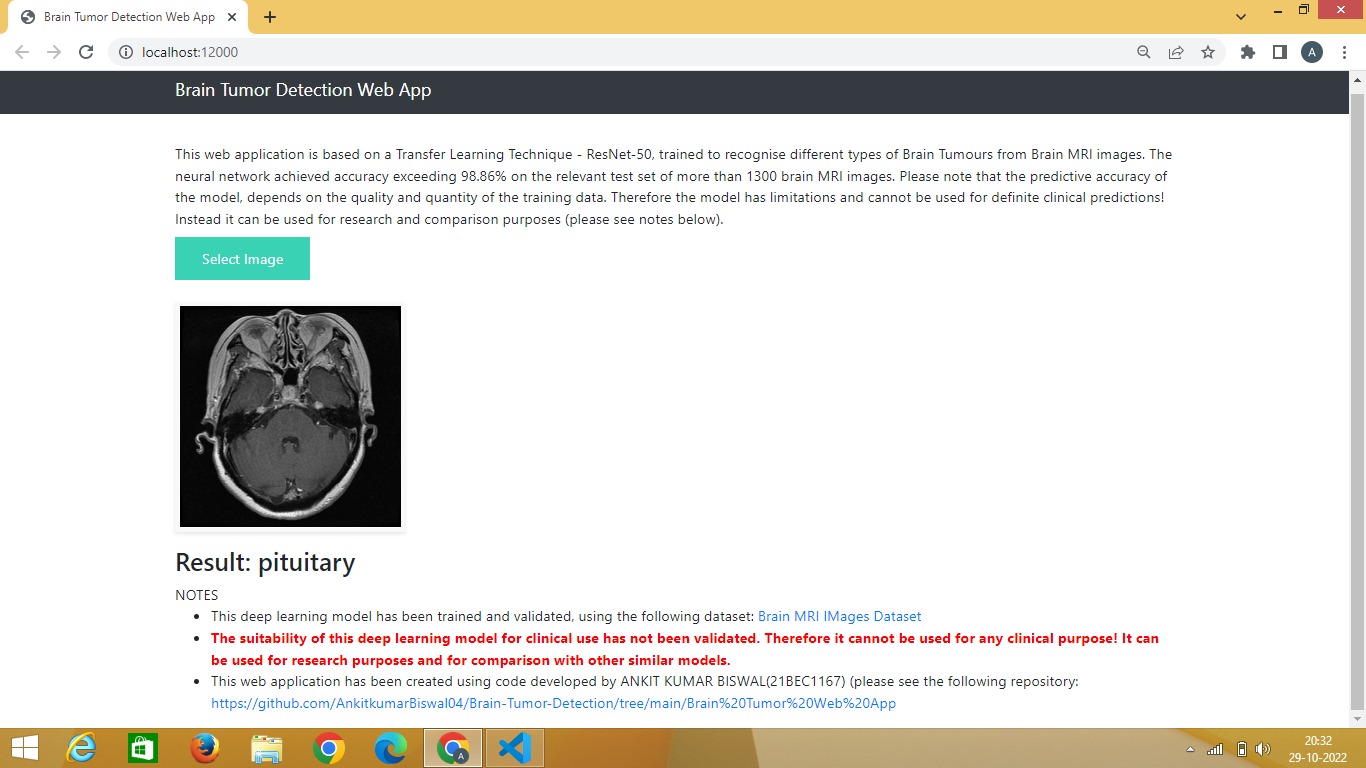
**This is for Pituitary case:-**



**Selecting the image :-**

****

**After selecting the image the predicted result is :-**

****

**REFERENCES**

* Mohsen H et al. Classification using Deep Learning Neural Networks for Brain Tumors. Future Computing and Informatics. 2017:1-4.
* Bauer S.  et al. Multi  scale Modeling for Image Analysis of Brain Tumor Studies. IEEE Transactions on Biomedical Engineering. 2012;59:1.  
  [CrossRef](https://doi.org/10.1109/TBME.2011.2163406)
* Islam A. et al. Multi-fractal Texture Estimation for Detection and Segmentation of Brain Tumors. IEEE.
* Huang M et al. Brain Tumor Segmentation Based on Local Independent Projection-based Classification. IEEE Transactions on Biomedical Engineering.  2013.IEEE.
* Hamamci A et al. Tumor-Cut Segmentation of Brain Tumors on Contrast Enhanced MR Images for Radiosurgery Applications. IEEE Transactions on Medical Imaging. 2012;31:3.  
  [CrossRef](https://doi.org/10.1109/TMI.2011.2181857)
* Bjoern H. M et al. The Multi modal Brain Tumor Image Segmentation Benchmark (BRATS). IEEE Transactions on Medical Imaging.
* Liu J et al. A Survey of MRI-Based Brain Tumor Segmentation Methods. TSINGHUA Science and Technology. 2011;19;6.
* Huda S et al. A Hybrid Feature Selection with Ensemble Classification for Imbalanced Healthcare Data A Case Study for Brain Tumor Diagnosis. IEEE Access.  2017;4.
* Karuppathal and Palanisamy V.  Fuzzy based automatic detection and classification approach for MRI-brain tumor. ARPN Journal of Engineering and Applied Sciences. 2014;9;12.
* Janani and  Meena P. image segmentation for tumor detection using fuzzy inference system. International Journal of Computer Science and Mobile Computing. 2013;2(5):244–248.
* Pereira S et al. Brain Tumor Segmentation using Convolutional Neural Networks in MRI Images. IEEE Transactions on Medical Imaging.
* Zhang J et al. Brain Tumor Segmentation Based on Refined Fully Convolutional Neural Networks with A Hierarchical Dice Loss, Cornell university library computer vision and pattern recognition. 2018.
* [Radiopaedia] http:// radiopedia.org.
* [BRATS 2015] [https://www.smir.ch/BRATS/Start 2015](https://www.smir.ch/BRATS/Start%202015).
* Abiwinanda N, Hanif M, Hesaputra ST, Handayani A, Mengko TR (2019) Brain tumor classification using convolutional neural network. IFMBE Proc 68(1):183–189. <https://doi.org/10.1007/978-981-10-9035-6_33>
* Ayadi W, Elhamzi W, Charfi I, Atri M (2021) Deep CNN for brain tumor classification. Neural Process Lett 53(1):671–700. <https://doi.org/10.1007/s11063-020-10398-2>
