```
!pip install imutils       # A series of functions to make basic image processing operation such as rotation,resizing etc

import numpy as np
from tqdm import tqdm      #tqdm is used for progress bar
import cv2                 #cv2 is an openCV library
import os                  # OS is used to interact with the operating system(Example:File system)
import shutil              #It is used for copy,create and remote operation on file.
import itertools           #It is used for memory efficient and precise code of iterating objects in code
import imutils
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools

from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16, preprocess_input
from keras import layers
from keras.models import Model, Sequential
from keras.callbacks import EarlyStopping

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

```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25860 sh
  Stored in directory: /root/.cache/pip/wheels/86/d7/0a/4923351ed1cec5d5e24c1eaf89
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
WARNING: Running pip as the 'root' user can result in broken permissions and confl
```

```
import os
from scipy.io import loadmat
import h5py
import numpy as np
from PIL import Image
import cv2

#Dataset Path
IMG_PATH = '../input/dataset3/All_images'

# Creating Directories to store images  into folders

!mkdir dataset_classified_for_training dataset_classified_for_training/meningioma
!mkdir dataset_classified_for_training/glioma
!mkdir dataset_classified_for_training/pituitary
!mkdir dataset_classified_for_training/no_tumor
!mkdir dataset_classified_for_validation dataset_classified_for_validation/meningioma
!mkdir dataset_classified_for_validation/glioma
!mkdir dataset_classified_for_validation/pituitary
!mkdir dataset_classified_for_validation/no_tumor
!mkdir dataset_classified_for_testing dataset_classified_for_testing/meningioma
!mkdir dataset_classified_for_testing/glioma
!mkdir dataset_classified_for_testing/pituitary
!mkdir dataset_classified_for_testing/no_tumor


total_files = 0
```

```
type_one = 0
type_two = 0
type_three = 0
type_four = 0

for img_name in os.listdir(IMG_PATH):
    h5_file = h5py.File(f'{IMG_PATH}/{img_name}', 'r') #formatted string literal f', OPENING MAT FILE FOR READING
    cjdata = h5_file['cjdata']                         # CJDATA is a MATLAB struct , a type of matrix
    image = np.array(cjdata.get('image')).astype(np.float64)
    label = cjdata.get('label')[0,0]

    #Dividing label 1 into training (80%) ,validation (10%) and testing(10%)

    if label == 1.0:
        type_one += 1       # To read number of type 1 images
        if type_one % 10 <= 6:
            png = f'./dataset_classified_for_training/meningioma/{img_name}'[:-3]+"png"
        elif type_one % 10 <= 8 and type_one % 10 >= 7:
            png = f'./dataset_classified_for_validation/meningioma/{img_name}'[:-3]+"png"
        elif type_one % 10 <= 9:
            png = f'./dataset_classified_for_testing/meningioma/{img_name}'[:-3]+"png"

    #Dividing label 2 into training (80%) ,validation (10%) and testing(10%)

    elif label == 2.0:
        type_two += 1      # To read number of type 2 images
        if type_two % 10 <= 6:
            png = f'./dataset_classified_for_training/glioma/{img_name}'[:-3]+"png"
        elif type_two % 10 <= 8 and type_two % 10 >= 7:
            png = f'./dataset_classified_for_validation/glioma/{img_name}'[:-3]+"png"
        elif type_two % 10 <= 9:
            png = f'./dataset_classified_for_testing/glioma/{img_name}'[:-3]+"png"

    #Dividing label 3 into training (80%) ,validation (10%) and testing(10%)

    elif label == 3.0:
        type_three += 1    # To read number of type 3 images
        if type_three % 10 <= 6:
            png = f'./dataset_classified_for_training/pituitary/{img_name}'[:-3]+"png"
        elif type_three % 10 <= 8 and type_three % 10 >= 7:
            png = f'./dataset_classified_for_validation/pituitary/{img_name}'[:-3]+"png"
        elif type_three % 10 <= 9:
            png = f'./dataset_classified_for_testing/pituitary/{img_name}'[:-3]+"png"

