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

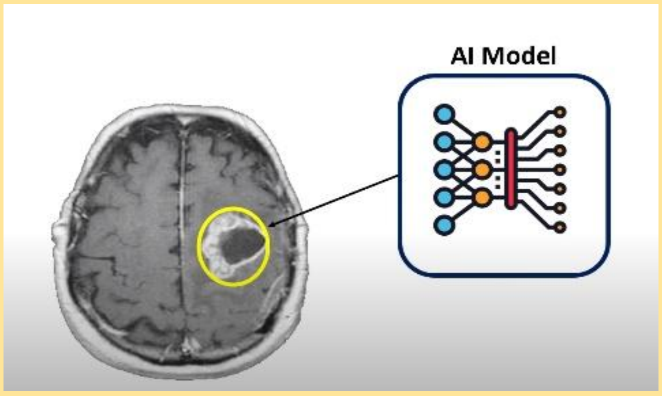
B.TECH - Electronics and Communications Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

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.

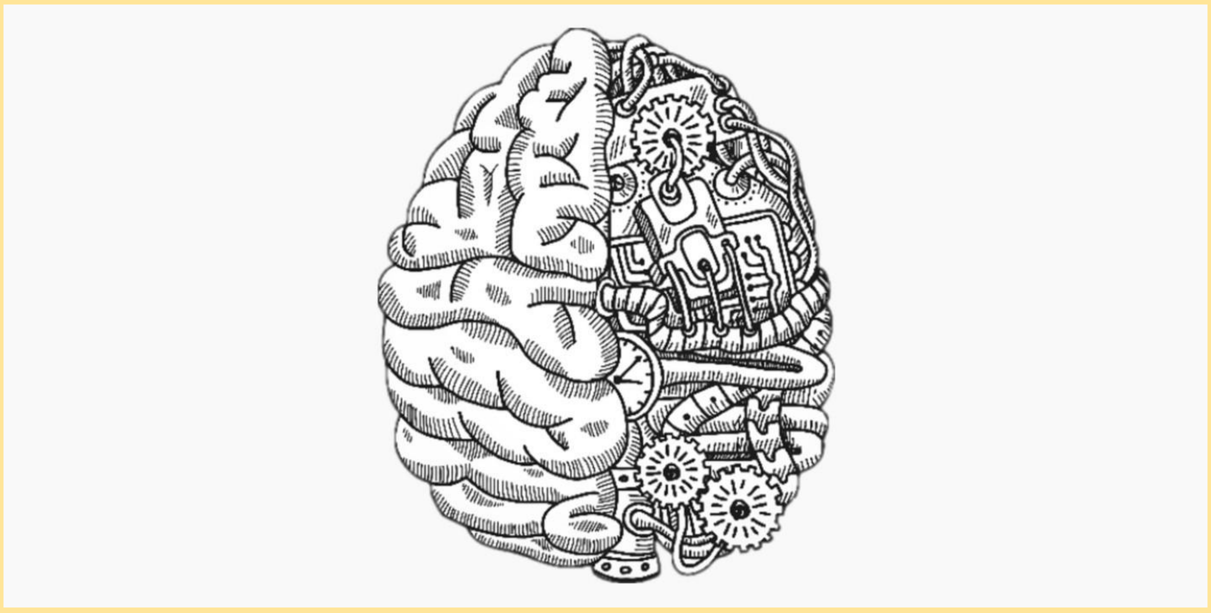
Context:-

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:-

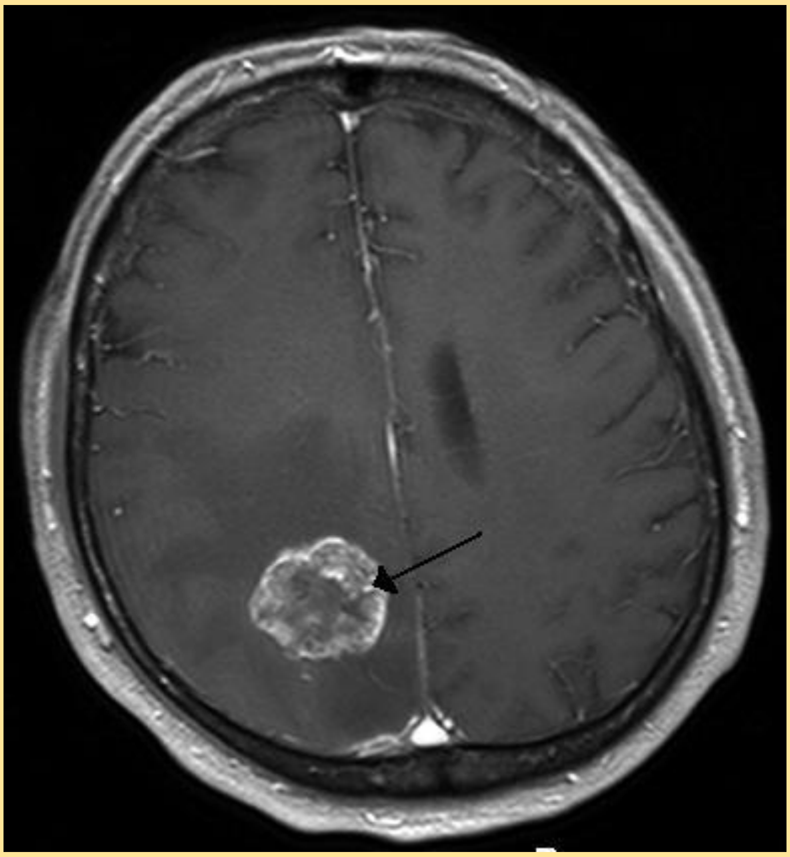
In this notebook, I've used CNN to perform Image Classification on the Brain Tumor dataset.
Since this dataset is small, if we train a neural network to it, it won't really give us a good result.

Therefore, I'm going to use the concept of Transfer Learning to train the model to get really accurate results.



What is Brain Tumor?

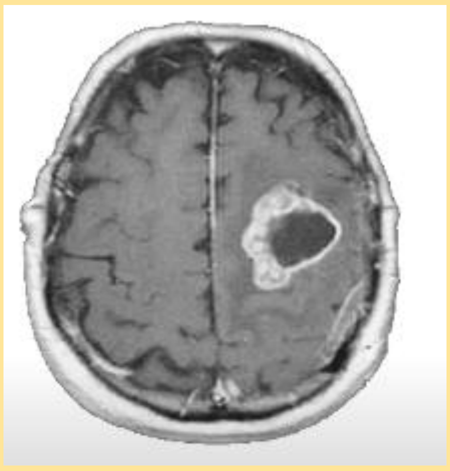
A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain 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)

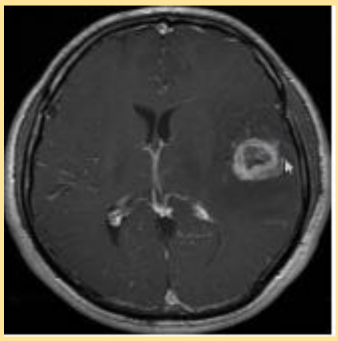
What is Glioma:-

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.

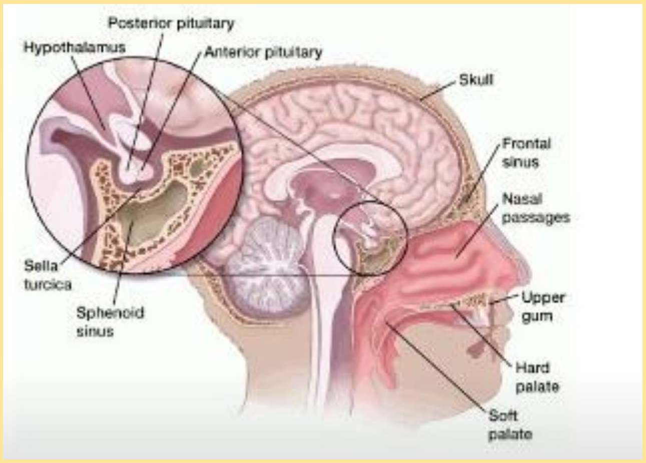
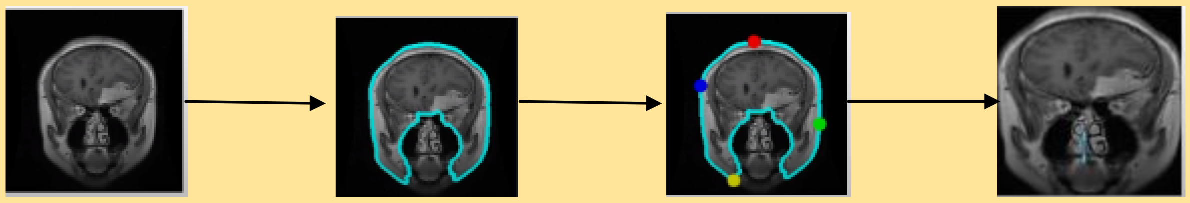


Image Pre-Processing:-

- 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.

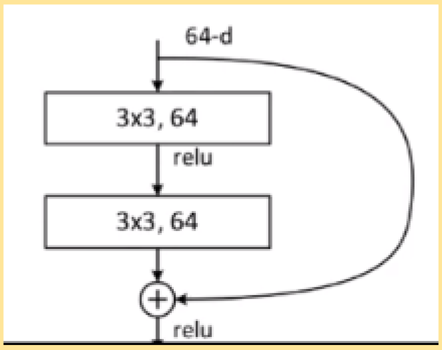


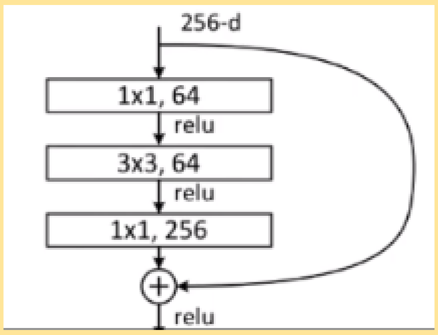
Step:1 Step:2 Step:3 Step:4

- **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.

