Brain Tumor Classification with ResNet 50 Transfer Learning

A brain tumor is an abnormal growth of tissue in the brain or central spine that can disrupt proper brain function. Doctors refer to a tumor based on where the tumor cells originated, and whether they are cancerous (malignant) or not (benign)

Table of Contents

Introduction

In this notebook, we will learn how to classify images of brain tumor by using transfer learning from a pre-trained network.

What we will learn:

Load the images.

Visulaise the Data distribution of all data.

Visualizing some of the images

Setting up callbacks

Transfer learning

ResNet

Graph the training loss and validation loss

Predict the results

Installing some Libraries

In [2]:
```
!pip install split-folders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Importing Libraries

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
import pathlib
import cv2
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.models import Sequential, Model,load_model
from keras.callbacks import EarlyStopping,ModelCheckpoint
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D,MaxPool2D
from keras.preprocessing import image
from keras.initializers import glorot_uniform
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, cohen_kappa_score, roc_auc_score, confusion_matrix
from sklearn.metrics import classification_report
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D,MaxPool2D,Dropout
import tensorflow as tf
import splitfolders
import pandas as pd
import glob
from sklearn.metrics import confusion_matrix
import itertools
import plotly.graph_objects as go
import plotly.express as px
#Suppressing Warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

Setting path

In [4]:
```
data_dir = "../input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset"
data_dir =pathlib.Path(data_dir)
```

In [5]:
```
Total_Images1 = glob.glob('../input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset/*/*.JPG')
#print("Total number of images: ", len(Total_Images1))

Total_Images2 = glob.glob('../input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset/*/*.jpg')

#print("Total number of images: ", len(Total_Images2))

Total_Images = Total_Images1 + Total_Images2
print("Total number of images: ", len(Total_Images))

Total_Images = pd.Series(Total_Images)
```

```
Total number of images:  245
```

Setting Data Frame

In [6]:
```
total_df = pd.DataFrame()