* Badža MM, Barjaktarović MC (2020) Classification of brain tumors from MRI images using a convolutional neural network. Appl Sci 10(6):1–13. <https://doi.org/10.3390/app10061999>
* Banan R, Hartmann C (2017) The new WHO 2016 classification of brain tumors—what neurosurgeons need to know. Acta Neurochir 159(3):403–418. <https://doi.org/10.1007/s00701-016-3062-3>
* Barboriak D (2015) Data from RIDER\_NEURO\_MRI. Cancer Imag Arch. <https://doi.org/10.7937/K9/TCIA.2015.VOSN3HN1>
* Cheng J, Huang W, Cao S, Yang R, Yang W, Yun Z, Wang Z, Feng Q (2015) Enhanced performance of brain tumor classification via tumor region augmentation and partition. PLoS ONE 10(10):1–13. <https://doi.org/10.1371/journal.pone.0140381>
* Çinar A, Yildirim M (2020) Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture. Med Hypotheses 139:109684. <https://doi.org/10.1016/j.mehy.2020.109684>
* Clark K, Vendt B, Smith K, Freymann J, Kirb J, Koppel P, Moore S, Phillips S, Maffitt D, Pringle M (2013) The cancer imaging archive (TCIA): Maintaining and operating a public information repository. J Digit Imag 26(6):1045–1057. <https://doi.org/10.1007/s10278-013-9622-7>
* Deepak S, Ameer P (2019) Brain tumor classification using deep CNN features via transfer learning. Comput Biol Med 111:103345. <https://doi.org/10.1016/j.compbiomed.2019.103345>
* Doğantekin A, Özyurt F, Avcı E, Koç M (2019) A novel approach for liver image classification PH-C-ELM. Measurement 137:332–338. <https://doi.org/10.1016/j.measurement.2019.01.060>
* El-Dahshan ESA, Hosny T, Salem ABM (2010) Hybrid intelligent techniques for MRI brain images classification. Digital Signal Process 20(2):433–441. <https://doi.org/10.1016/j.dsp.2009.07.002>
* Ertosun MG, Rubin DL (2015) Automated grading of gliomas using deep learning in digital pathology images: a modular approach with ensemble of convolutional neural networks. Annu Symp Proc AMIA Symp 2015:1899–1908
* Irmak E (2020) Implementation of convolutional neural network approach for COVID-19 disease detection. Physiol Genom 52(12):590–601. <https://doi.org/10.1152/physiolgenomics.00084.2020>
* Irmak E (2021) COVID-19 disease severity assessment using CNN model. IET Image Process. <https://doi.org/10.1049/ipr2.12153>
* Kabir Anaraki A, Ayati M, Kazemi F (2019) Magnetic Resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. Biocybern Biomed Eng 39(1):63–74. <https://doi.org/10.1016/j.bbe.2018.10.004>
* Khan HA, Jue W, Mushtaq M, Mushtaq MU (2020) Brain tumor classification in MRI image using convolutional neural network. Math Biosci Eng 17(5):6203–6216. <https://doi.org/10.3934/MBE.2020328>