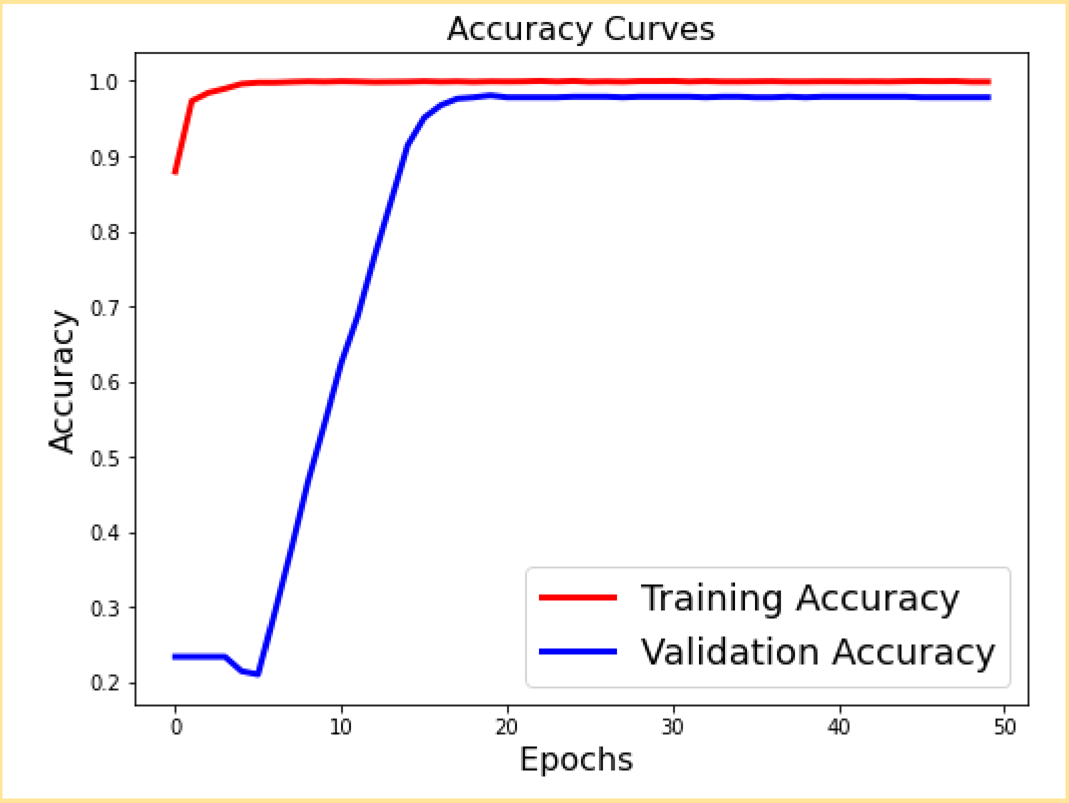


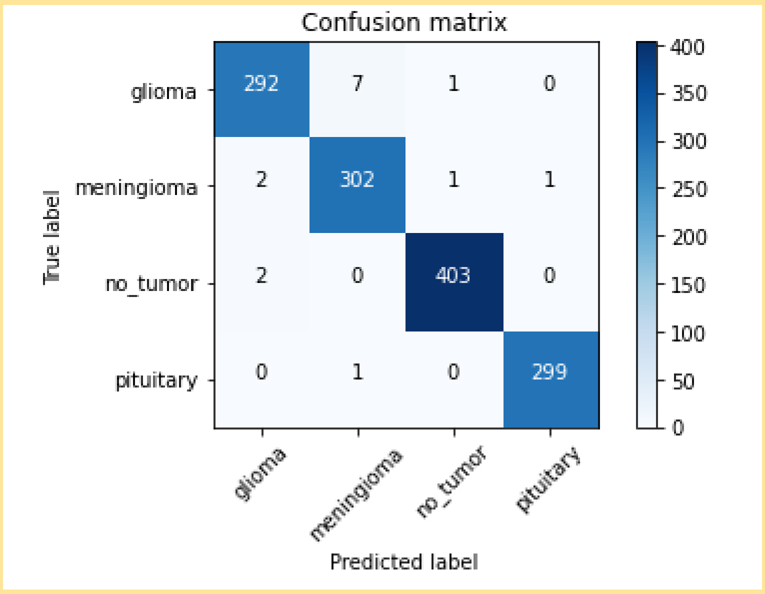


Results and Discussion:-

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:-

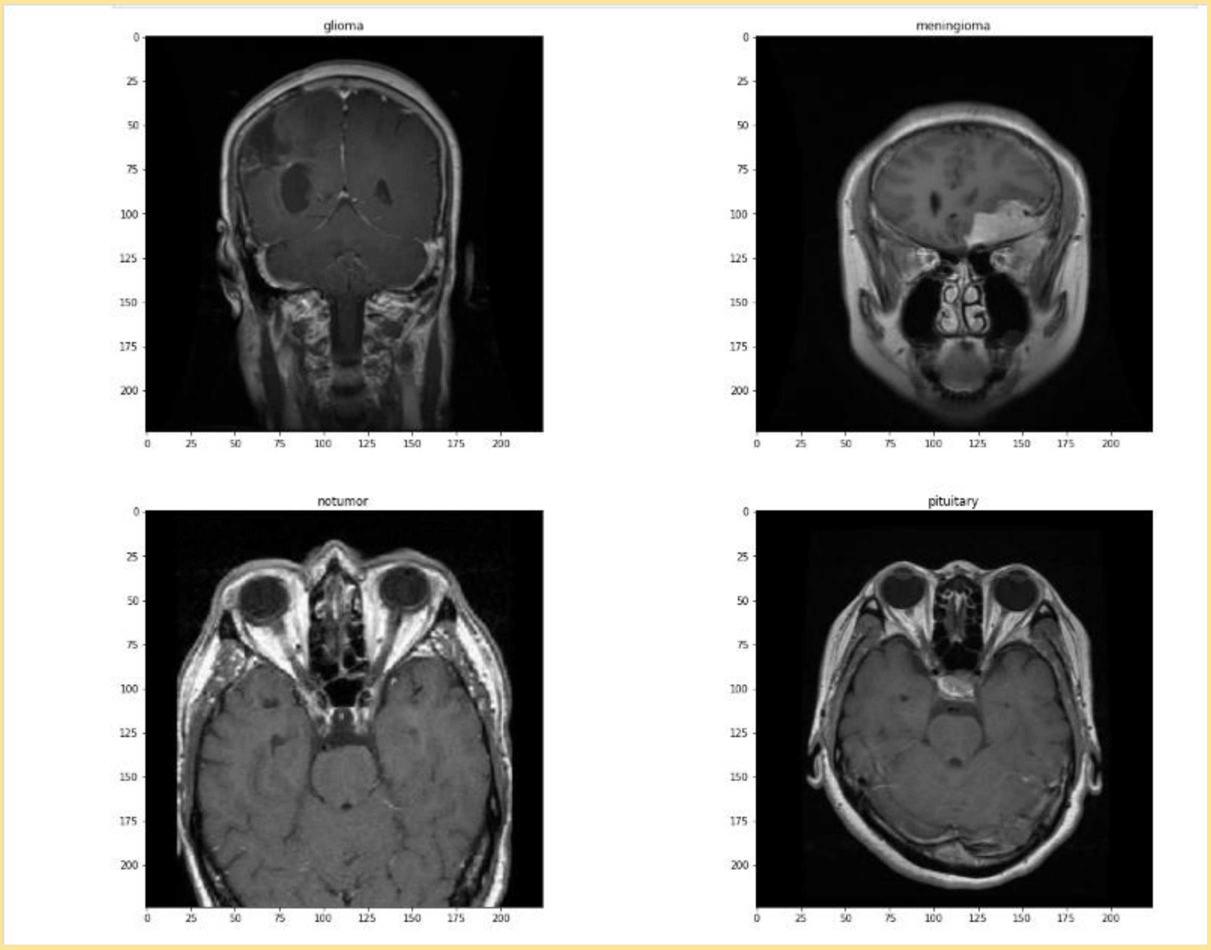
```
import tensorflow
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img, save_img, img_to_array
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras import 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])
```

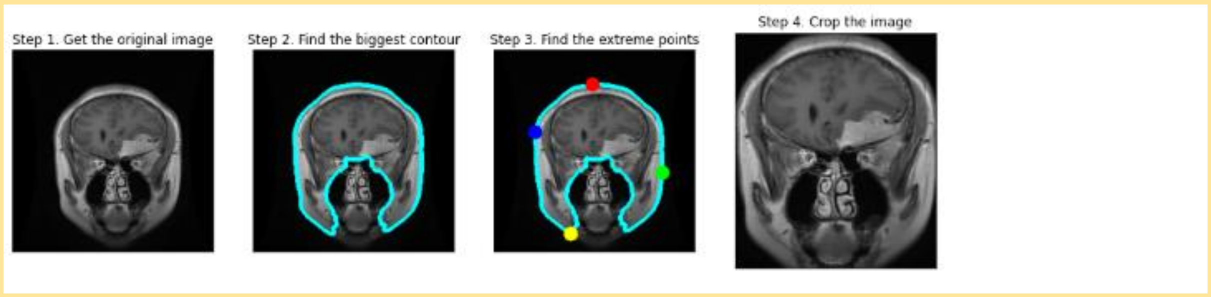


Cropping Images Demo:-

```
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)
```

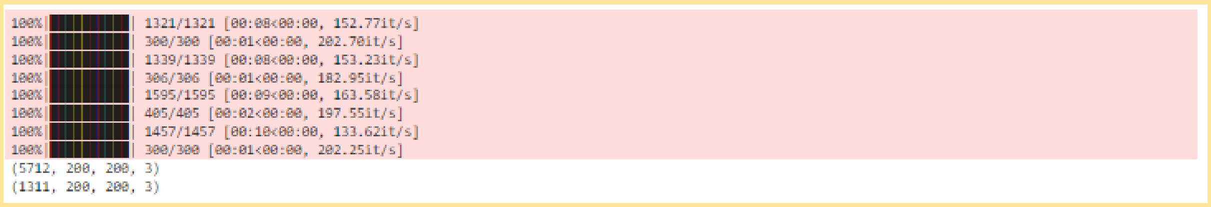
```
        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])
        ADD_PIXELS = 0
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS,
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()
```


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

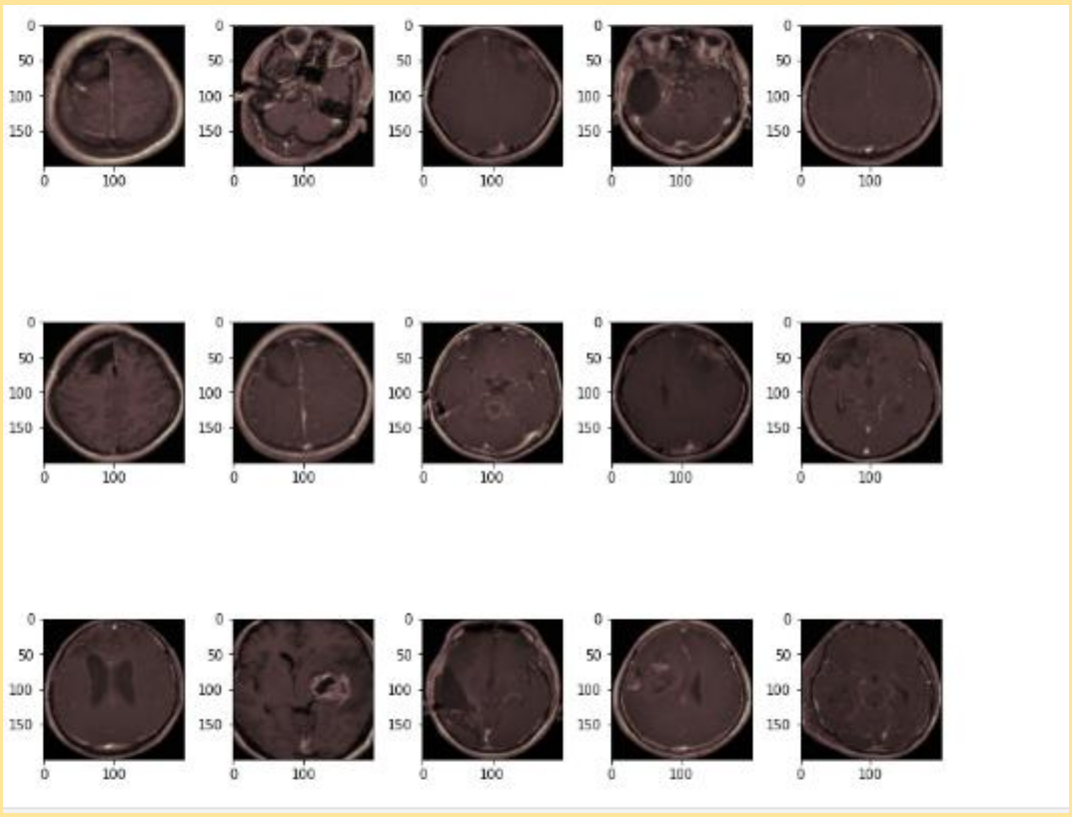
```
labels = ['glioma', 'meningioma', 'notumor', 'pituitary']

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)
```

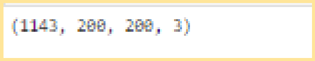


```
100%|██████████| 1321/1321 [00:08<00:00, 152.77it/s]
100%|██████████| 300/300 [00:01<00:00, 202.70it/s]
100%|██████████| 1339/1339 [00:08<00:00, 153.23it/s]
100%|██████████| 306/306 [00:01<00:00, 182.95it/s]
100%|██████████| 1595/1595 [00:09<00:00, 163.58it/s]
100%|██████████| 405/405 [00:02<00:00, 197.55it/s]
100%|██████████| 1457/1457 [00:10<00:00, 133.62it/s]
100%|██████████| 300/300 [00:01<00:00, 202.25it/s]
(5712, 200, 200, 3)
(1311, 200, 200, 3)
```

```
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()
```



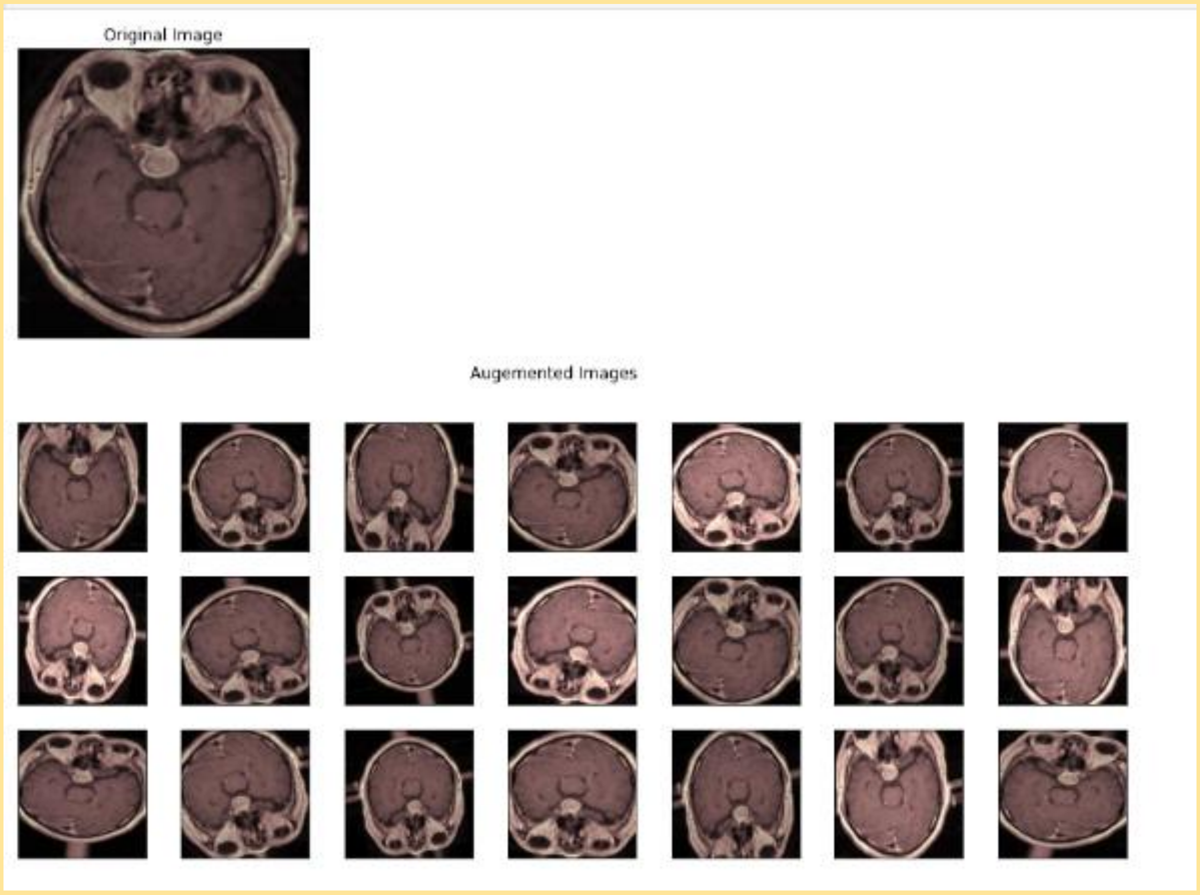
```
x_train, y_train = shuffle(x_train,y_train, random_state=42)

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)
```



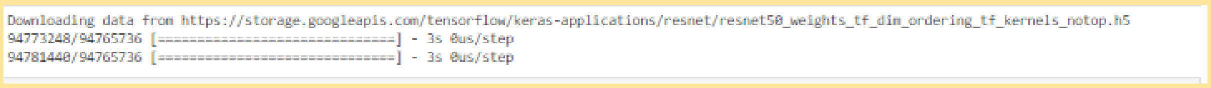
```
(1143, 200, 200, 3)
```

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()
```

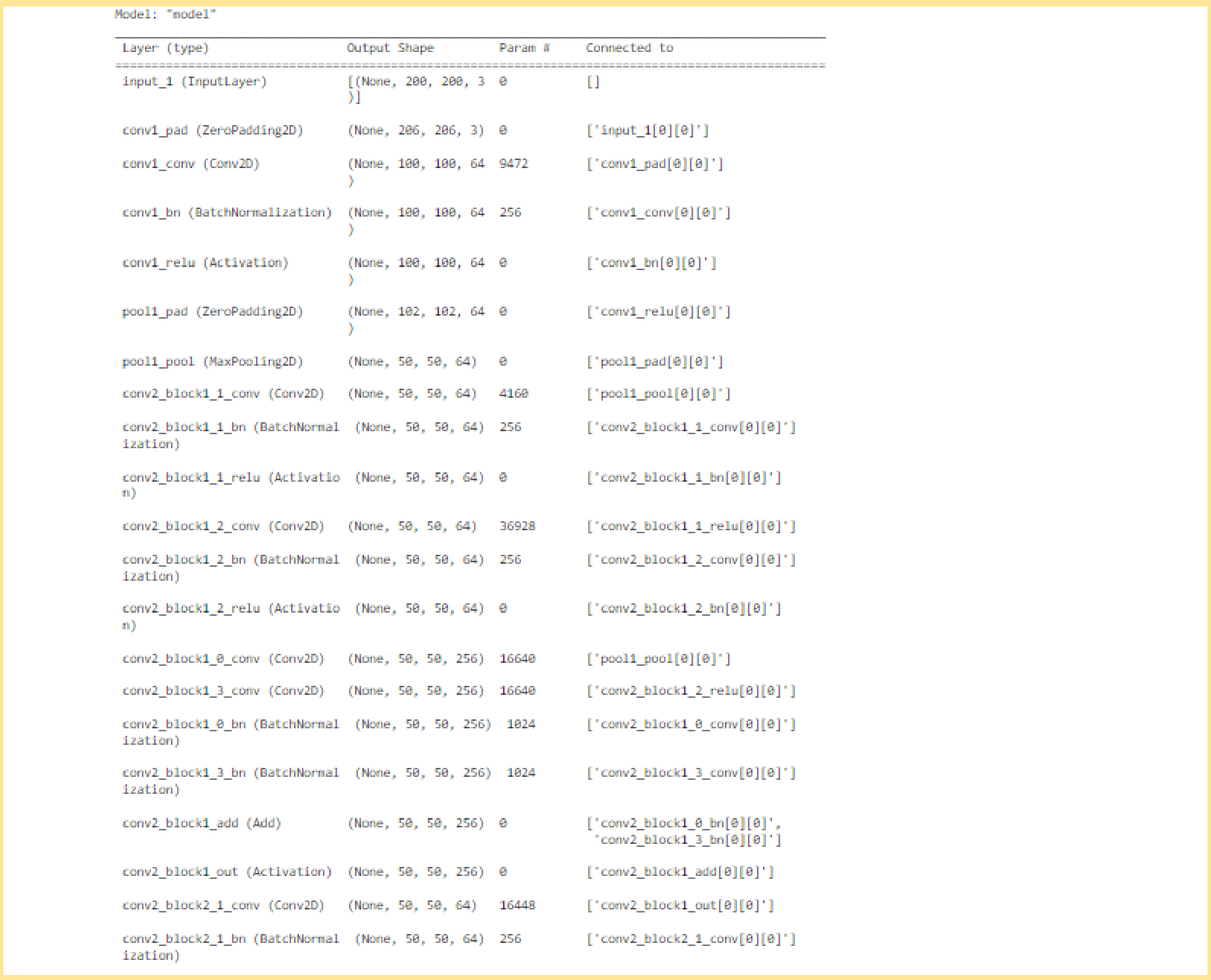