# generate Filename field
total_df['Filename'] = Total_Images.map( lambda img_name: img_name.split("/")[-1])


# generate ClassId field
total_df['ClassId'] = Total_Images.map(lambda img_name: img_name.split("/")[-2])

total_df.head()
```

Out[6]:

	Filename	ClassId
0	N20.JPG	no
1	N1.JPG	no
2	N26.JPG	no
3	N22.JPG	no
4	N19.JPG	no

In [7]:
```
class_id_distributionTotal = total_df['ClassId'].value_counts()
class_id_distributionTotal.head(10)
```

Out[7]:
```
yes    154
no      91
Name: ClassId, dtype: int64
```

Data Distribution of All Data

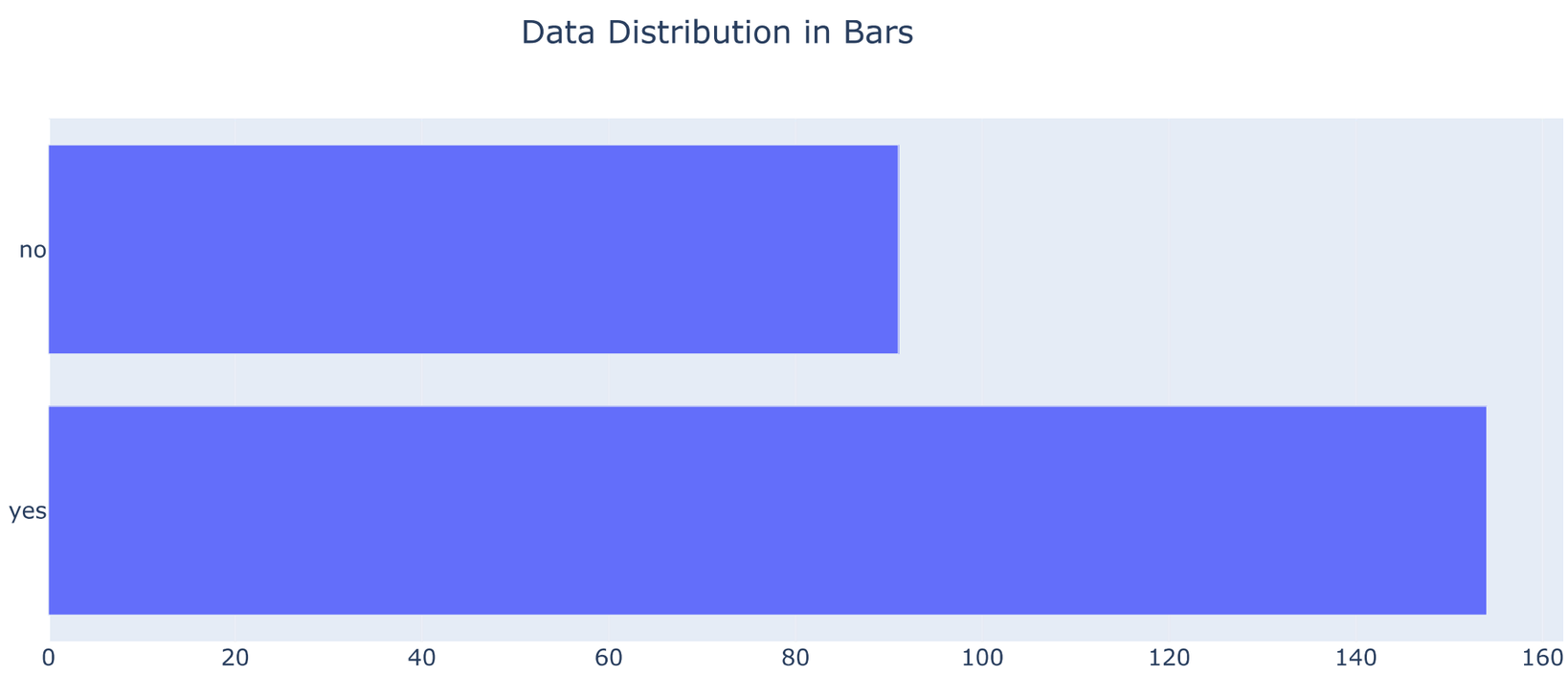
Below chart shows the percentage of data of two different classes.

"Yes" class contains 62.9% from data.

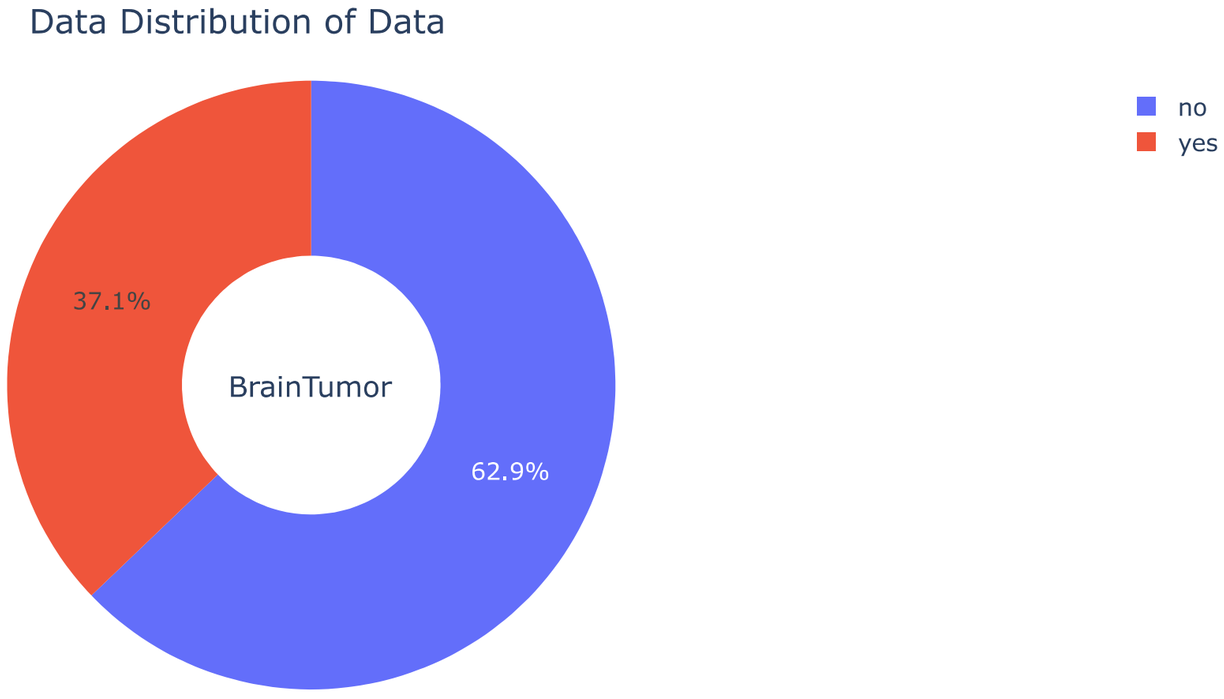
"No" class contains 37.1% from data.

Clearly the data is not equaly distributed.

In [8]:
```
fig = go.Figure(go.Bar(
            x=  class_id_distributionTotal.values,
            y=class_id_distributionTotal.index,
            orientation='h'))

fig.update_layout(title='Data Distribution in Bars',font_size=15,title_x=0.45)


fig.show()
```



In [9]:
```
fig=px.pie(class_id_distributionTotal.head(10),values= 'ClassId', names=total_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Data',font_size=15,title_x=0.45,annotations=[dict(text='BrainTumor',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



Splitting data into Train, Test, Val data set

using split folder so we can divide the data in to three parts.

Training Set contain 80% of data.

Test Set contain 10% of data.

Validation Set contain 10% of data

In [10]:
```
splitfolders.ratio(data_dir, output="output", seed=101, ratio=(.8, .1, .1))
```

```
Copying files: 253 files [00:01, 145.72 files/s]
```

Setting Path for Train , Test, val.

In [11]:
```
train_path='./output/train/'
val_path='./output/val'
test_path='./output/test'
class_names=os.listdir(train_path)
class_names_val=os.listdir(val_path)
class_names_test=os.listdir(test_path)
```

Counting the number of Images in each sub folder

In [12]:
```
import glob
train_image1 = glob.glob('./output/train/*/*.jpg')
train_image2 = glob.glob('./output/train/*/*.JPG')
Total_TrainImages = train_image1 + train_image2
print("Total number of training images: ", len(Total_TrainImages))


test_image1 = glob.glob('./output/test/*/*.jpg')
test_image2 = glob.glob('./output/test/*/*.JPG')
Total_TestImages = test_image1 + test_image2
print("Total number of test images: ", len(Total_TestImages))


Val_image1 = glob.glob('./output/val/*/*.jpg')
Val_image2 = glob.glob('./output/val/*/*.JPG')
Total_ValImages = Val_image1 + Val_image2
print("Total number of val images: ", len(Total_ValImages))
```

```
Total number of training images:  194
Total number of test images:  27
Total number of val images:  24
```