    #Dividing label 4 into training (80%) ,validation (10%) and testing(10%)

    else:
        type_four += 1     # To read number of type 4 images
        if type_four % 10 <= 6:
            png = f'./dataset_classified_for_training/no_tomor/{img_name}'[:-3]+"png"
        elif type_four % 10 <= 8 and type_four % 10 >= 7:
            png = f'./dataset_classified_for_validation/no_tomor/{img_name}'[:-3]+"png"
        elif type_four % 10 <= 9:
            png = f'./dataset_classified_for_testing/no_tomor/{img_name}'[:-3]+"png"


    tumorBorder = np.array(cjdata.get('tumorBorder'))[0]
    tumorMask = np.array(cjdata.get('tumorMask'))
    h5_file.close()
    hi = np.max(image)
    lo = np.min(image)
    image = (((image - lo)/(hi-lo))*255).astype(np.uint8)
    im = Image.fromarray(image)
    im.save(png)
    total_files += 1
    print("saving", png, "File No: ", total_files, "type:", label)

print("Finished converting all files: ", total_files)
print("Total type four images :",type_four)  #new changes
```

```
saving ./dataset_classified_for_training/pituitary/1590.png File No:  93 type: 3.0
saving ./dataset_classified_for_training/pituitary/1831.png File No:  94 type: 3.0
saving ./dataset_classified_for_validation/pituitary/1260.png File No:  95 type: 3.0
saving ./dataset_classified_for_validation/pituitary/1547.png File No:  96 type: 3.0
saving ./dataset_classified_for_validation/meningioma/398.png File No:  97 type: 1.0
saving ./dataset_classified_for_training/glioma/1949.png File No:  98 type: 2.0
saving ./dataset_classified_for_testing/meningioma/263.png File No:  99 type: 1.0
saving ./dataset_classified_for_training/meningioma/131.png File No:  100 type: 1.0
saving ./dataset_classified_for_training/glioma/2913.png File No:  101 type: 2.0
saving ./dataset_classified_for_training/meningioma/306.png File No:  102 type: 1.0
saving ./dataset_classified_for_training/glioma/2316.png File No:  103 type: 2.0
saving ./dataset_classified_for_testing/pituitary/1832.png File No:  104 type: 3.0
saving ./dataset_classified_for_training/meningioma/512.png File No:  105 type: 1.0
saving ./dataset_classified_for_training/pituitary/1615.png File No:  106 type: 3.0
saving ./dataset_classified_for_training/glioma/2854.png File No:  107 type: 2.0
saving ./dataset_classified_for_training/pituitary/1522.png File No:  108 type: 3.0
saving ./dataset_classified_for_training/pituitary/1839.png File No:  109 type: 3.0
saving ./dataset_classified_for_training/pituitary/1538.png File No:  110 type: 3.0
saving ./dataset_classified_for_training/meningioma/314.png File No:  111 type: 1.0
saving ./dataset_classified_for_training/glioma/2878.png File No:  112 type: 2.0
saving ./dataset_classified_for_training/meningioma/409.png File No:  113 type: 1.0
saving ./dataset_classified_for_training/pituitary/961.png File No:  114 type: 3.0
saving ./dataset_classified_for_validation/glioma/2755.png File No:  115 type: 2.0