```
# ImageDataGenerator transforms each image in the batch by a series
of random translations, rotations, etc.
datagen = ImageDataGenerator(
      rotation_range=10,
     width_shift_range=0.05,
     height_shift_range=0.05,
     horizontal_flip=True)

# 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
```

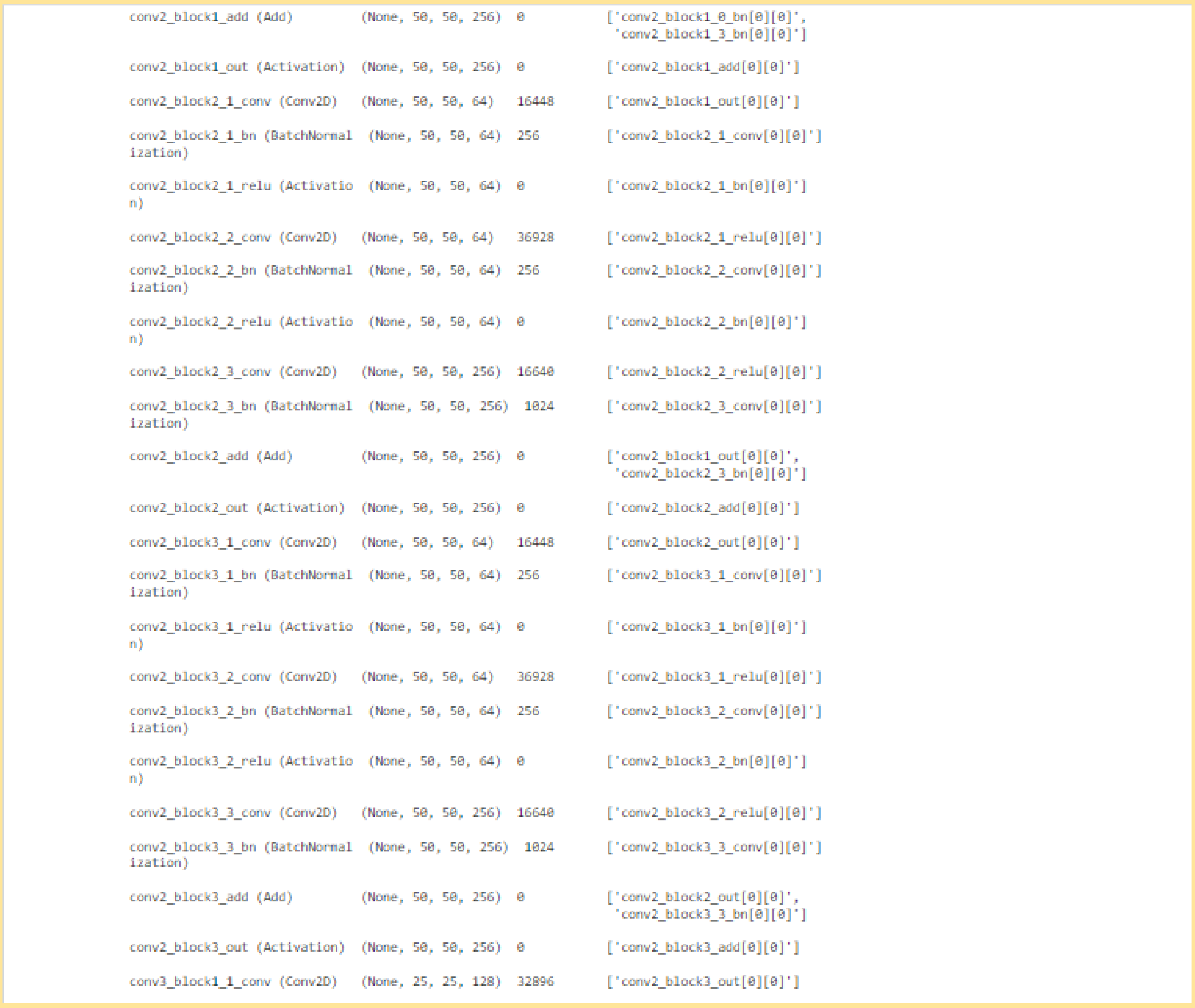


```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 3s 0us/step
94781440/94765736 [==============================] - 3s 0us/step
```

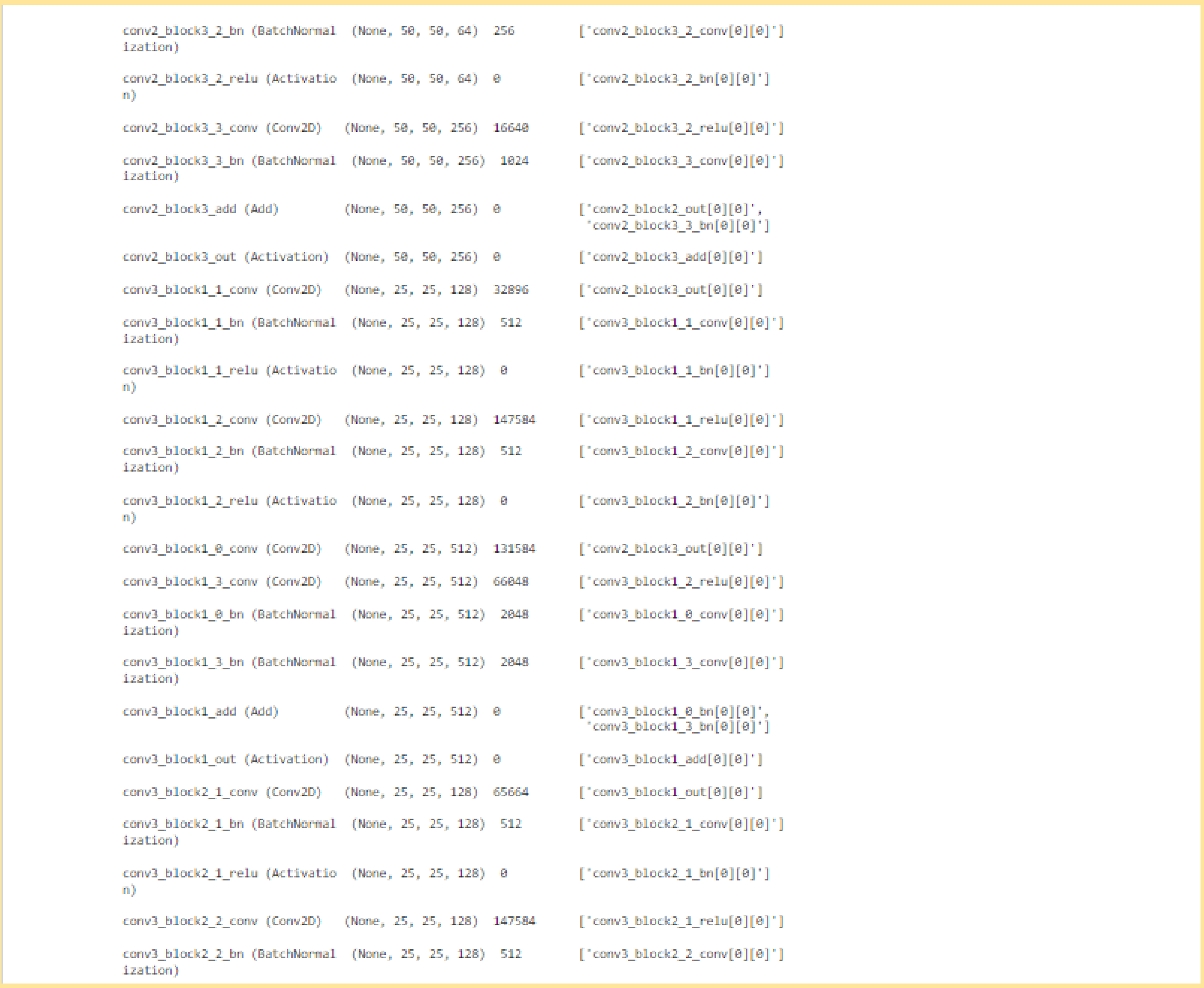
```
model = conv_base.output
model = GlobalAveragePooling2D()(model)
model = Dropout(0.4)(model)
model = Dense(4, activation="softmax")(model)
model = Model(inputs= conv_base.input, outputs= model)

#compile our model.
adam = Adam(learning_rate=0.0001)
model.compile(optimizer=adam, loss = 'categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```



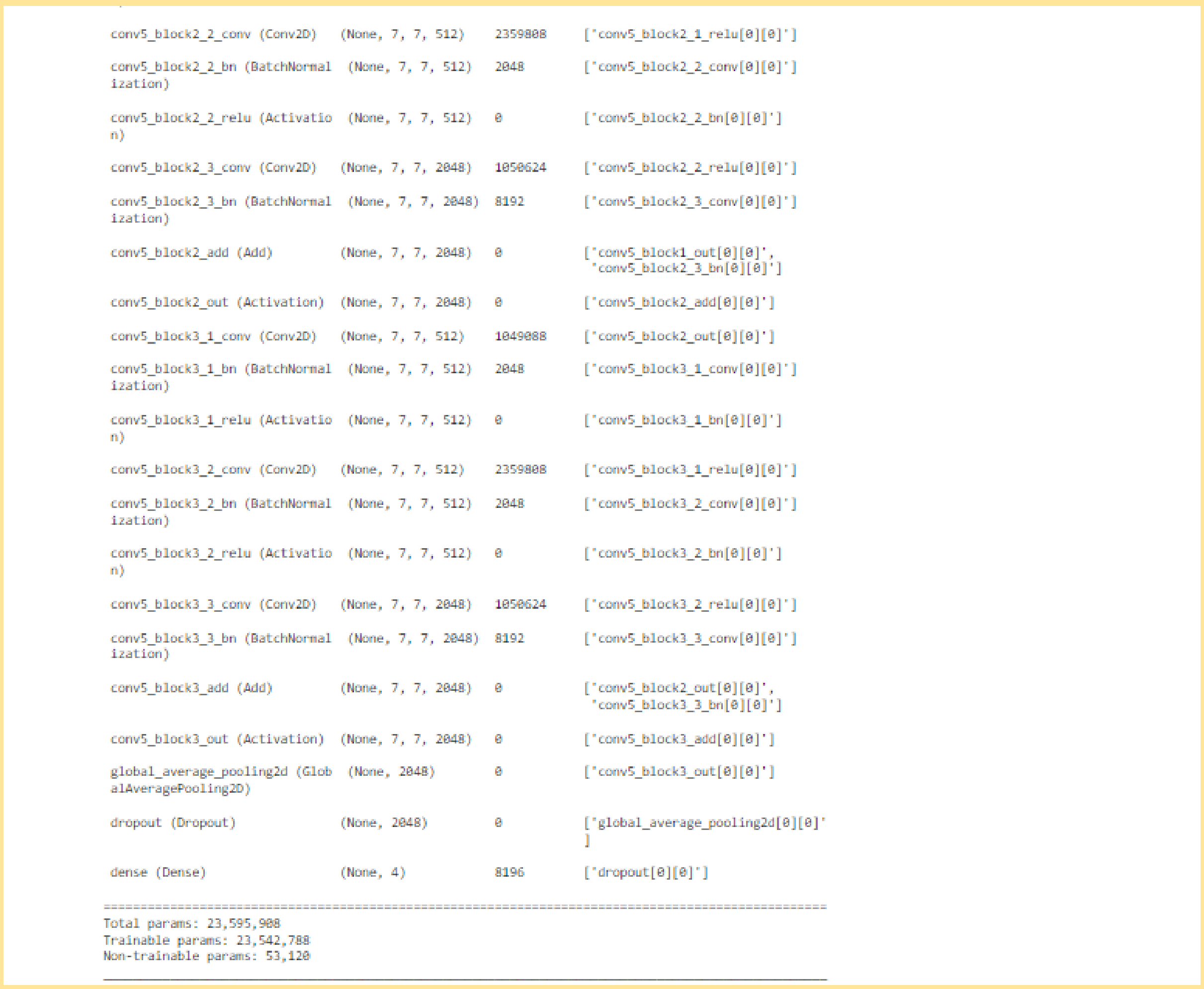
```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 200, 200, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 206, 206, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 100, 100, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 100, 100, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 100, 100, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 102, 102, 64  0           ['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 50, 50, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 50, 50, 64)   4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 50, 50, 64)   36928       ['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 50, 50, 256)  16640       ['pool1_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 50, 50, 256)  16640       ['conv2_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 50, 50, 256)  1024       ['conv2_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 50, 50, 256)  1024       ['conv2_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 50, 50, 256)  0           ['conv2_block1_0_bn[0][0]',
                                                                  'conv2_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 50, 50, 256)  0           ['conv2_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 50, 50, 64)   16448       ['conv2_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block2_1_conv[0][0]']
 ization)
```



```
 conv2_block1_add (Add)         (None, 50, 50, 256)  0           ['conv2_block1_0_bn[0][0]',
                                                                  'conv2_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 50, 50, 256)  0           ['conv2_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 50, 50, 64)   16448       ['conv2_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 50, 50, 64)   36928       ['conv2_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 50, 50, 256)  16640       ['conv2_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 50, 50, 256)  1024       ['conv2_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 50, 50, 256)  0           ['conv2_block1_out[0][0]',
                                                                  'conv2_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 50, 50, 256)  0           ['conv2_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 50, 50, 64)   16448       ['conv2_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 50, 50, 64)   36928       ['conv2_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 50, 50, 256)  16640       ['conv2_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 50, 50, 256)  1024       ['conv2_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 50, 50, 256)  0           ['conv2_block2_out[0][0]',
                                                                  'conv2_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 50, 50, 256)  0           ['conv2_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 25, 25, 128)  32896       ['conv2_block3_out[0][0]']
```