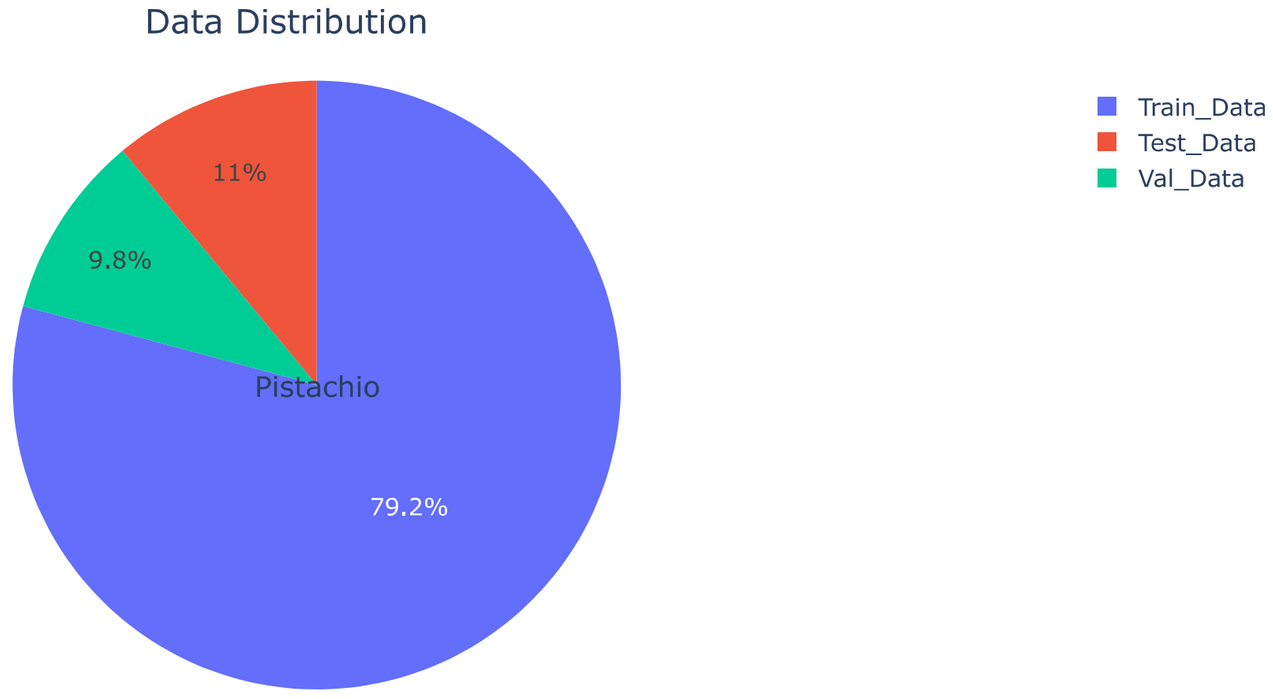
Below Pie Charts Shows

"Trainng Set" contains 79.2% from data.

"Test Set" contains 11% from data..

"Val Set" contains 9% from data.

In [13]:
```
random_x = [len(Total_TrainImages), len(Total_TestImages), len(Total_ValImages)]
names = ['Train_Data', 'Test_Data', 'Val_Data']
fig = px.pie(values=random_x, names=names)
fig.update_layout(title='Data Distribution',font_size=15,title_x=0.45,annotations=[dict(text='Pistachio',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



Making the data frame for Train Data

In [14]:
```
train_image_names = pd.Series(Total_TrainImages)
train_df = pd.DataFrame()