saving ./dataset_classified_for_training/meningioma/656.png File No:  116 type: 1.0
saving ./dataset_classified_for_validation/glioma/2350.png File No:  117 type: 2.0
saving ./dataset_classified_for_testing/glioma/898.png File No:  118 type: 2.0
saving ./dataset_classified_for_training/glioma/2957.png File No:  119 type: 2.0
saving ./dataset_classified_for_training/glioma/2280.png File No:  120 type: 2.0
saving ./dataset_classified_for_training/glioma/2591.png File No:  121 type: 2.0
saving ./dataset_classified_for_training/pituitary/1482.png File No:  122 type: 3.0
saving ./dataset_classified_for_training/pituitary/1722.png File No:  123 type: 3.0
saving ./dataset_classified_for_training/glioma/828.png File No:  124 type: 2.0
saving ./dataset_classified_for_validation/pituitary/1729.png File No:  125 type: 3.0
saving ./dataset_classified_for_training/glioma/2590.png File No:  126 type: 2.0
saving ./dataset_classified_for_training/glioma/2260.png File No:  127 type: 2.0
saving ./dataset_classified_for_training/meningioma/219.png File No:  128 type: 1.0
saving ./dataset_classified_for_validation/meningioma/299.png File No:  129 type: 1.0
saving ./dataset_classified_for_training/glioma/2453.png File No:  130 type: 2.0
saving ./dataset_classified_for_validation/glioma/836.png File No:  131 type: 2.0
saving ./dataset_classified_for_validation/glioma/878.png File No:  132 type: 2.0
saving ./dataset_classified_for_validation/meningioma/16.png File No:  133 type: 1.0
saving ./dataset_classified_for_testing/meningioma/191.png File No:  134 type: 1.0
saving ./dataset_classified_for_testing/glioma/759.png File No:  135 type: 2.0
saving ./dataset_classified_for_training/meningioma/617.png File No:  136 type: 1.0
saving ./dataset_classified_for_training/glioma/3049.png File No:  137 type: 2.0
saving ./dataset_classified_for_training/glioma/2337.png File No:  138 type: 2.0
saving ./dataset_classified_for_training/glioma/2030.png File No:  139 type: 2.0
saving ./dataset_classified_for_validation/pituitary/1330.png File No:  140 type: 3.0
saving ./dataset_classified_for_training/glioma/783.png File No:  141 type: 2.0
saving ./dataset_classified_for_training/glioma/1854.png File No:  142 type: 2.0
saving ./dataset_classified_for_testing/pituitary/1671.png File No:  143 type: 3.0
saving ./dataset_classified_for_training/pituitary/1306.png File No:  144 type: 3.0
saving ./dataset_classified_for_training/glioma/2546.png File No:  145 type: 2.0
saving ./dataset_classified_for_training/pituitary/933.png File No:  146 type: 3.0
saving ./dataset_classified_for_training/meningioma/667.png File No:  147 type: 1.0
saving ./dataset_classified_for_training/meningioma/251.png File No:  148 type: 1.0
saving ./dataset_classified_for_training/glioma/827.png File No:  149 type: 2.0
saving ./dataset_classified_for_training/meningioma/541.png File No:  150 type: 1.0
```