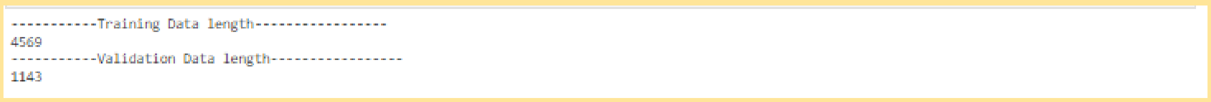


```
conv2_block3_2_bn (BatchNormal  (None, 50, 50, 64)  256         ['conv2_block3_2_conv[0][0]']
ization)

conv2_block3_2_relu (Activatio  (None, 50, 50, 64)  0           ['conv2_block3_2_bn[0][0]']
n)

conv2_block3_3_conv (Conv2D)   (None, 50, 50, 256)  16640       ['conv2_block3_2_relu[0][0]']

conv2_block3_3_bn (BatchNormal  (None, 50, 50, 256)  1024       ['conv2_block3_3_conv[0][0]']
ization)

conv2_block3_add (Add)         (None, 50, 50, 256)  0           ['conv2_block2_out[0][0]',
                                                                 'conv2_block3_3_bn[0][0]']

conv2_block3_out (Activation)  (None, 50, 50, 256)  0           ['conv2_block3_add[0][0]']

conv3_block1_1_conv (Conv2D)   (None, 25, 25, 128)  32896       ['conv2_block3_out[0][0]']

conv3_block1_1_bn (BatchNormal  (None, 25, 25, 128)  512        ['conv3_block1_1_conv[0][0]']
ization)

conv3_block1_1_relu (Activatio  (None, 25, 25, 128)  0          ['conv3_block1_1_bn[0][0]']
n)

conv3_block1_2_conv (Conv2D)   (None, 25, 25, 128)  147584      ['conv3_block1_1_relu[0][0]']

conv3_block1_2_bn (BatchNormal  (None, 25, 25, 128)  512        ['conv3_block1_2_conv[0][0]']
ization)

conv3_block1_2_relu (Activatio  (None, 25, 25, 128)  0          ['conv3_block1_2_bn[0][0]']
n)

conv3_block1_0_conv (Conv2D)   (None, 25, 25, 512)  131584      ['conv2_block3_out[0][0]']

conv3_block1_3_conv (Conv2D)   (None, 25, 25, 512)  66048       ['conv3_block1_2_relu[0][0]']

conv3_block1_0_bn (BatchNormal  (None, 25, 25, 512)  2048       ['conv3_block1_0_conv[0][0]']
ization)

conv3_block1_3_bn (BatchNormal  (None, 25, 25, 512)  2048       ['conv3_block1_3_conv[0][0]']
ization)

conv3_block1_add (Add)         (None, 25, 25, 512)  0           ['conv3_block1_0_bn[0][0]',
                                                                 'conv3_block1_3_bn[0][0]']

conv3_block1_out (Activation)  (None, 25, 25, 512)  0           ['conv3_block1_add[0][0]']

conv3_block2_1_conv (Conv2D)   (None, 25, 25, 128)  65664       ['conv3_block1_out[0][0]']

conv3_block2_1_bn (BatchNormal  (None, 25, 25, 128)  512        ['conv3_block2_1_conv[0][0]']
ization)

conv3_block2_1_relu (Activatio  (None, 25, 25, 128)  0          ['conv3_block2_1_bn[0][0]']
n)

conv3_block2_2_conv (Conv2D)   (None, 25, 25, 128)  147584      ['conv3_block2_1_relu[0][0]']

conv3_block2_2_bn (BatchNormal  (None, 25, 25, 128)  512        ['conv3_block2_2_conv[0][0]']
ization)
```



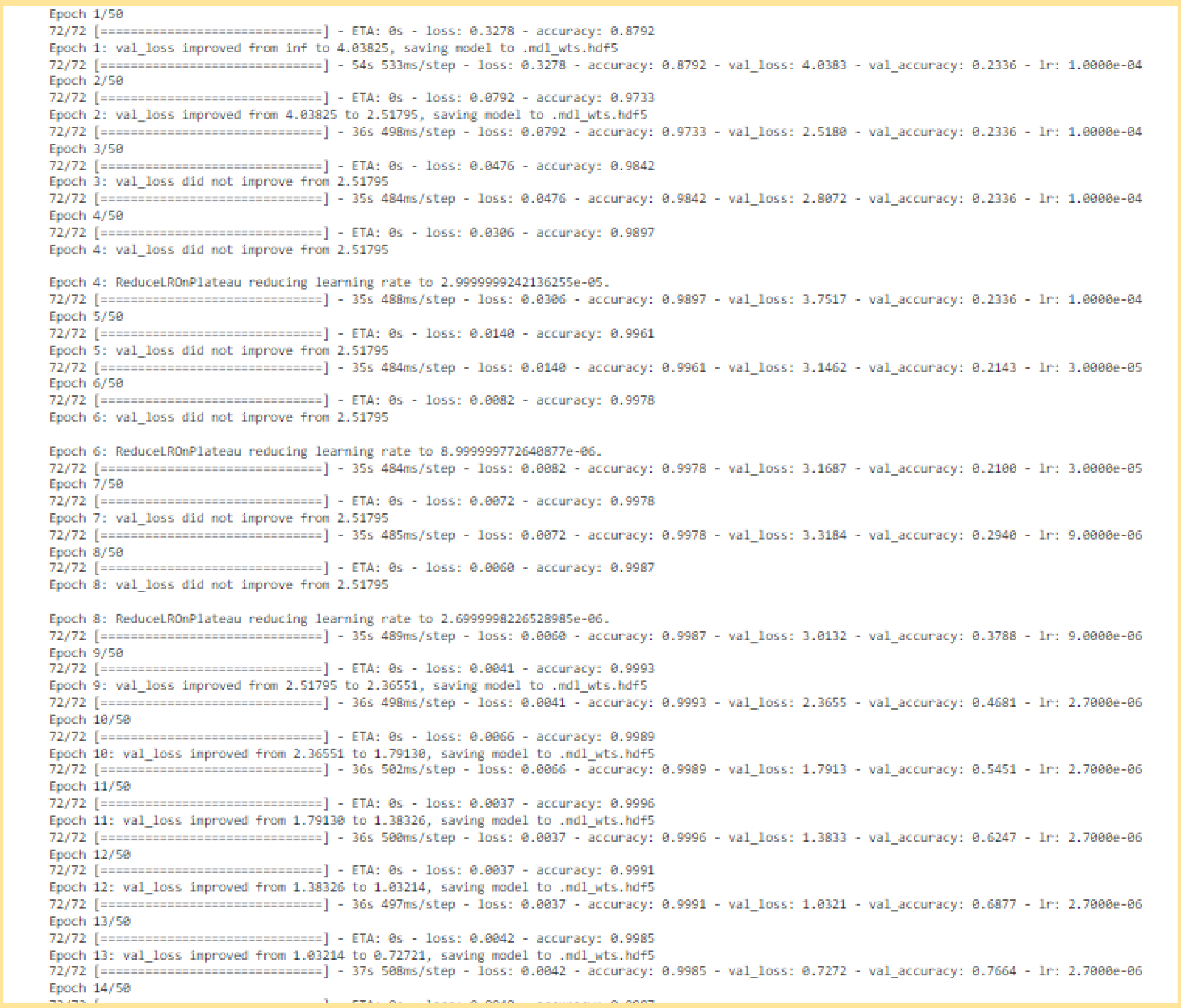
```
 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5_block1_out[0][0]',
                                                                  'conv5_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5_block2_out[0][0]',
                                                                  'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block3_add[0][0]']

 global_average_pooling2d (Glob  (None, 2048)        0           ['conv5_block3_out[0][0]']
 alAveragePooling2D)

 dropout (Dropout)              (None, 2048)         0           ['global_average_pooling2d[0][0]'
                                                                 ]

 dense (Dense)                  (None, 4)            8196        ['dropout[0][0]']

==================================================================================================
Total params: 23,595,908
Trainable params: 23,542,788
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

```
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)
```

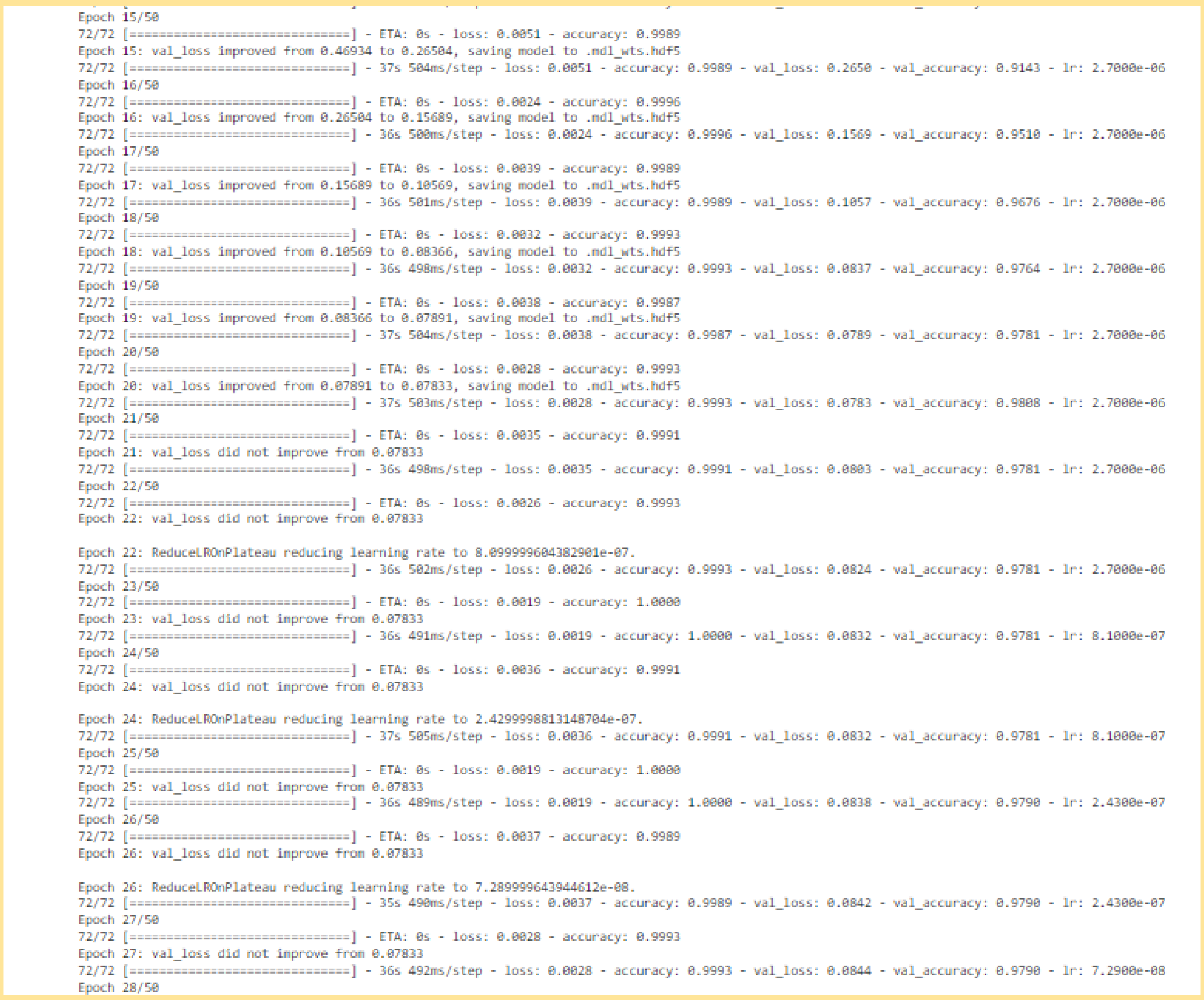


```
-----------Training Data length-----------------
4569
-----------Validation Data length-----------------
1143
```

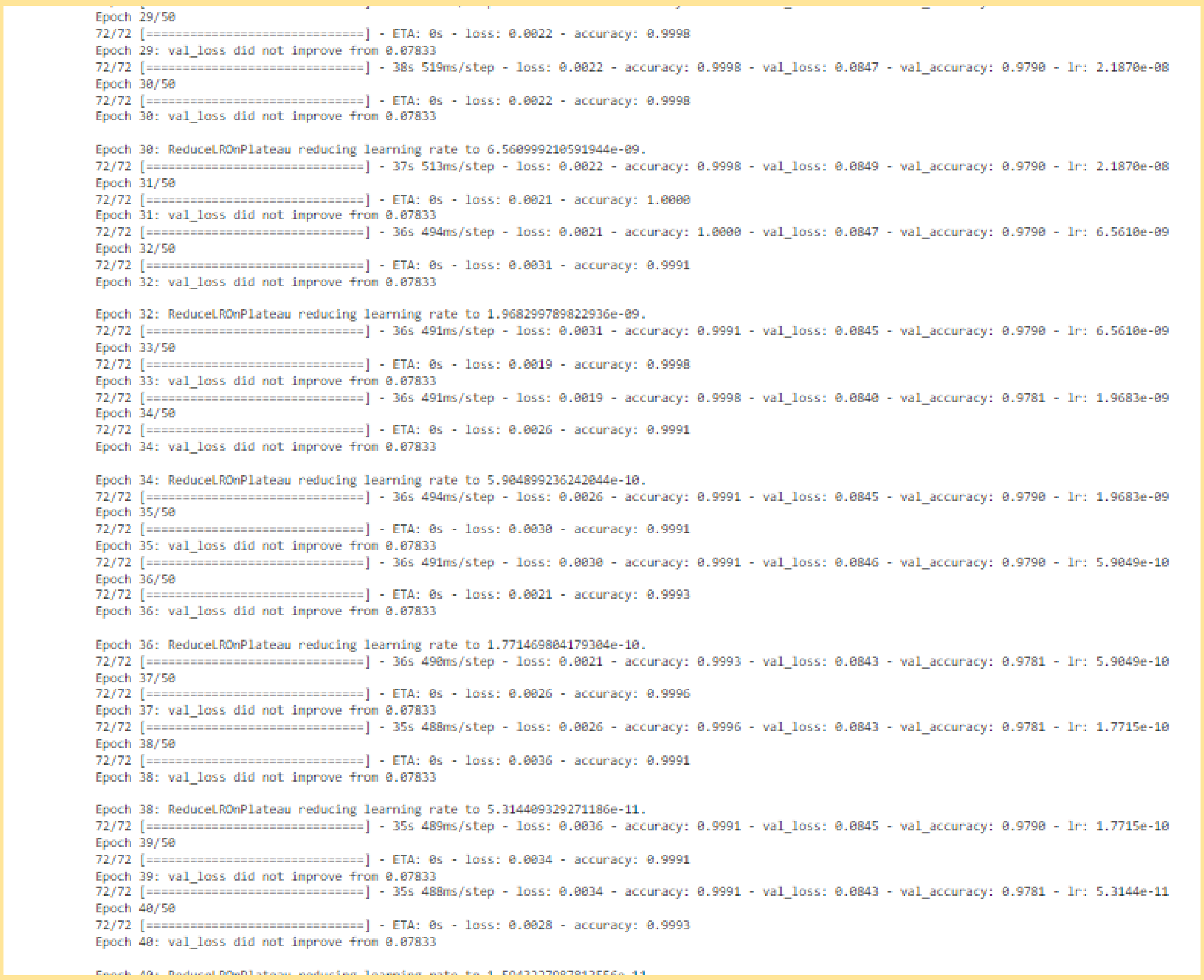
```
history = model.fit(datagen.flow(x_train, y_train,
batch_size=64),validation_data = (x_val,y_val),epochs = 50,callbacks
= callbacks)
```



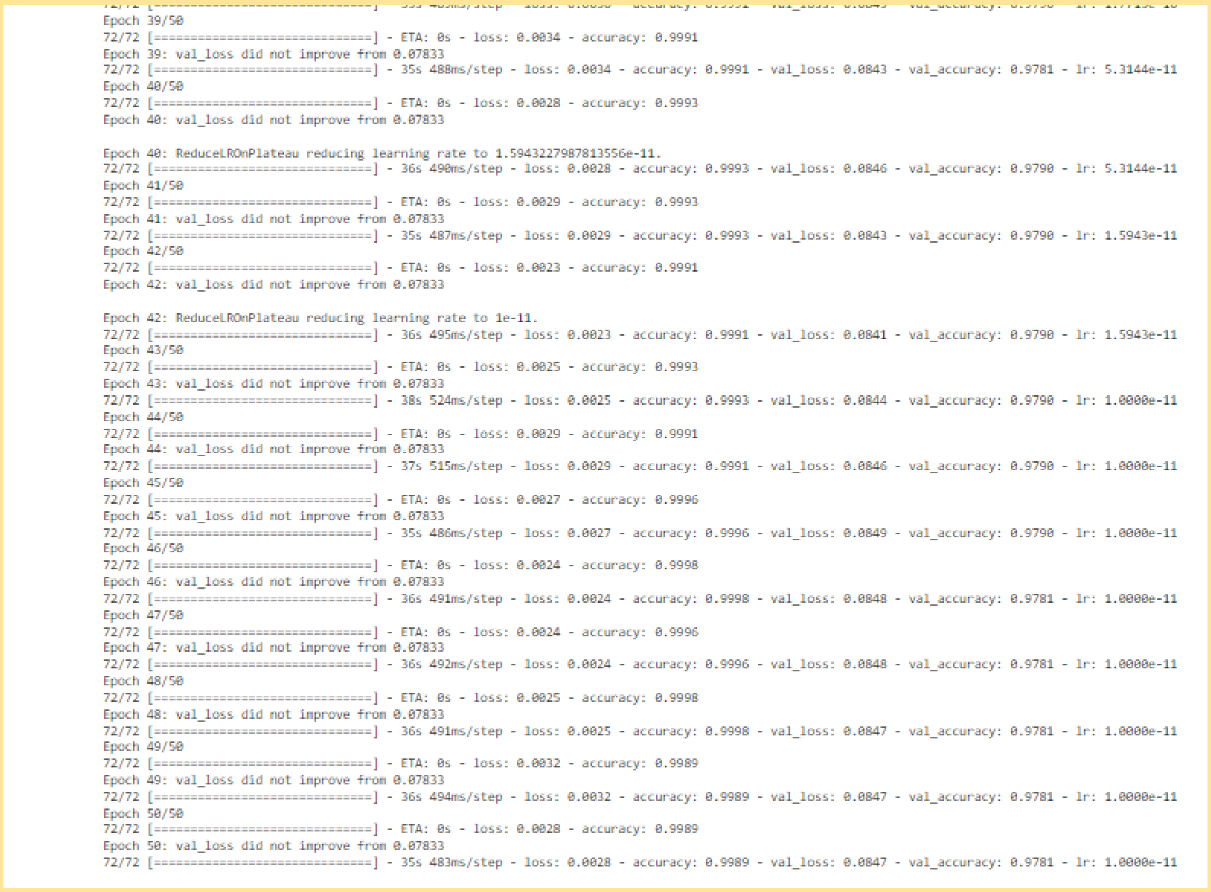
```
Epoch 1/50
72/72 [==============================] - ETA: 0s - loss: 0.3278 - accuracy: 0.8792
Epoch 1: val_loss improved from inf to 4.03825, saving model to .mdl_wts.hdf5
72/72 [==============================] - 54s 533ms/step - loss: 0.3278 - accuracy: 0.8792 - val_loss: 4.0383 - val_accuracy: 0.2336 - lr: 1.0000e-04
Epoch 2/50
72/72 [==============================] - ETA: 0s - loss: 0.0792 - accuracy: 0.9733
Epoch 2: val_loss improved from 4.03825 to 2.51795, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 498ms/step - loss: 0.0792 - accuracy: 0.9733 - val_loss: 2.5180 - val_accuracy: 0.2336 - lr: 1.0000e-04
Epoch 3/50
72/72 [==============================] - ETA: 0s - loss: 0.0476 - accuracy: 0.9842
Epoch 3: val_loss did not improve from 2.51795
72/72 [==============================] - 35s 484ms/step - loss: 0.0476 - accuracy: 0.9842 - val_loss: 2.8072 - val_accuracy: 0.2336 - lr: 1.0000e-04
Epoch 4/50
72/72 [==============================] - ETA: 0s - loss: 0.0306 - accuracy: 0.9897
Epoch 4: val_loss did not improve from 2.51795