# generate Filename field
train_df['Filename'] = train_image_names.map( lambda img_name: img_name.split("/")[-1])


# generate ClassId field
train_df['ClassId'] = train_image_names.map(lambda img_name: img_name.split("/")[-2])

train_df.head()
```

Out[14]:

	Filename	ClassId
0	Y25.jpg	yes
1	Y96.jpg	yes
2	Y60.jpg	yes
3	Y162.jpg	yes
4	Y3.jpg	yes

In [15]:
```
class_id_distribution_Train = train_df['ClassId'].value_counts()
class_id_distribution_Train.head(10)
```

Out[15]:
```
yes    123
no      71
Name: ClassId, dtype: int64
```

Visulaize in form of bar

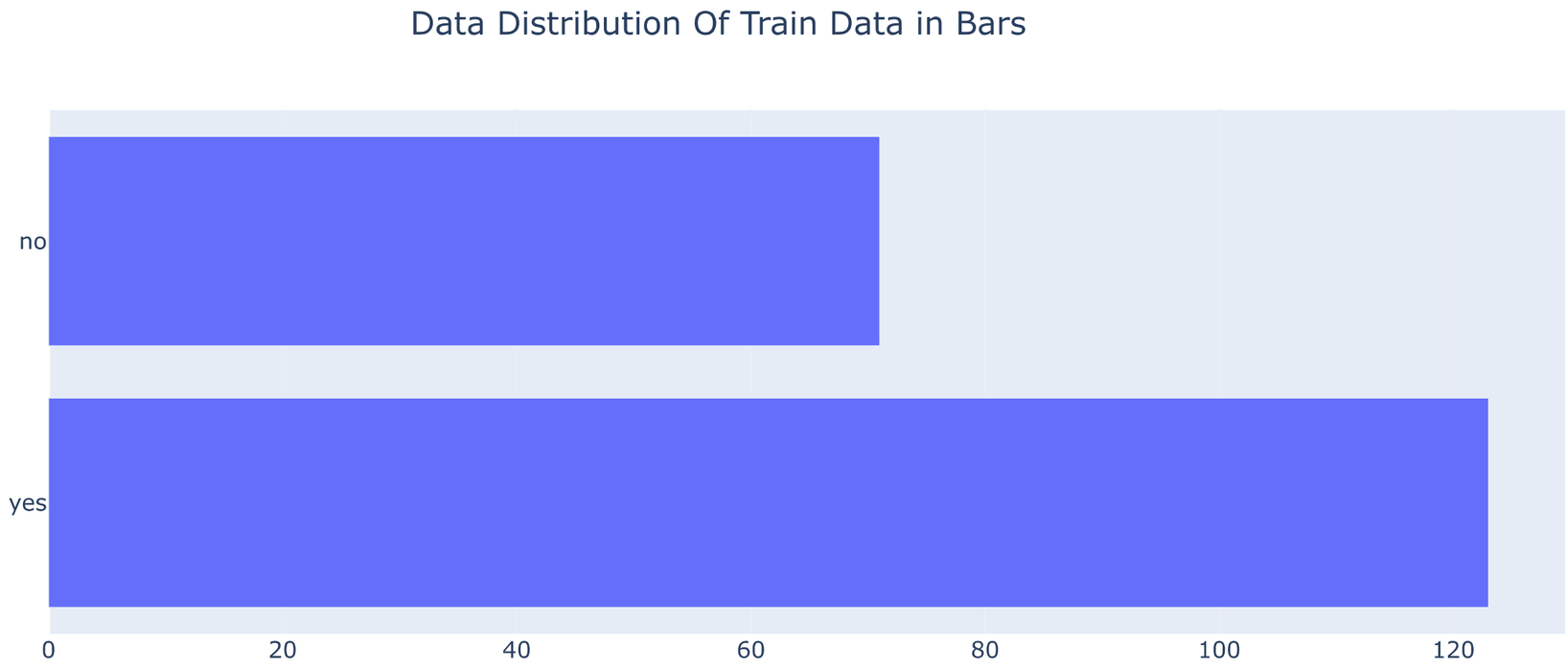
Data Distribution of Train Data

Below chart shows the percentage of data of two different classes in training data set.

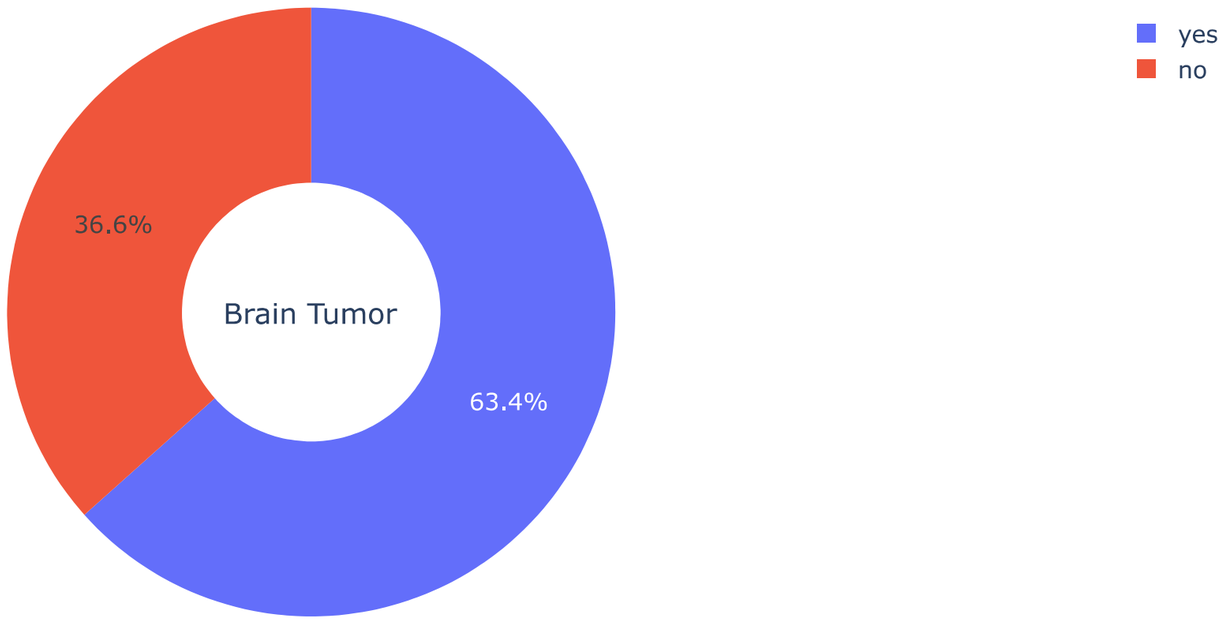
"Yes" class contains 63.4% from data.

"No" class contains 36.6% from data.

In [16]:
```
fig = go.Figure(go.Bar(
            x= class_id_distribution_Train.values,
            y=class_id_distribution_Train.index,
            orientation='h'))

fig.update_layout(title='Data Distribution Of Train Data in Bars',font_size=15,title_x=0.45)

fig.show()
```



In [17]:
```
fig=px.pie(class_id_distribution_Train.head(10),values= 'ClassId', names=train_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Train Data in Pie Chart',font_size=15,title_x=0.45,annotations=[dict(text='Brain Tumor',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```



Making Data Frame of Test Data

In [18]:
```
test_image_names = pd.Series(Total_TestImages)
test_df = pd.DataFrame()

# generate Filename field
test_df['Filename'] = test_image_names.map( lambda img_name: img_name.split("/")[-1])


# generate ClassId field
test_df['ClassId'] = test_image_names.map(lambda img_name: img_name.split("/")[-2])

test_df.head()
```