```
#Dividing the no tumor into training ,validation and testing

no_tumor_image_path = '../input/datasetno/no_tumor'
for img_name in os.listdir(no_tumor_image_path):
    type_four += 1
    if type_four % 10 <= 6:
        jpg = f'./dataset_classified_for_training/no_tumor/{img_name}'+".jpg"
    elif type_four % 10 <= 8 and type_four % 10 >= 7:
        jpg = f'./dataset_classified_for_validation/no_tumor/{img_name}'+".jpg"
    elif type_four % 10 <= 9:
        jpg = f'./dataset_classified_for_testing/no_tumor/{img_name}'+".jpg"
    shutil.copy(no_tumor_image_path+'/'+img_name, jpg)
    total_files += 1
    print("saving", jpg, "File No: ", total_files, "type:", label)
print("Finished converting all type 4 files: ", type_four)
print("total files: ", total_files)
```

```
saving ./dataset_classified_for_training/no_tumor/image(265).jpg.jpg File No:  3097 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(289).jpg.jpg File No:  3098 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(278).jpg.jpg File No:  3099 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(215).jpg.jpg File No:  3100 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(155).jpg.jpg File No:  3101 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(184).jpg.jpg File No:  3102 type: 3.0
saving ./dataset_classified_for_testing/no_tumor/image (40).jpg.jpg File No:  3103 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (26).jpg.jpg File No:  3104 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(193).jpg.jpg File No:  3105 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(74).jpg.jpg File No:  3106 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(35).jpg.jpg File No:  3107 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(6).jpg.jpg File No:  3108 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(15).jpg.jpg File No:  3109 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(132).jpg.jpg File No:  3110 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(135).jpg.jpg File No:  3111 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image (23).jpg.jpg File No:  3112 type: 3.0
saving ./dataset_classified_for_testing/no_tumor/image(168).jpg.jpg File No:  3113 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(81).jpg.jpg File No:  3114 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(204).jpg.jpg File No:  3115 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(207).jpg.jpg File No:  3116 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(13).jpg.jpg File No:  3117 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(293).jpg.jpg File No:  3118 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(226).jpg.jpg File No:  3119 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(123).jpg.jpg File No:  3120 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(237).jpg.jpg File No:  3121 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image (34).jpg.jpg File No:  3122 type: 3.0
saving ./dataset_classified_for_testing/no_tumor/image (39).jpg.jpg File No:  3123 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(25).jpg.jpg File No:  3124 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(143).jpg.jpg File No:  3125 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(102).jpg.jpg File No:  3126 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(313).jpg.jpg File No:  3127 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (31).jpg.jpg File No:  3128 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (53).jpg.jpg File No:  3129 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (46).jpg.jpg File No:  3130 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(325).jpg.jpg File No:  3131 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(79).jpg.jpg File No:  3132 type: 3.0
saving ./dataset_classified_for_testing/no_tumor/image(88).jpg.jpg File No:  3133 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (49).jpg.jpg File No:  3134 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(5).jpg.jpg File No:  3135 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(234).jpg.jpg File No:  3136 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(197).jpg.jpg File No:  3137 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (37).jpg.jpg File No:  3138 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image (50).jpg.jpg File No:  3139 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(292).jpg.jpg File No:  3140 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(288).jpg.jpg File No:  3141 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(53).jpg.jpg File No:  3142 type: 3.0
saving ./dataset_classified_for_testing/no_tumor/image (24).jpg.jpg File No:  3143 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(94).jpg.jpg File No:  3144 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(294).jpg.jpg File No:  3145 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(150).jpg.jpg File No:  3146 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(14).jpg.jpg File No:  3147 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(148).jpg.jpg File No:  3148 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(176).jpg.jpg File No:  3149 type: 3.0
saving ./dataset_classified_for_training/no_tumor/image(248).jpg.jpg File No:  3150 type: 3.0
saving ./dataset_classified_for_validation/no_tumor/image(32).jpg.jpg File No:  3151 type: 3.0
```

```
print(os.listdir('./dataset_classified_for_training/'))
print(os.listdir('./dataset_classified_for_validation/'))
print(os.listdir('./dataset_classified_for_testing/'))

print(len(os.listdir('./dataset_classified_for_training/meningioma')))
print(len(os.listdir('./dataset_classified_for_validation/meningioma')))
print(len(os.listdir('./dataset_classified_for_testing/meningioma')))
print(len(os.listdir('./dataset_classified_for_training/meningioma'))+len(os.listdir('./dataset_classified_for_validation/me
print(type_one)
print(type_two)
print(type_three)
print(type_four)
print(type_one+type_two+type_three+type_four)
```

```
['glioma', 'meningioma', 'pituitary', 'no_tumor']
['glioma', 'meningioma', 'pituitary', 'no_tumor']
['glioma', 'meningioma', 'pituitary', 'no_tumor']
496
142
70
708
708
1426
930
```

```
395
3459
```

```
# Function to load data into workspace

def load_data(dir_path, img_size=(100,100)):

    X = []    # X is for storing images
    y = []    # y is for storing labels
    i = 0
    labels = dict()  # Labels stores different classes (4 Classes total)
    for path in tqdm(sorted(os.listdir(dir_path))):
        print(path)
        if not path.startswith('.'):
            labels[i] = path
            print(len(os.listdir(dir_path + path)))
            for file in os.listdir(dir_path + path):
                if not file.startswith('.'):
                    img = cv2.imread(dir_path + path + '/' + file)
                    X.append(img)
                    y.append(i)
            i += 1
    X = np.array(X)
    y = np.array(y)
    print(f'{len(X)} images loaded from {dir_path} directory.')
    print(labels)   # Printing all classes of images
    return X, y, labels