* [Article](https://doi.org/10.3934%2FMBE.2020328) [MathSciNet](http://www.ams.org/mathscinet-getitem?mr=4160269) [Google Scholar](http://scholar.google.com/scholar_lookup?&title=Brain%20tumor%20classification%20in%20MRI%20image%20using%20convolutional%20neural%20network&journal=Math%20Biosci%20Eng&doi=10.3934%2FMBE.2020328&volume=17&issue=5&pages=6203-6216&publication_year=2020&author=Khan%2CHA&author=Jue%2CW&author=Mushtaq%2CM&author=Mushtaq%2CMU)
* Khawaldeh S, Pervaiz U, Rafiq A, Alkhawaldeh RS (2017) Noninvasive grading of glioma tumor using magnetic resonance imaging with convolutional neural networks. Appl Sci 8(1):1–17. <https://doi.org/10.3390/app8010027>
* Kleihues P, Burger PC, Scheithauer BW (2012) Histological typing of tumours of the central nervous system, 2nd edn. Springer, Berlin
* Lisa S, Flanders Adam E, Mikkelsen JR, Tom Andrews DW (2015) Data From REMBRANDT. Cancer Imag Arch. <https://doi.org/10.7937/K9/TCIA.2015.588OZUZB>
* Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Vander Laak JAWM, Van Ginneken B, Sánchez CI (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88. <https://doi.org/10.1016/j.media.2017.07.005>
* Lotan E, Jain R, Razavian N, Fatterpekar GM, Lui YW (2019) State of the art: Machine learning applications in glioma imaging. Am J Roentgenol 212(1):26–37
* Mehmood A, Maqsood M, Bashir M, Shuyuan Y (2020) A deep siamese convolution neural network for multi-class classification of alzheimer disease. Brain Sci 10(2):1–15. <https://doi.org/10.3390/brainsci10020084>
* Mehmood A, Yang S, Feng Z, Wang M, Ahmad ALS, Khan R, Maqsood M, Yaqub M (2021) A transfer learning approach for early diagnosis of alzheimer’s disease on MRI images. Neuroscience 15(460):43–52. <https://doi.org/10.1016/j.neuroscience.2021.01.002>
* Mehrotra R, Ansari MA, Agrawal R, Anand RS (2020) A Transfer learning approach for AI-based classification of brain tumors. Mach Learn Appl 2(9):1–12. <https://doi.org/10.1016/j.mlwa.2020.100003>
* Mohsen H, El-Dahshan ESA, El-Horbaty ESM, Salem ABM (2018) Classification using deep learning neural networks for brain tumors. Future Comput Informat J 3(1):68–71. <https://doi.org/10.1016/j.fcij.2017.12.001>
* Muhammad K, Khan S, Ser JD, Albuquerque VHC (2021) Deep learning for multigrade brain tumor classification in smart healthcare systems: a prospective survey. IEEE Trans Neural Netw Learn Syst 32(2):507–522. <https://doi.org/10.1109/TNNLS.2020.2995800>
* Mzoughi H, Njeh I, Wali A, Slima M, Ben BenHamida A, Mhiri C, Mahfoudhe K (2020) Deep Multi-Scale 3D convolutional neural network (CNN) for MRI gliomas brain tumor classification. J Digit Imag 33(4):903–915. <https://doi.org/10.1007/s10278-020-00347-9>