Out[18]:

	Filename	ClassId
0	Y187.jpg	yes
1	Y79.jpg	yes
2	Y185.jpg	yes
3	Y92.jpg	yes
4	Y29.jpg	yes

In [19]:
```
class_id_distribution_test = test_df['ClassId'].value_counts()
class_id_distribution_test.head(10)
```

Out[19]:
```
yes    16
no     11
Name: ClassId, dtype: int64
```

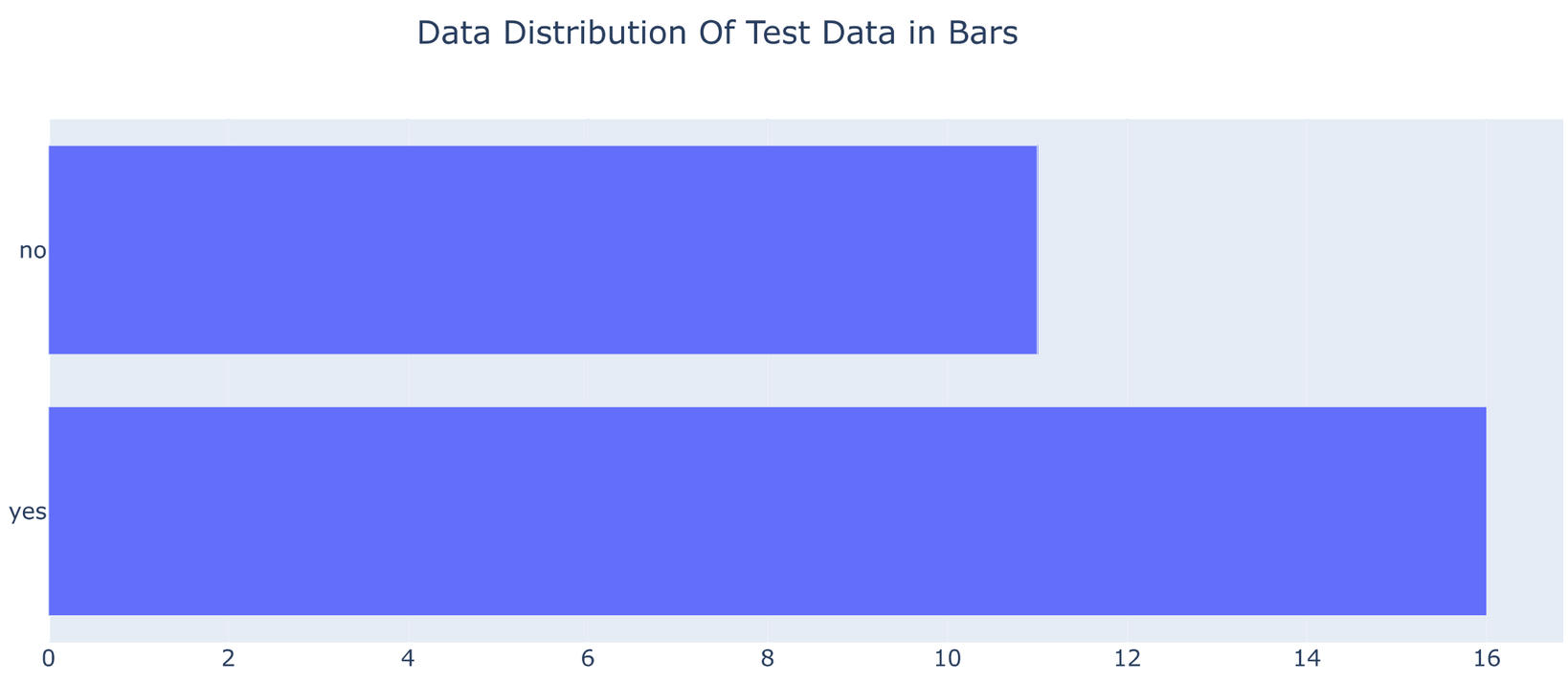
Data Distribution of Test Data

Below chart shows the percentage of data of two different classes.