# For 0 => Glioma
# For 1 => Meningioma
# For 2 => No Tumor
# For 3 => Pituitary

#Function for plotting confusion matrix



TRAIN_DIR = './dataset_classified_for_training/'
TEST_DIR = './dataset_classified_for_testing/'
VAL_DIR = './dataset_classified_for_validation/'
IMG_SIZE = (224,224)

# use predefined function to load the image data into workspace
X_train, y_train, labels = load_data(TRAIN_DIR, IMG_SIZE)
X_test, y_test, _ = load_data(TEST_DIR, IMG_SIZE)
X_val, y_val, _ = load_data(VAL_DIR, IMG_SIZE)
```

```
  0%|          | 0/4 [00:00<?, ?it/s]glioma
1000
 25%|██▌       | 1/4 [00:03<00:10,  3.66s/it]meningioma
496
 50%|█████     | 2/4 [00:05<00:05,  2.64s/it]no_tumor
278
 75%|███████▌  | 3/4 [00:06<00:01,  1.69s/it]pituitary
651
100%|██████████| 4/4 [00:08<00:00,  2.23s/it]
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:20: VisibleDeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shap

2425 images loaded from ./dataset_classified_for_training/ directory.
{0: 'glioma', 1: 'meningioma', 2: 'no_tumor', 3: 'pituitary'}
  0%|          | 0/4 [00:00<?, ?it/s]glioma
142
 25%|██▌       | 1/4 [00:00<00:01,  1.79it/s]meningioma
70
 75%|███████▌  | 3/4 [00:00<00:00,  3.54it/s]no_tumor
39
pituitary
93
100%|██████████| 4/4 [00:01<00:00,  2.63it/s]
344 images loaded from ./dataset_classified_for_testing/ directory.
{0: 'glioma', 1: 'meningioma', 2: 'no_tumor', 3: 'pituitary'}
  0%|          | 0/4 [00:00<?, ?it/s]glioma
284
```

```
 25%|██▌       | 1/4 [00:01<00:03,  1.08s/it]meningioma
142
 50%|█████     | 2/4 [00:01<00:01,  1.30it/s]no_tumor
78
 75%|███████▌  | 3/4 [00:01<00:00,  1.95it/s]pituitary
186
100%|██████████| 4/4 [00:02<00:00,  1.54it/s]690 images loaded from ./dataset_classified_for_validation/ directory.
{0: 'glioma', 1: 'meningioma', 2: 'no_tumor', 3: 'pituitary'}
```

```
print(y_train)
y = dict()
y[0] = []
y[1] = []
y[2] = []
y[3] = []

for set_name in (y_train, y_val, y_test):
    y[0].append(np.sum(set_name == 0))
    y[1].append(np.sum(set_name == 1))
    y[2].append(np.sum(set_name == 2))
    y[3].append(np.sum(set_name == 3))

print(y[0])  #new changes
print(y[1])
print(y[2])
print(y[3])

trace0 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[0],
    name='Glioma',
    marker=dict(color='#00FF00'),
    opacity=0.7
)
trace1 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[1],
    name='Meningioma',
    marker=dict(color='#0000FF'),
    opacity=0.7
)
trace2 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[2],
    name='No_tumor',
    marker=dict(color='#FFFF00'),
    opacity=0.7
)
trace3 = go.Bar(
    x=['Train Set', 'Validation Set', 'Test Set'],
    y=y[3],
    name='Pituitary',
    marker=dict(color='#00FFFF'),
    opacity=0.7
)
data = [trace0, trace1, trace2, trace3]
layout = go.Layout(
    title='Count of classes in each set',
    xaxis={'title': 'Set'},
    yaxis={'title': 'Count'}
)
fig = go.Figure(data, layout)
iplot(fig)
```