Epoch 4: ReduceLROnPlateau reducing learning rate to 2.9999999242136255e-05.
72/72 [==============================] - 35s 488ms/step - loss: 0.0306 - accuracy: 0.9897 - val_loss: 3.7517 - val_accuracy: 0.2336 - lr: 1.0000e-04
Epoch 5/50
72/72 [==============================] - ETA: 0s - loss: 0.0140 - accuracy: 0.9961
Epoch 5: val_loss did not improve from 2.51795
72/72 [==============================] - 35s 484ms/step - loss: 0.0140 - accuracy: 0.9961 - val_loss: 3.1462 - val_accuracy: 0.2143 - lr: 3.0000e-05
Epoch 6/50
72/72 [==============================] - ETA: 0s - loss: 0.0082 - accuracy: 0.9978
Epoch 6: val_loss did not improve from 2.51795

Epoch 6: ReduceLROnPlateau reducing learning rate to 8.999999772640877e-06.
72/72 [==============================] - 35s 484ms/step - loss: 0.0082 - accuracy: 0.9978 - val_loss: 3.1687 - val_accuracy: 0.2100 - lr: 3.0000e-05
Epoch 7/50
72/72 [==============================] - ETA: 0s - loss: 0.0072 - accuracy: 0.9978
Epoch 7: val_loss did not improve from 2.51795
72/72 [==============================] - 35s 485ms/step - loss: 0.0072 - accuracy: 0.9978 - val_loss: 3.3184 - val_accuracy: 0.2940 - lr: 9.0000e-06
Epoch 8/50
72/72 [==============================] - ETA: 0s - loss: 0.0060 - accuracy: 0.9987
Epoch 8: val_loss did not improve from 2.51795

Epoch 8: ReduceLROnPlateau reducing learning rate to 2.6999998226528985e-06.
72/72 [==============================] - 35s 489ms/step - loss: 0.0060 - accuracy: 0.9987 - val_loss: 3.0132 - val_accuracy: 0.3788 - lr: 9.0000e-06
Epoch 9/50
72/72 [==============================] - ETA: 0s - loss: 0.0041 - accuracy: 0.9993
Epoch 9: val_loss improved from 2.51795 to 2.36551, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 498ms/step - loss: 0.0041 - accuracy: 0.9993 - val_loss: 2.3655 - val_accuracy: 0.4681 - lr: 2.7000e-06
Epoch 10/50
72/72 [==============================] - ETA: 0s - loss: 0.0066 - accuracy: 0.9989
Epoch 10: val_loss improved from 2.36551 to 1.79130, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 502ms/step - loss: 0.0066 - accuracy: 0.9989 - val_loss: 1.7913 - val_accuracy: 0.5451 - lr: 2.7000e-06
Epoch 11/50
72/72 [==============================] - ETA: 0s - loss: 0.0037 - accuracy: 0.9996
Epoch 11: val_loss improved from 1.79130 to 1.38326, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 500ms/step - loss: 0.0037 - accuracy: 0.9996 - val_loss: 1.3833 - val_accuracy: 0.6247 - lr: 2.7000e-06
Epoch 12/50
72/72 [==============================] - ETA: 0s - loss: 0.0037 - accuracy: 0.9991
Epoch 12: val_loss improved from 1.38326 to 1.03214, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 497ms/step - loss: 0.0037 - accuracy: 0.9991 - val_loss: 1.0321 - val_accuracy: 0.6877 - lr: 2.7000e-06
Epoch 13/50
72/72 [==============================] - ETA: 0s - loss: 0.0042 - accuracy: 0.9985
Epoch 13: val_loss improved from 1.03214 to 0.72721, saving model to .mdl_wts.hdf5
72/72 [==============================] - 37s 508ms/step - loss: 0.0042 - accuracy: 0.9985 - val_loss: 0.7272 - val_accuracy: 0.7664 - lr: 2.7000e-06
Epoch 14/50
```



```
Epoch 15/50
72/72 [==============================] - ETA: 0s - loss: 0.0051 - accuracy: 0.9989
Epoch 15: val_loss improved from 0.46934 to 0.26504, saving model to .mdl_wts.hdf5
72/72 [==============================] - 37s 504ms/step - loss: 0.0051 - accuracy: 0.9989 - val_loss: 0.2650 - val_accuracy: 0.9143 - lr: 2.7000e-06
Epoch 16/50
72/72 [==============================] - ETA: 0s - loss: 0.0024 - accuracy: 0.9996
Epoch 16: val_loss improved from 0.26504 to 0.15689, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 500ms/step - loss: 0.0024 - accuracy: 0.9996 - val_loss: 0.1569 - val_accuracy: 0.9510 - lr: 2.7000e-06
Epoch 17/50
72/72 [==============================] - ETA: 0s - loss: 0.0039 - accuracy: 0.9989
Epoch 17: val_loss improved from 0.15689 to 0.10569, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 501ms/step - loss: 0.0039 - accuracy: 0.9989 - val_loss: 0.1057 - val_accuracy: 0.9676 - lr: 2.7000e-06
Epoch 18/50
72/72 [==============================] - ETA: 0s - loss: 0.0032 - accuracy: 0.9993
Epoch 18: val_loss improved from 0.10569 to 0.08366, saving model to .mdl_wts.hdf5
72/72 [==============================] - 36s 498ms/step - loss: 0.0032 - accuracy: 0.9993 - val_loss: 0.0837 - val_accuracy: 0.9764 - lr: 2.7000e-06
Epoch 19/50
72/72 [==============================] - ETA: 0s - loss: 0.0038 - accuracy: 0.9987
Epoch 19: val_loss improved from 0.08366 to 0.07891, saving model to .mdl_wts.hdf5
72/72 [==============================] - 37s 504ms/step - loss: 0.0038 - accuracy: 0.9987 - val_loss: 0.0789 - val_accuracy: 0.9781 - lr: 2.7000e-06
Epoch 20/50
72/72 [==============================] - ETA: 0s - loss: 0.0028 - accuracy: 0.9993
Epoch 20: val_loss improved from 0.07891 to 0.07833, saving model to .mdl_wts.hdf5
72/72 [==============================] - 37s 503ms/step - loss: 0.0028 - accuracy: 0.9993 - val_loss: 0.0783 - val_accuracy: 0.9808 - lr: 2.7000e-06
Epoch 21/50
72/72 [==============================] - ETA: 0s - loss: 0.0035 - accuracy: 0.9991
Epoch 21: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 498ms/step - loss: 0.0035 - accuracy: 0.9991 - val_loss: 0.0803 - val_accuracy: 0.9781 - lr: 2.7000e-06
Epoch 22/50
72/72 [==============================] - ETA: 0s - loss: 0.0026 - accuracy: 0.9993
Epoch 22: val_loss did not improve from 0.07833

Epoch 22: ReduceLROnPlateau reducing learning rate to 8.099999604382901e-07.
72/72 [==============================] - 36s 502ms/step - loss: 0.0026 - accuracy: 0.9993 - val_loss: 0.0824 - val_accuracy: 0.9781 - lr: 2.7000e-06
Epoch 23/50
72/72 [==============================] - ETA: 0s - loss: 0.0019 - accuracy: 1.0000
Epoch 23: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 491ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0832 - val_accuracy: 0.9781 - lr: 8.1000e-07
Epoch 24/50
72/72 [==============================] - ETA: 0s - loss: 0.0036 - accuracy: 0.9991
Epoch 24: val_loss did not improve from 0.07833

Epoch 24: ReduceLROnPlateau reducing learning rate to 2.4299998813148704e-07.
72/72 [==============================] - 37s 505ms/step - loss: 0.0036 - accuracy: 0.9991 - val_loss: 0.0832 - val_accuracy: 0.9781 - lr: 8.1000e-07
Epoch 25/50
72/72 [==============================] - ETA: 0s - loss: 0.0019 - accuracy: 1.0000
Epoch 25: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 489ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0838 - val_accuracy: 0.9790 - lr: 2.4300e-07
Epoch 26/50
72/72 [==============================] - ETA: 0s - loss: 0.0037 - accuracy: 0.9989
Epoch 26: val_loss did not improve from 0.07833

Epoch 26: ReduceLROnPlateau reducing learning rate to 7.289999643944612e-08.
72/72 [==============================] - 35s 490ms/step - loss: 0.0037 - accuracy: 0.9989 - val_loss: 0.0842 - val_accuracy: 0.9790 - lr: 2.4300e-07
Epoch 27/50
72/72 [==============================] - ETA: 0s - loss: 0.0028 - accuracy: 0.9993
Epoch 27: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 492ms/step - loss: 0.0028 - accuracy: 0.9993 - val_loss: 0.0844 - val_accuracy: 0.9790 - lr: 7.2900e-08
Epoch 28/50
```



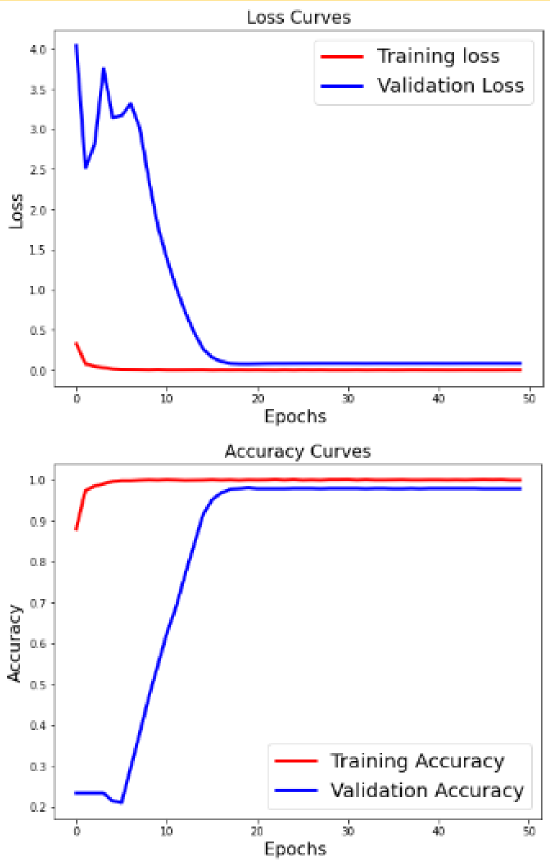
```
Epoch 29/50
72/72 [==============================] - ETA: 0s - loss: 0.0022 - accuracy: 0.9998
Epoch 29: val_loss did not improve from 0.07833
72/72 [==============================] - 38s 519ms/step - loss: 0.0022 - accuracy: 0.9998 - val_loss: 0.0847 - val_accuracy: 0.9790 - lr: 2.1870e-08
Epoch 30/50
72/72 [==============================] - ETA: 0s - loss: 0.0022 - accuracy: 0.9998
Epoch 30: val_loss did not improve from 0.07833

Epoch 30: ReduceLROnPlateau reducing learning rate to 6.560999210591944e-09.
72/72 [==============================] - 37s 513ms/step - loss: 0.0022 - accuracy: 0.9998 - val_loss: 0.0849 - val_accuracy: 0.9790 - lr: 2.1870e-08
Epoch 31/50
72/72 [==============================] - ETA: 0s - loss: 0.0021 - accuracy: 1.0000
Epoch 31: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 494ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0847 - val_accuracy: 0.9790 - lr: 6.5610e-09
Epoch 32/50
72/72 [==============================] - ETA: 0s - loss: 0.0031 - accuracy: 0.9991
Epoch 32: val_loss did not improve from 0.07833

Epoch 32: ReduceLROnPlateau reducing learning rate to 1.968299789822936e-09.
72/72 [==============================] - 36s 491ms/step - loss: 0.0031 - accuracy: 0.9991 - val_loss: 0.0845 - val_accuracy: 0.9790 - lr: 6.5610e-09
Epoch 33/50
72/72 [==============================] - ETA: 0s - loss: 0.0019 - accuracy: 0.9998
Epoch 33: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 491ms/step - loss: 0.0019 - accuracy: 0.9998 - val_loss: 0.0840 - val_accuracy: 0.9781 - lr: 1.9683e-09
Epoch 34/50
72/72 [==============================] - ETA: 0s - loss: 0.0026 - accuracy: 0.9991
Epoch 34: val_loss did not improve from 0.07833