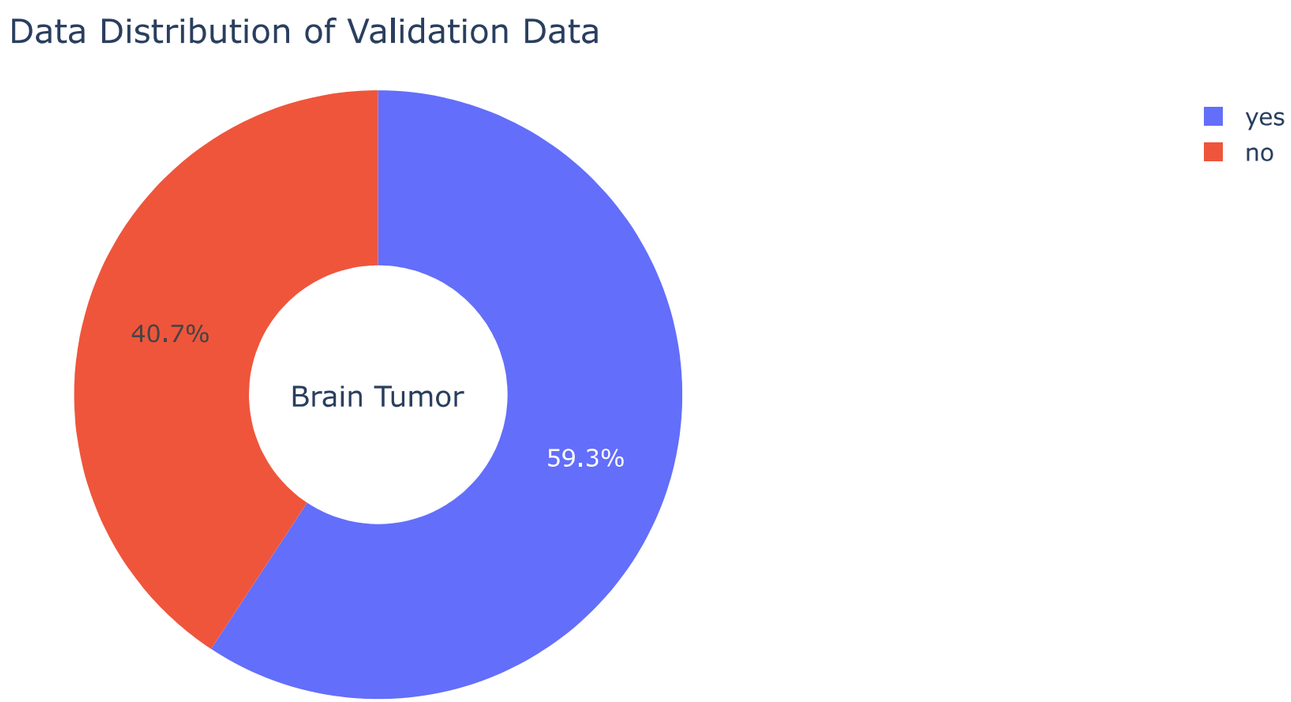
"Yes" class contains 59.3% from data.

"No" class contains 40.7% from data.

In [20]:
```
fig = go.Figure(go.Bar(
            x=class_id_distribution_test.values,
            y=class_id_distribution_test.index,
            orientation='h'))

fig.update_layout(title='Data Distribution Of Test Data in Bars',font_size=15,title_x=0.45)

fig.show()
```

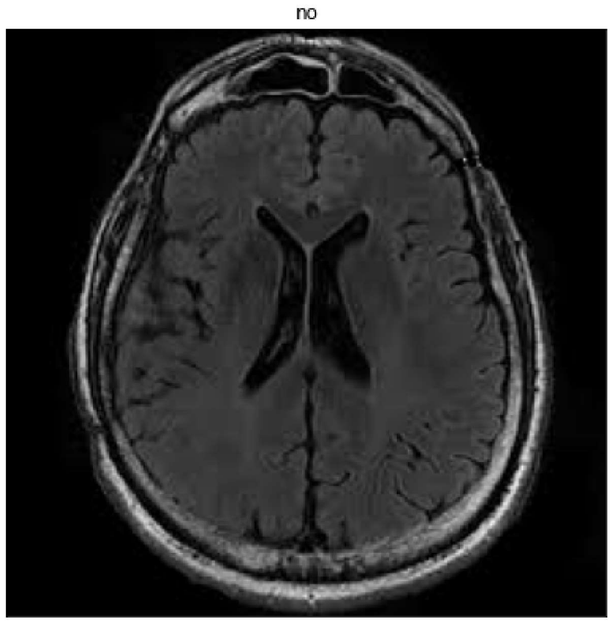


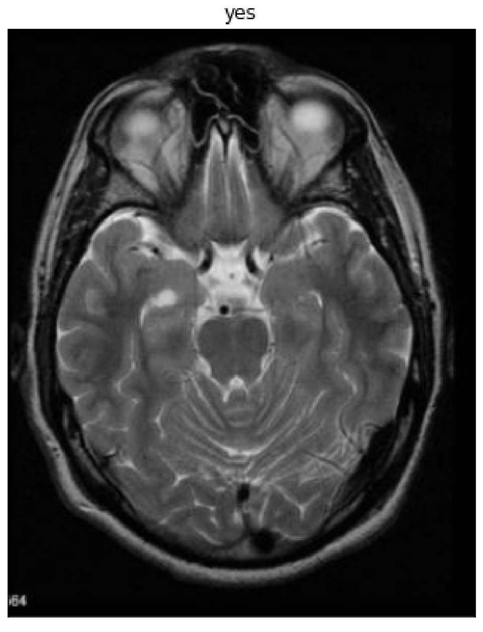
In [21]:
```
fig=px.pie(class_id_distribution_test.head(10),values= 'ClassId', names=test_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Validation Data',font_size=15,title_x=0.45,annotations=[dict(text='Brain Tumor',font_size=18, showarrow=False,height=800,width=700)])
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```

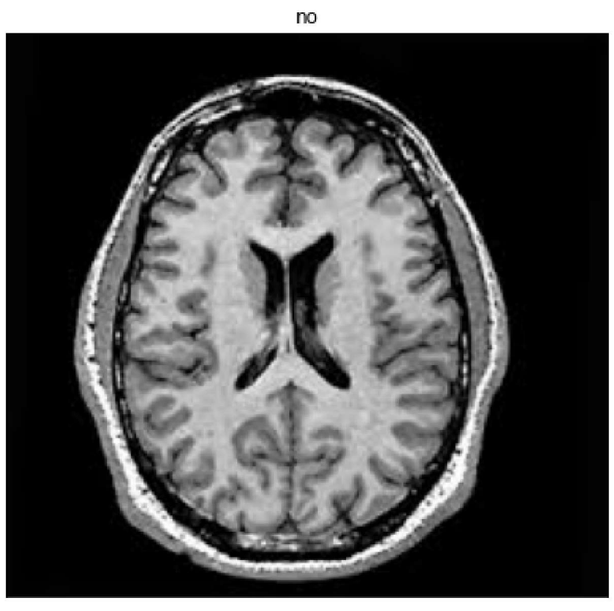


Displaying The Images

In [22]:
```
plot_df = train_df.sample(6).reset_index()
plt.figure(figsize=(15, 15))