```
[0 0 0 ... 3 3 3]
[1000, 284, 142]
[496, 142, 70]
[278, 78, 39]
[651, 186, 93]
```

```
# Displays selected number of images

def plot_samples(X, y, labels_dict, n=50):

    for index in range(len(labels_dict)):
        imgs = X[np.argwhere(y == index)][:n]
        j = 10
        i = int(n/j)

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

            plt.xticks([])
            plt.yticks([])
            c += 1
        plt.suptitle('Tumor: {}'.format(labels_dict[index]))
        plt.show()

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


```
# This function locates the extreme poins and crops the images
def crop_imgs(set_name, add_pixels_value=0):

    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

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

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

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

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

        return np.array(set_new)


# apply this for each set
X_train_crop = crop_imgs(set_name=X_train)
X_val_crop = crop_imgs(set_name=X_val)
X_test_crop = crop_imgs(set_name=X_test)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:30: VisibleDeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shap
```

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


```
# Function to save all cropped images

def save_new_images(x_set, y_set, folder_name):
    i = 0
    for (img, imclass) in zip(x_set, y_set):
        if imclass == 0:
            cv2.imwrite(folder_name+'GLIOMA/'+str(i)+'.jpg', img)
        elif imclass == 1:
            cv2.imwrite(folder_name+'MENINGIOMA/'+str(i)+'.jpg', img)
        elif imclass == 2:
            cv2.imwrite(folder_name+'NO_TUMOR/'+str(i)+'.jpg', img)
        else:
            cv2.imwrite(folder_name+'PITUITARY/'+str(i)+'.jpg', img)
        i += 1
```

```
# saving new images to the folder

!mkdir TRAIN_CROP TRAIN_CROP/GLIOMA TRAIN_CROP/MENINGIOMA TRAIN_CROP/NO_TUMOR TRAIN_CROP/PITUITARY
!mkdir VAL_CROP VAL_CROP/GLIOMA VAL_CROP/MENINGIOMA VAL_CROP/NO_TUMOR VAL_CROP/PITUITARY
!mkdir TEST_CROP TEST_CROP/GLIOMA TEST_CROP/MENINGIOMA TEST_CROP/NO_TUMOR TEST_CROP/PITUITARY

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

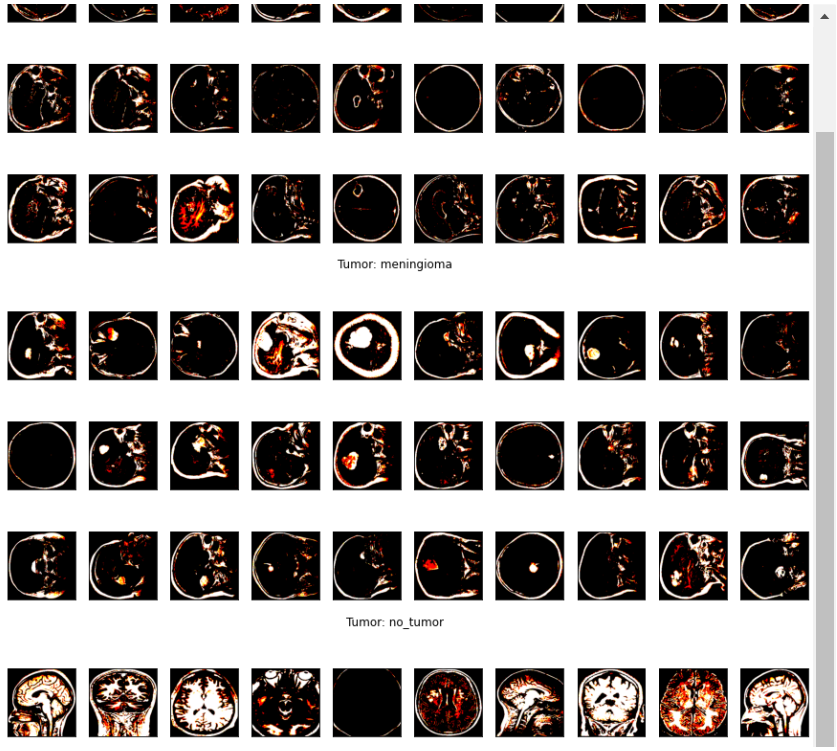
```
#This function prepares the image for vgg-16
def preprocess_imgs(set_name, img_size):