Epoch 34: ReduceLROnPlateau reducing learning rate to 5.904899236242044e-10.
72/72 [==============================] - 36s 494ms/step - loss: 0.0026 - accuracy: 0.9991 - val_loss: 0.0845 - val_accuracy: 0.9790 - lr: 1.9683e-09
Epoch 35/50
72/72 [==============================] - ETA: 0s - loss: 0.0030 - accuracy: 0.9991
Epoch 35: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 491ms/step - loss: 0.0030 - accuracy: 0.9991 - val_loss: 0.0846 - val_accuracy: 0.9790 - lr: 5.9049e-10
Epoch 36/50
72/72 [==============================] - ETA: 0s - loss: 0.0021 - accuracy: 0.9993
Epoch 36: val_loss did not improve from 0.07833

Epoch 36: ReduceLROnPlateau reducing learning rate to 1.771469804179304e-10.
72/72 [==============================] - 36s 490ms/step - loss: 0.0021 - accuracy: 0.9993 - val_loss: 0.0843 - val_accuracy: 0.9781 - lr: 5.9049e-10
Epoch 37/50
72/72 [==============================] - ETA: 0s - loss: 0.0026 - accuracy: 0.9996
Epoch 37: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 488ms/step - loss: 0.0026 - accuracy: 0.9996 - val_loss: 0.0843 - val_accuracy: 0.9781 - lr: 1.7715e-10
Epoch 38/50
72/72 [==============================] - ETA: 0s - loss: 0.0036 - accuracy: 0.9991
Epoch 38: val_loss did not improve from 0.07833

Epoch 38: ReduceLROnPlateau reducing learning rate to 5.314409329271186e-11.
72/72 [==============================] - 35s 489ms/step - loss: 0.0036 - accuracy: 0.9991 - val_loss: 0.0845 - val_accuracy: 0.9790 - lr: 1.7715e-10
Epoch 39/50
72/72 [==============================] - ETA: 0s - loss: 0.0034 - accuracy: 0.9991
Epoch 39: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 488ms/step - loss: 0.0034 - accuracy: 0.9991 - val_loss: 0.0843 - val_accuracy: 0.9781 - lr: 5.3144e-11
Epoch 40/50
72/72 [==============================] - ETA: 0s - loss: 0.0028 - accuracy: 0.9993
Epoch 40: val_loss did not improve from 0.07833
```



```
Epoch 39/50
72/72 [==============================] - ETA: 0s - loss: 0.0034 - accuracy: 0.9991
Epoch 39: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 488ms/step - loss: 0.0034 - accuracy: 0.9991 - val_loss: 0.0843 - val_accuracy: 0.9781 - lr: 5.3144e-11
Epoch 40/50
72/72 [==============================] - ETA: 0s - loss: 0.0028 - accuracy: 0.9993
Epoch 40: val_loss did not improve from 0.07833

Epoch 40: ReduceLROnPlateau reducing learning rate to 1.5943227987813556e-11.
72/72 [==============================] - 36s 490ms/step - loss: 0.0028 - accuracy: 0.9993 - val_loss: 0.0846 - val_accuracy: 0.9790 - lr: 5.3144e-11
Epoch 41/50
72/72 [==============================] - ETA: 0s - loss: 0.0029 - accuracy: 0.9993
Epoch 41: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 487ms/step - loss: 0.0029 - accuracy: 0.9993 - val_loss: 0.0843 - val_accuracy: 0.9790 - lr: 1.5943e-11
Epoch 42/50
72/72 [==============================] - ETA: 0s - loss: 0.0023 - accuracy: 0.9991
Epoch 42: val_loss did not improve from 0.07833

Epoch 42: ReduceLROnPlateau reducing learning rate to 1e-11.
72/72 [==============================] - 36s 495ms/step - loss: 0.0023 - accuracy: 0.9991 - val_loss: 0.0841 - val_accuracy: 0.9790 - lr: 1.5943e-11
Epoch 43/50
72/72 [==============================] - ETA: 0s - loss: 0.0025 - accuracy: 0.9993
Epoch 43: val_loss did not improve from 0.07833
72/72 [==============================] - 38s 524ms/step - loss: 0.0025 - accuracy: 0.9993 - val_loss: 0.0844 - val_accuracy: 0.9790 - lr: 1.0000e-11
Epoch 44/50
72/72 [==============================] - ETA: 0s - loss: 0.0029 - accuracy: 0.9991
Epoch 44: val_loss did not improve from 0.07833
72/72 [==============================] - 37s 515ms/step - loss: 0.0029 - accuracy: 0.9991 - val_loss: 0.0846 - val_accuracy: 0.9790 - lr: 1.0000e-11
Epoch 45/50
72/72 [==============================] - ETA: 0s - loss: 0.0027 - accuracy: 0.9996
Epoch 45: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 486ms/step - loss: 0.0027 - accuracy: 0.9996 - val_loss: 0.0849 - val_accuracy: 0.9790 - lr: 1.0000e-11
Epoch 46/50
72/72 [==============================] - ETA: 0s - loss: 0.0024 - accuracy: 0.9998
Epoch 46: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 491ms/step - loss: 0.0024 - accuracy: 0.9998 - val_loss: 0.0848 - val_accuracy: 0.9781 - lr: 1.0000e-11
Epoch 47/50
72/72 [==============================] - ETA: 0s - loss: 0.0024 - accuracy: 0.9996
Epoch 47: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 492ms/step - loss: 0.0024 - accuracy: 0.9996 - val_loss: 0.0848 - val_accuracy: 0.9781 - lr: 1.0000e-11
Epoch 48/50
72/72 [==============================] - ETA: 0s - loss: 0.0025 - accuracy: 0.9998
Epoch 48: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 491ms/step - loss: 0.0025 - accuracy: 0.9998 - val_loss: 0.0847 - val_accuracy: 0.9781 - lr: 1.0000e-11
Epoch 49/50
72/72 [==============================] - ETA: 0s - loss: 0.0032 - accuracy: 0.9989
Epoch 49: val_loss did not improve from 0.07833
72/72 [==============================] - 36s 494ms/step - loss: 0.0032 - accuracy: 0.9989 - val_loss: 0.0847 - val_accuracy: 0.9781 - lr: 1.0000e-11
Epoch 50/50
72/72 [==============================] - ETA: 0s - loss: 0.0028 - accuracy: 0.9989
Epoch 50: val_loss did not improve from 0.07833
72/72 [==============================] - 35s 483ms/step - loss: 0.0028 - accuracy: 0.9989 - val_loss: 0.0847 - val_accuracy: 0.9781 - lr: 1.0000e-11
```

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()
```

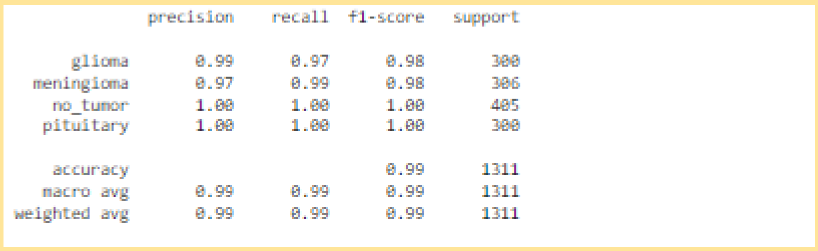


Loading Model :-

```
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 :-

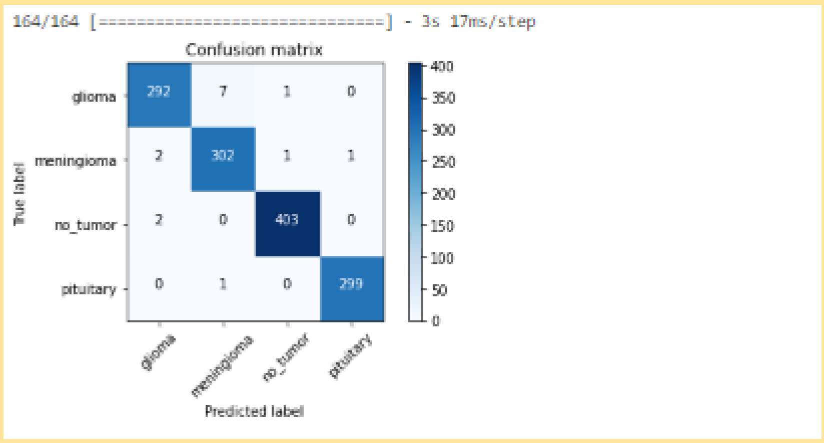
```
import seaborn as sns
predicted_classes = np.argmax(model.predict(x_test), axis = 1)
print(classification_report(np.argmax(y_test,axis=1),
predicted_classes,target_names=['glioma','meningioma','no_tumor','pi
tuitary']))
```



```
              precision    recall  f1-score   support

      glioma       0.99      0.97      0.98       300
  meningioma       0.97      0.99      0.98       306
    no_tumor       1.00      1.00      1.00       405
   pituitary       1.00      1.00      1.00       300

    accuracy                           0.99      1311
   macro avg       0.99      0.99      0.99      1311
weighted avg       0.99      0.99      0.99      1311
```

```python
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
```

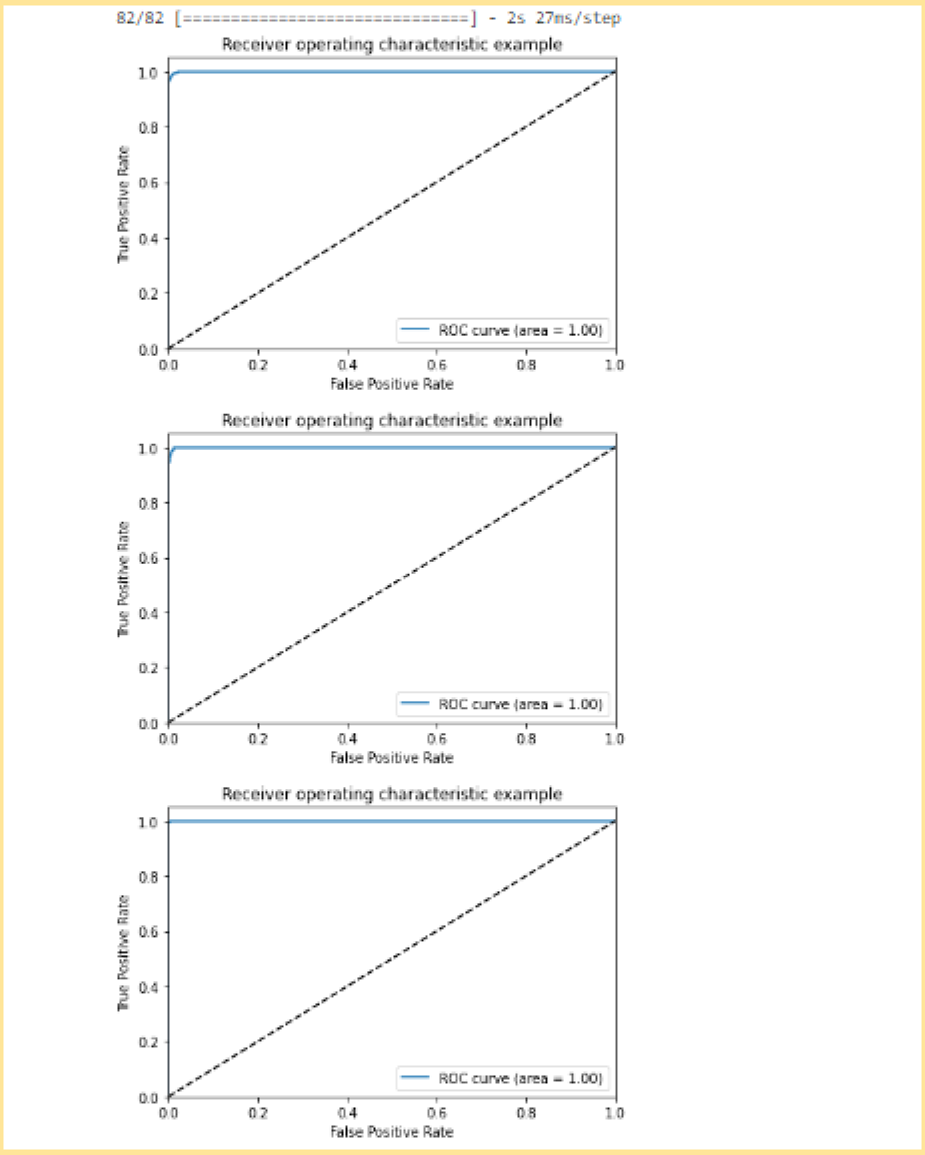
```
plot_confusion_matrix(confusion_mtx, classes = range(4))
```



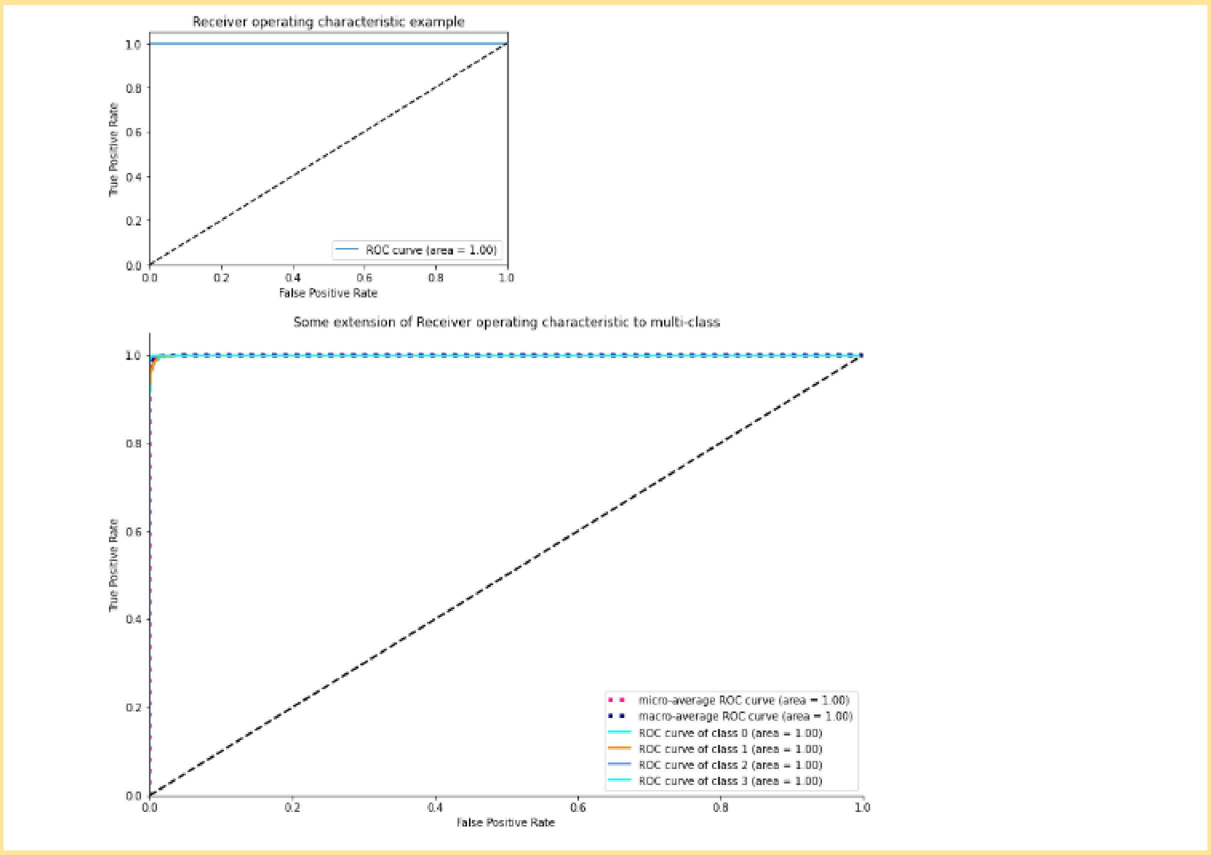
```
164/164 [==============================] - 3s 17ms/step
```