for i in range(4):
    img_name = plot_df.loc[i, 'Filename']
    label_str = (plot_df.loc[i, 'ClassId'])
    plt.subplot(2,2,i+1)
    plt.imshow(plt.imread(os.path.join(train_path,label_str, img_name)))
    plt.title(label_str)
    plt.xticks([])
    plt.yticks([])
    plt.yticks([])
```







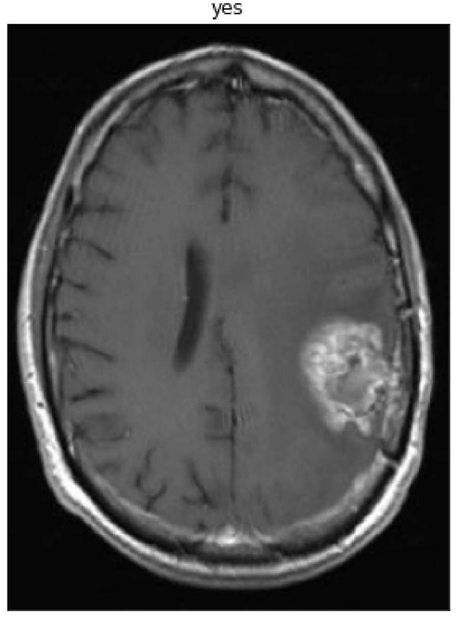


Image Data Generator

It is use for the Pre processing of data.

Flow from directory :

Takes the path to a directory & generates batches of augmented data. :

In [23]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(zoom_range=0.15,width_shift_range=0.2,height_shift_range=0.2,shear_range=0.15)
test_datagen = ImageDataGenerator()
val_datagen = ImageDataGenerator()
train_generator = train_datagen.flow_from_directory(train_path,target_size=(224, 224),batch_size=32,shuffle=True,class_mode='binary')
test_generator = test_datagen.flow_from_directory(test_path,target_size=(224,224),batch_size=32,shuffle=False,class_mode='binary')
val_generator = val_datagen.flow_from_directory(val_path,target_size=(224,224),batch_size=32,shuffle=False,class_mode='binary')
```

```
Found 202 images belonging to 2 classes.
Found 27 images belonging to 2 classes.
Found 24 images belonging to 2 classes.
```

Transfer Learning

A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image classification task, such as the ImageNet dataset. The weights of the pre-trained model can be utilized for the classification of other tasks. You don't have to train your model from scratch.

The intuition behind transfer learning for image classification is that when a model is trained on a sufficiently large and diverse dataset, it can function effectively as a generic model and be applied to various other classification tasks. The feature map of a pre-trained model can be utilized.

In [24]:
```
from tensorflow.keras.applications import ResNet50

model = ResNet50(
      input_shape = (224,224,3),
      include_top = False,
      weights = 'imagenet'
    )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 4s 0us/step
94781440/94765736 [==============================] - 4s 0us/step
```

Making layer Train able to False so it will not learn from Scratch.

In [25]:
```
for layers in model.layers:
    layers.trainable = False
```

You can see the number of trainable parameters reduced to a large extent.