    set_new = []
    for img in set_name:
        img = cv2.resize(
            img,
            dsize=img_size,
            interpolation=cv2.INTER_CUBIC
        )
        set_new.append(preprocess_input(img))
    print(img.shape)   # Printing the size of the pre-processed images
    return np.array(set_new)


X_train_prep = preprocess_imgs(set_name=X_train_crop, img_size=IMG_SIZE)
X_test_prep = preprocess_imgs(set_name=X_test_crop, img_size=IMG_SIZE)
X_val_prep = preprocess_imgs(set_name=X_val_crop, img_size=IMG_SIZE)
```

```
(224, 224, 3)
(224, 224, 3)
(224, 224, 3)
```

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




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

train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    brightness_range=[0.5, 1.5],
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)


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


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

```
Found 2425 images belonging to 4 classes.
Found 690 images belonging to 4 classes.
```

```
from keras.callbacks import Callback
ACCURACY_THRESHOLD = 0.92
class myCallback(Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('acc') is not None and logs.get('acc') >= ACCURACY_THRESHOLD):
            print(ACCURACY_THRESHOLD*100)
            self.model.stop_training = true


from keras.callbacks import ReduceLROnPlateau

 # Callback Functions
callbacks = myCallback()
early_stop=EarlyStopping(patience=3)
reduceLR=ReduceLROnPlateau(patience=2)



from keras.layers import MaxPooling2D,Conv2D,Dense,BatchNormalization,Dropout,GlobalAveragePooling2D,Flatten,Input
import keras as k

def decay(epoch):
    return 0.001 / (1 + 1 * 30)


new_callbacks = []
new_callbacks += [k.callbacks.LearningRateScheduler(decay, verbose=1)]


vgg_model = VGG16(weights='imagenet',include_top=False ,input_shape=(224,224,3))
for layers in vgg_model.layers[:15]:
    layers.trainable=False
x=vgg_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(512,activation='relu')(x)
x=Dropout(0.5)(x)
output=Dense(4,activation='softmax')(x)
model2=Model(inputs=vgg_model.input,outputs=output)
model2.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58892288/58889256 [==============================] - 2s 0us/step
58900480/58889256 [==============================] - 2s 0us/step
```

```
# To find which layers are trainable
for i,layer in enumerate(vgg_model.layers):
    print(i,layer.name , layer.trainable)
```

```
0 input_1 False
1 block1_conv1 False
2 block1_conv2 False
3 block1_pool False
4 block2_conv1 False
5 block2_conv2 False
6 block2_pool False
7 block3_conv1 False
8 block3_conv2 False
9 block3_conv3 False
10 block3_pool False
11 block4_conv1 False
12 block4_conv2 False
13 block4_conv3 False
14 block4_pool False
15 block5_conv1 True
16 block5_conv2 True
17 block5_conv3 True
18 block5_pool True
```

```
model2.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 2052
=================================================================
Total params: 14,979,396
Trainable params: 7,344,132
Non-trainable params: 7,635,264
_________________________________________________________________
```

```
r2=model2.fit_generator(
    train_generator,
    epochs=40,
    validation_data=validation_generator,
    validation_steps=25,
    callbacks=[new_callbacks],
    verbose=1
)
```

```
Epoch 14/40

Epoch 00014: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 38s 505ms/step - loss: 0.1267 - accuracy: 0.9530 - val_loss: 0.2870 - val_accuracy: 0.9200
Epoch 15/40

Epoch 00015: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 39s 516ms/step - loss: 0.1164 - accuracy: 0.9555 - val_loss: 0.1311 - val_accuracy: 0.9425
Epoch 16/40