```
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()
```

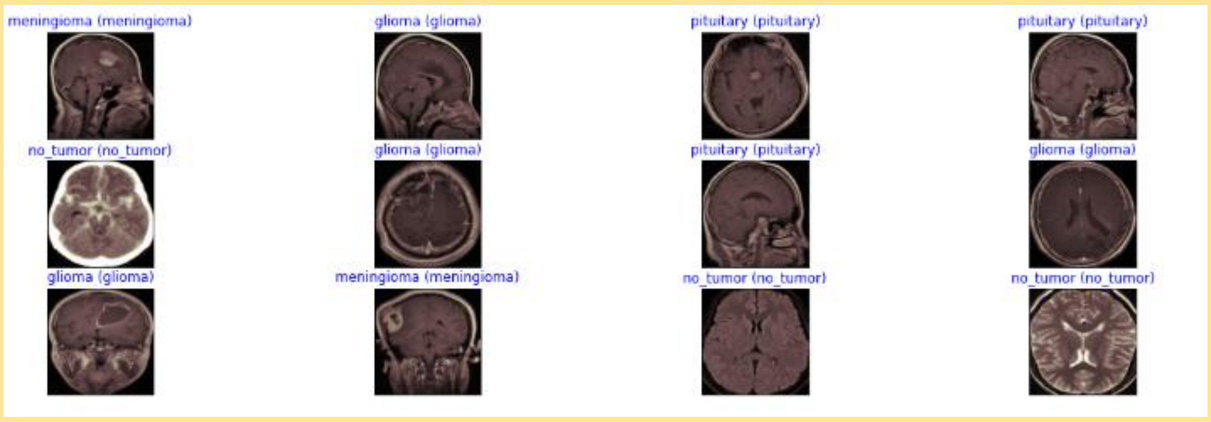