Epoch 00016: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 38s 503ms/step - loss: 0.0841 - accuracy: 0.9728 - val_loss: 0.2085 - val_accuracy: 0.9375
Epoch 17/40

Epoch 00017: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 39s 510ms/step - loss: 0.0703 - accuracy: 0.9753 - val_loss: 0.1390 - val_accuracy: 0.9525
Epoch 18/40

Epoch 00018: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 38s 500ms/step - loss: 0.0627 - accuracy: 0.9757 - val_loss: 0.1865 - val_accuracy: 0.9500
Epoch 19/40

Epoch 00019: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 39s 511ms/step - loss: 0.0638 - accuracy: 0.9786 - val_loss: 0.1452 - val_accuracy: 0.9475
Epoch 20/40

Epoch 00020: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 40s 520ms/step - loss: 0.0666 - accuracy: 0.9802 - val_loss: 0.1166 - val_accuracy: 0.9650
Epoch 21/40

Epoch 00021: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 39s 514ms/step - loss: 0.0433 - accuracy: 0.9856 - val_loss: 0.1395 - val_accuracy: 0.9550
Epoch 22/40

Epoch 00022: LearningRateScheduler setting learning rate to 3.2258064516129034e-05.
76/76 [==============================] - 39s 509ms/step - loss: 0.0529 - accuracy: 0.9786 - val_loss: 0.1381 - val_accuracy: 0.9600
Epoch 23/40
```

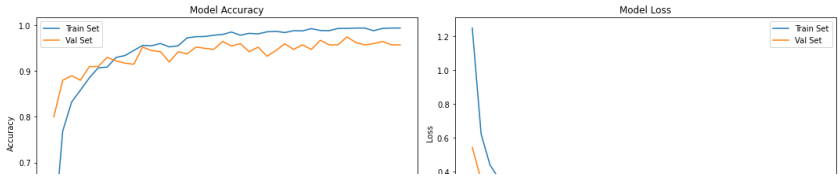
```
# plot model performance
acc = r2.history['accuracy']
val_acc = r2.history['val_accuracy']
loss = r2.history['loss']
val_loss = r2.history['val_loss']
epochs_range = range(1, len(r2.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()
plt.show()
```




```
from sklearn.metrics import classification_report

test_pred_transfer=np.argmax(model2.predict(X_val_prep),axis=1)
print(classification_report(y_val,test_pred_transfer))
```

```
                  precision    recall  f1-score   support

               0       0.96      0.98      0.97       284
               1       0.95      0.88      0.92       142
               2       1.00      0.97      0.99        78
               3       0.95      0.98      0.96       186

        accuracy                           0.96       690
       macro avg       0.96      0.95      0.96       690
    weighted avg       0.96      0.96      0.96       690
```

```
test_pred_transfer=np.argmax(model2.predict(X_test_prep),axis=1)
print(classification_report(y_test,test_pred_transfer))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.96      0.97       142
               1       0.93      0.93      0.93        70
               2       1.00      1.00      1.00        39
               3       0.96      1.00      0.98        93

        accuracy                           0.97       344
       macro avg       0.97      0.97      0.97       344
    weighted avg       0.97      0.97      0.97       344
```

```
def inverse_classes(num):
    if num==0:
        return 'Glioma Tumor'
    elif num==1:
        return 'Meningioma Tumor'
    elif num==2:
        return 'No Tumor'
    else:
        return 'Pituitary Tumor'

# Prediction using VGG16 model
plt.figure(figsize=(15,12))
for i in range(4):
    plt.subplot(3,2,(i%12)+1)
    index=np.random.randint(250)
    pred_class=inverse_classes(np.argmax(model2.predict(np.reshape(X_test_prep[index],(-1,224,224,3))),axis=1))
    plt.title('This image is of {0} and is predicted as {1}'.format(inverse_classes(y_test[index]),pred_class),fontdict={'si
    plt.imshow(X_test_prep[index])
    plt.tight_layout()
```