```
82/82 [==============================] - 2s 27ms/step
```



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:-

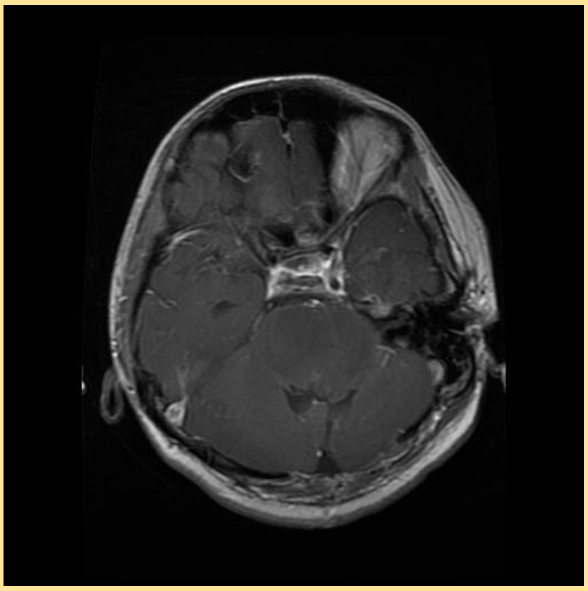
The main goal of this research work is to design efficient automatic brain tumor classification with high accuracy, performance and low complexity. In the conventional brain tumor classification is performed by using Fuzzy C Means (FCM) based segmentation, texture and shape feature extraction and SVM 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:-

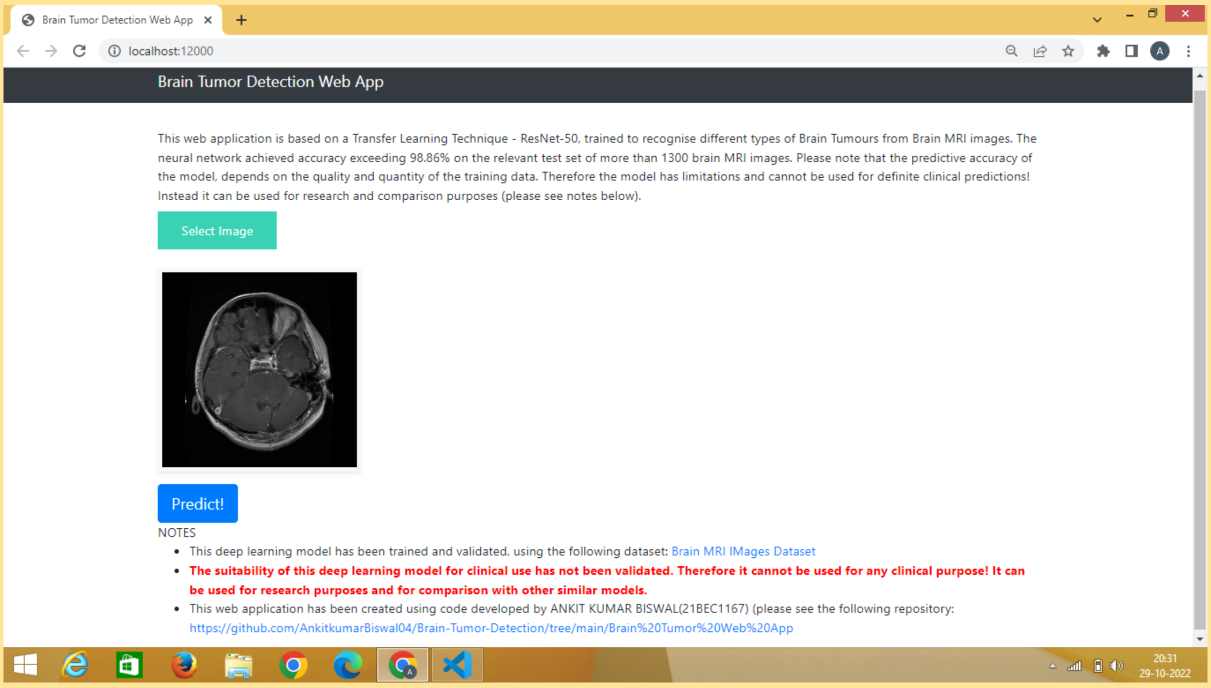
This web application is based on a Transfer Learning Technique - ResNet-50, trained to recognise different types of Brain Tumours from Brain MRI images. The neural network achieved accuracy exceeding 98.86% on the relevant test set of more than 1300 brain MRI images. Please note that the predictive accuracy 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 :-



Brain Tumor Detection Web App

localhost:12000

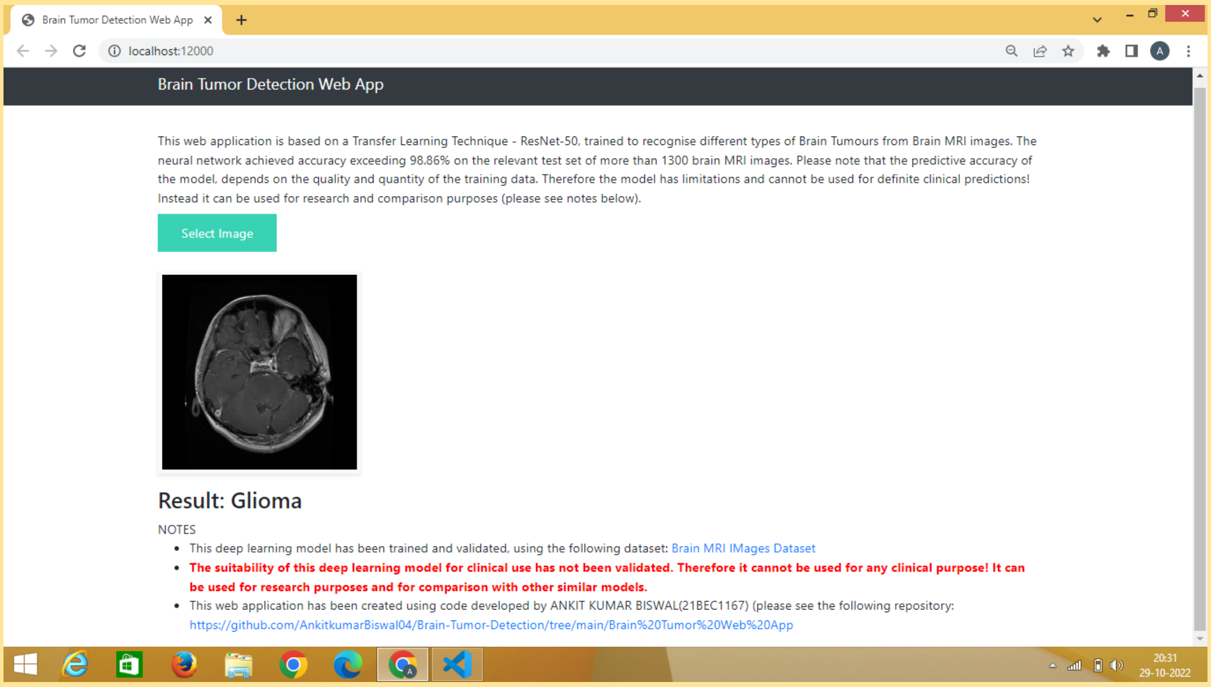
Brain Tumor Detection Web App

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 (please see notes below).

NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

After selecting the image the predicted result is :-



Brain Tumor Detection Web App

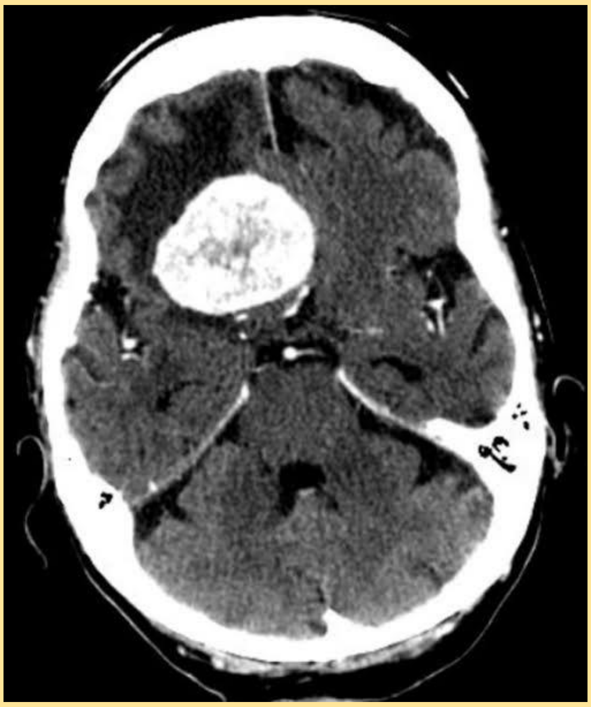
This web application is based on a Transfer Learning Technique - ResNet-50, trained to recognise different types of Brain Tumours from Brain MRI images. The neural network achieved accuracy exceeding 98.86% on the relevant test set of more than 1300 brain MRI images. Please note that the predictive accuracy 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 (please see notes below).

Result: Glioma

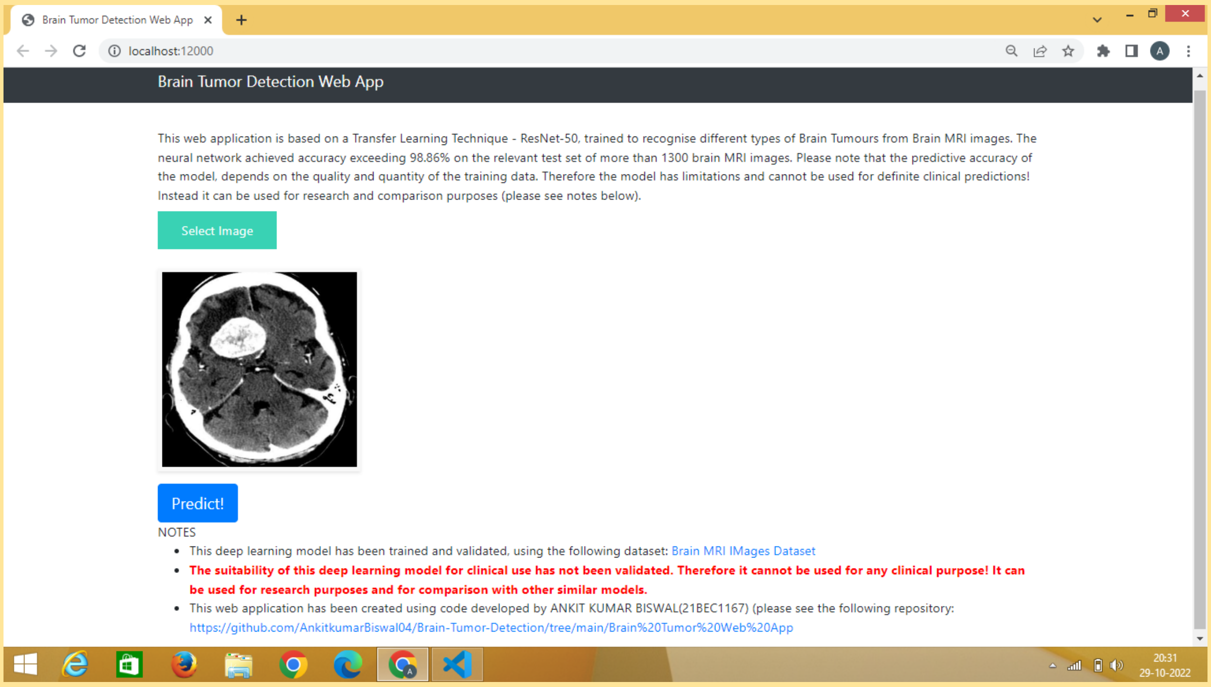
NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

This is for Meningioma case :-



Selecting the image :-



Brain Tumor Detection Web App

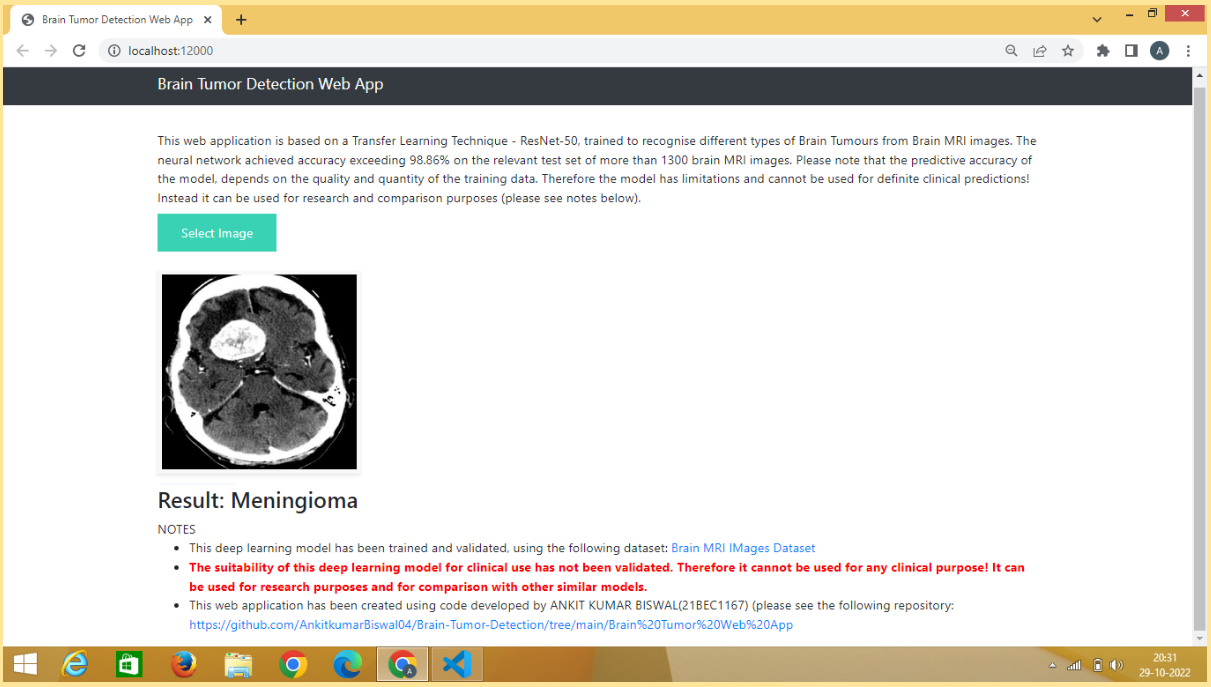
This web application is based on a Transfer Learning Technique - ResNet-50, trained to recognise different types of Brain Tumours from Brain MRI images. The neural network achieved accuracy exceeding 98.86% on the relevant test set of more than 1300 brain MRI images. Please note that the predictive accuracy 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 (please see notes below).

NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

20:31
29-10-2022

After selecting the image the predicted result is :-



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 (please see notes below).

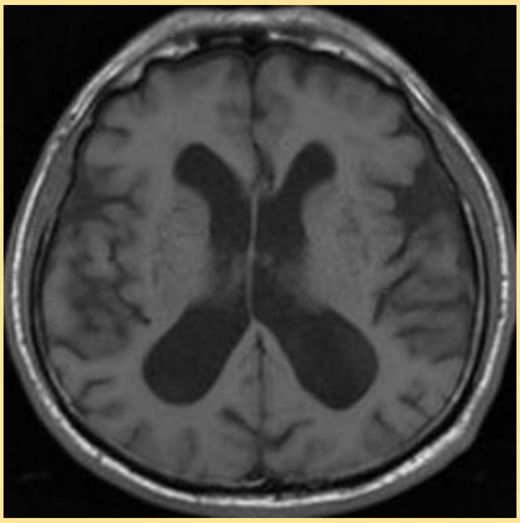
Result: Meningioma

NOTES

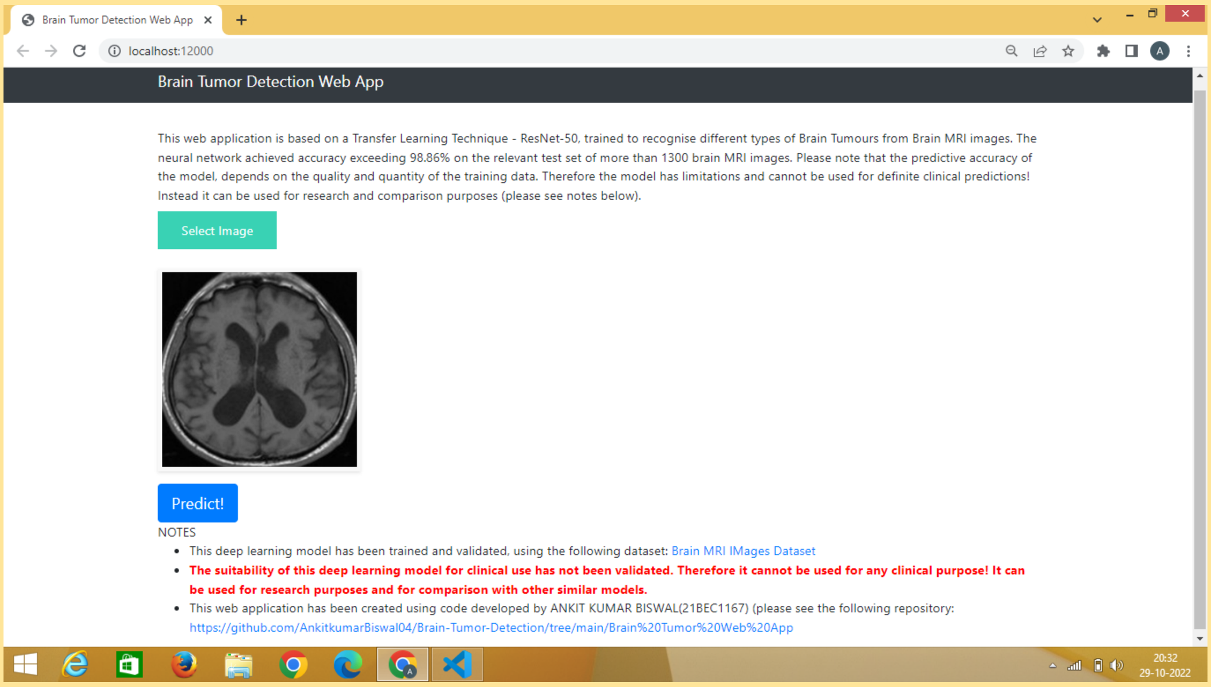
- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

20:31
29-10-2022

This is for No Tumor case :-



Selecting the image :-



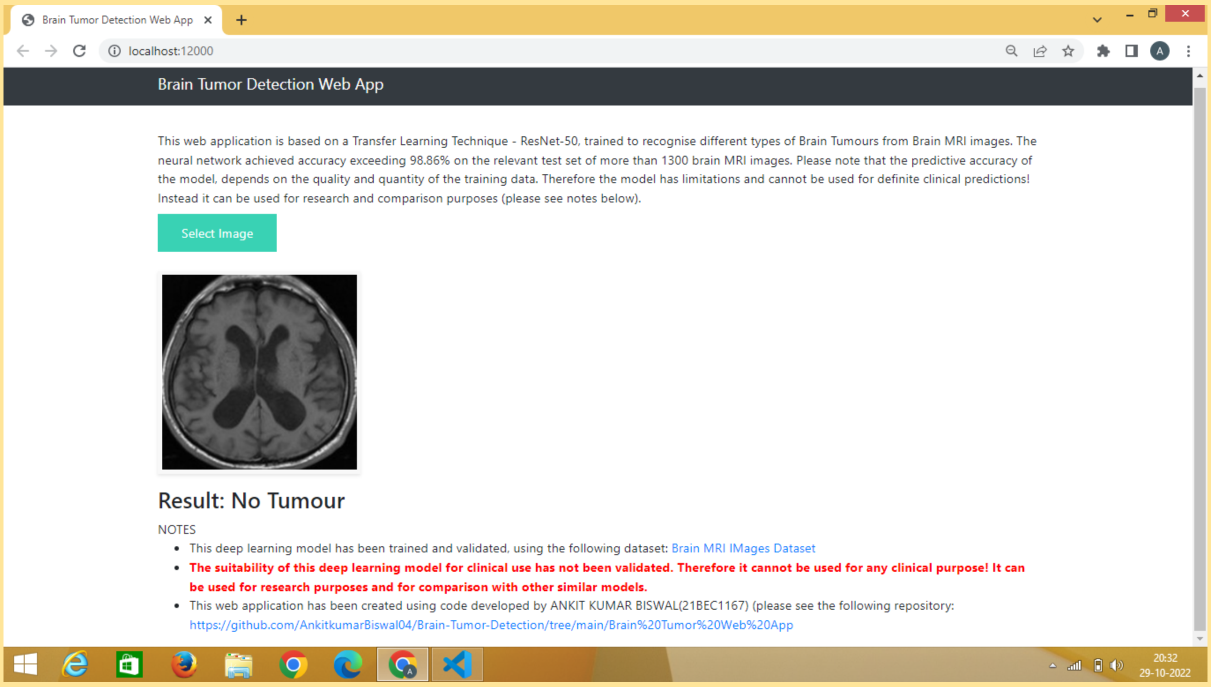
Brain Tumor Detection Web App

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 (please see notes below).

NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

After selecting the image the predicted result is :-



Brain Tumor Detection Web App

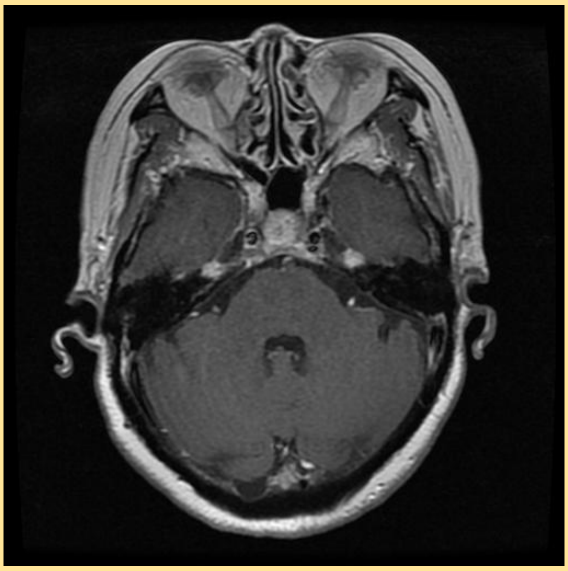
This web application is based on a Transfer Learning Technique - ResNet-50, trained to recognise different types of Brain Tumours from Brain MRI images. The neural network achieved accuracy exceeding 98.86% on the relevant test set of more than 1300 brain MRI images. Please note that the predictive accuracy 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 (please see notes below).

Result: No Tumour

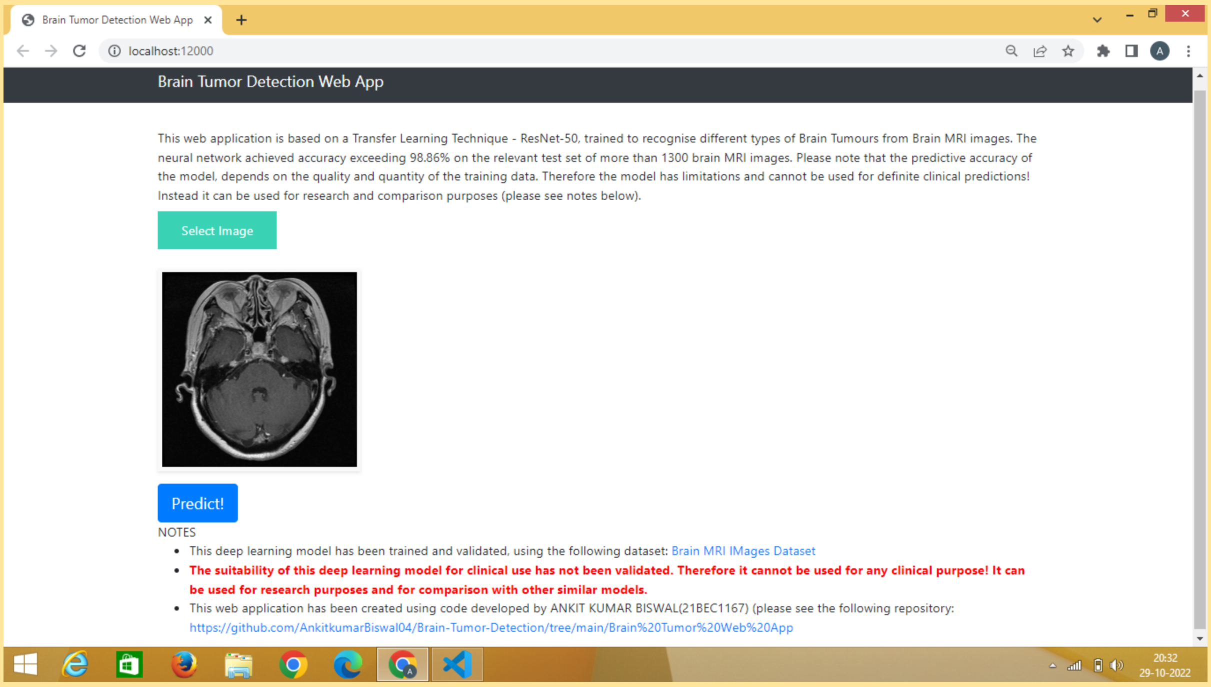
NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

This is for Pituitary case:-



Selecting the image :-



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 (please see notes below).

Select Image

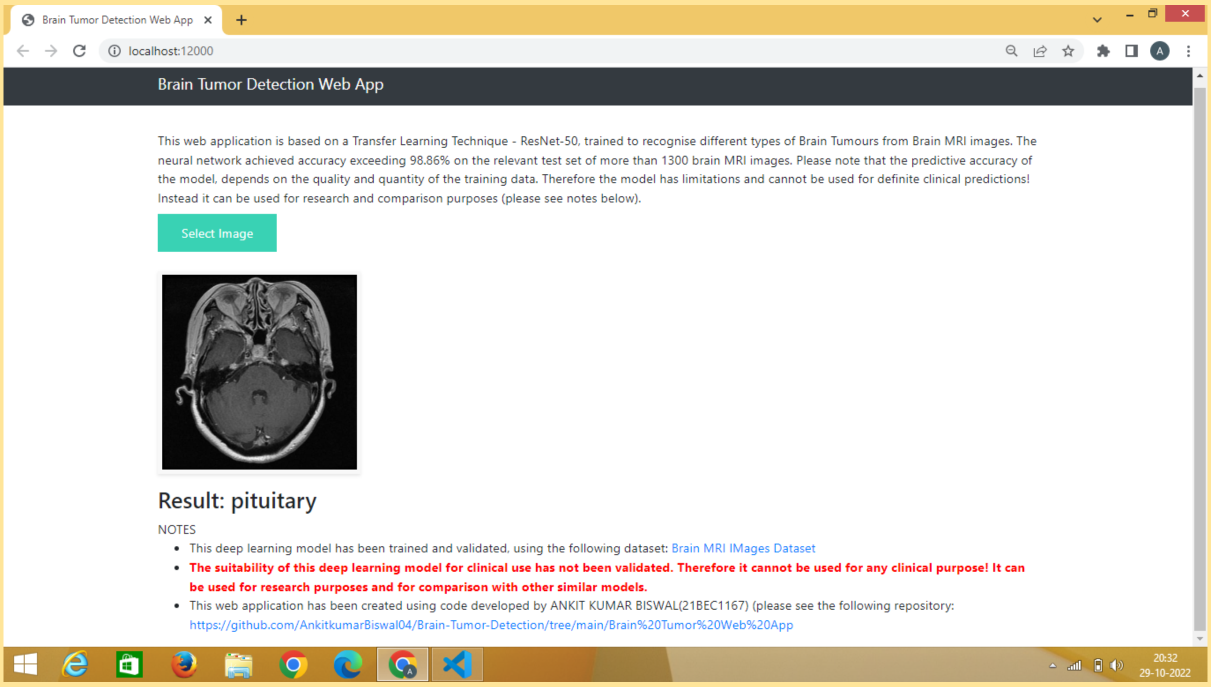
Predict!

NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

20:32 29-10-2022

After selecting the image the predicted result is :-



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 (please see notes below).

Select Image

Result: pituitary

NOTES

- This deep learning model has been trained and validated, using the following dataset: Brain MRI IMages Dataset
- **The suitability of this deep learning model for clinical use has not been validated. Therefore it cannot be used for any clinical purpose! It can be used for research purposes and for comparison with other similar models.**
- This web application has been created using code developed by ANKIT KUMAR BISWAL(21BEC1167) (please see the following repository: https://github.com/AnkitkumarBiswal04/Brain-Tumor-Detection/tree/main/Brain%20Tumor%20Web%20App

20:32 29-10-2022

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
- 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
- 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.
- 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 MathSciNet Google Scholar
- 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