In [26]:
```
from keras.layers import  Dropout
x = Flatten()(model.output)
x = Dropout(0.5)(x)
x = Dense(1, activation = "sigmoid")(x)

model = keras.Model(model.input, x)
model.compile(loss = "binary_crossentropy", optimizer = "adam", metrics = "accuracy")
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block1_2_relu[0][0]
conv2_block1_0_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_0_conv[0][0]
conv2_block1_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_3_conv[0][0]
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_bn[0][0] conv2_block1_3_bn[0][0]
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block1_out[0][0]
conv2_block2_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_1_conv[0][0]
conv2_block2_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_1_bn[0][0]
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block2_1_relu[0][0]
conv2_block2_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_2_conv[0][0]
conv2_block2_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_2_bn[0][0]
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block2_2_relu[0][0]
conv2_block2_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block2_3_conv[0][0]
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out[0][0]

			conv2_block2_3_bn[0][0]
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	conv2_block2_add[0][0]
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block2_out[0][0]
conv2_block3_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block3_1_conv[0][0]
conv2_block3_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block3_1_bn[0][0]
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block3_1_relu[0][0]
conv2_block3_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block3_2_conv[0][0]
conv2_block3_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block3_2_bn[0][0]
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block3_2_relu[0][0]
conv2_block3_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block3_3_conv[0][0]
conv2_block3_add (Add)	(None, 56, 56, 256)	0	conv2_block2_out[0][0] conv2_block3_3_bn[0][0]
conv2_block3_out (Activation)	(None, 56, 56, 256)	0	conv2_block3_add[0][0]
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32896	conv2_block3_out[0][0]
conv3_block1_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block1_1_conv[0][0]
conv3_block1_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block1_1_bn[0][0]
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block1_1_relu[0][0]
conv3_block1_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block1_2_conv[0][0]
conv3_block1_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block1_2_bn[0][0]
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131584	conv2_block3_out[0][0]
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block1_2_relu[0][0]
conv3_block1_0_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block1_0_conv[0][0]
conv3_block1_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block1_3_conv[0][0]
conv3_block1_add (Add)	(None, 28, 28, 512)	0	conv3_block1_0_bn[0][0] conv3_block1_3_bn[0][0]
conv3_block1_out (Activation)	(None, 28, 28, 512)	0	conv3_block1_add[0][0]
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block1_out[0][0]
conv3_block2_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block2_1_conv[0][0]
conv3_block2_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block2_1_bn[0][0]
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block2_1_relu[0][0]
conv3_block2_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block2_2_conv[0][0]

conv3_block2_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block2_2_bn[0][0]
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block2_2_relu[0][0]
conv3_block2_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block2_3_conv[0][0]
conv3_block2_add (Add)	(None, 28, 28, 512)	0	conv3_block1_out[0][0] conv3_block2_3_bn[0][0]
conv3_block2_out (Activation)	(None, 28, 28, 512)	0	conv3_block2_add[0][0]
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block2_out[0][0]
conv3_block3_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block3_1_conv[0][0]
conv3_block3_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block3_1_bn[0][0]
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block3_1_relu[0][0]
conv3_block3_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block3_2_conv[0][0]
conv3_block3_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block3_2_bn[0][0]
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block3_2_relu[0][0]
conv3_block3_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block3_3_conv[0][0]
conv3_block3_add (Add)	(None, 28, 28, 512)	0	conv3_block2_out[0][0] conv3_block3_3_bn[0][0]
conv3_block3_out (Activation)	(None, 28, 28, 512)	0	conv3_block3_add[0][0]
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65664	conv3_block3_out[0][0]
conv3_block4_1_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_1_conv[0][0]
conv3_block4_1_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][0]
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147584	conv3_block4_1_relu[0][0]
conv3_block4_2_bn (BatchNormali	(None, 28, 28, 128)	512	conv3_block4_2_conv[0][0]
conv3_block4_2_relu (Activation	(None, 28, 28, 128)	0	conv3_block4_2_bn[0][0]
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66048	conv3_block4_2_relu[0][0]
conv3_block4_3_bn (BatchNormali	(None, 28, 28, 512)	2048	conv3_block4_3_conv[0][0]
conv3_block4_add (Add)	(None, 28, 28, 512)	0	conv3_block3_out[0][0] conv3_block4_3_bn[0][0]
conv3_block4_out (Activation)	(None, 28, 28, 512)	0	conv3_block4_add[0][0]
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131328	conv3_block4_out[0][0]
conv4_block1_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block1_1_conv[0][0]
conv4_block1_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block1_1_bn[0][0]
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block1_1_relu[0][0]

conv4_block1_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block1_2_conv[0][0]
conv4_block1_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block1_2_bn[0][0]
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525312	conv3_block4_out[0][0]
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block1_2_relu[0][0]
conv4_block1_0_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block1_0_conv[0][0]
conv4_block1_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block1_3_conv[0][0]
conv4_block1_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_0_bn[0][0] conv4_block1_3_bn[0][0]
conv4_block1_out (Activation)	(None, 14, 14, 1024)	0	conv4_block1_add[0][0]
conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block1_out[0][0]
conv4_block2_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block2_1_conv[0][0]
conv4_block2_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block2_1_bn[0][0]
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block2_1_relu[0][0]
conv4_block2_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block2_2_conv[0][0]
conv4_block2_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block2_2_bn[0][0]
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block2_2_relu[0][0]
conv4_block2_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block2_3_conv[0][0]
conv4_block2_add (Add)	(None, 14, 14, 1024)	0	conv4_block1_out[0][0] conv4_block2_3_bn[0][0]
conv4_block2_out (Activation)	(None, 14, 14, 1024)	0	conv4_block2_add[0][0]
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block2_out[0][0]
conv4_block3_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block3_1_conv[0][0]
conv4_block3_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block3_1_bn[0][0]
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block3_1_relu[0][0]
conv4_block3_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block3_2_conv[0][0]
conv4_block3_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block3_2_bn[0][0]
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block3_2_relu[0][0]
conv4_block3_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block3_3_conv[0][0]
conv4_block3_add (Add)	(None, 14, 14, 1024)	0	conv4_block2_out[0][0] conv4_block3_3_bn[0][0]
conv4_block3_out (Activation)	(None, 14, 14, 1024)	0	conv4_block3_add[0][0]

conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block3_out[0][0]
conv4_block4_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block4_1_conv[0][0]
conv4_block4_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block4_1_bn[0][0]
conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block4_1_relu[0][0]
conv4_block4_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block4_2_conv[0][0]
conv4_block4_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block4_2_bn[0][0]
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block4_2_relu[0][0]
conv4_block4_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block4_3_conv[0][0]
conv4_block4_add (Add)	(None, 14, 14, 1024)	0	conv4_block3_out[0][0] conv4_block4_3_bn[0][0]
conv4_block4_out (Activation)	(None, 14, 14, 1024)	0	conv4_block4_add[0][0]
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block4_out[0][0]
conv4_block5_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block5_1_conv[0][0]
conv4_block5_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block5_1_bn[0][0]
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block5_1_relu[0][0]
conv4_block5_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block5_2_conv[0][0]
conv4_block5_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block5_2_bn[0][0]
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block5_2_relu[0][0]
conv4_block5_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block5_3_conv[0][0]
conv4_block5_add (Add)	(None, 14, 14, 1024)	0	conv4_block4_out[0][0] conv4_block5_3_bn[0][0]
conv4_block5_out (Activation)	(None, 14, 14, 1024)	0	conv4_block5_add[0][0]
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262400	conv4_block5_out[0][0]
conv4_block6_1_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block6_1_conv[0][0]
conv4_block6_1_relu (Activation	(None, 14, 14, 256)	0	conv4_block6_1_bn[0][0]
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590080	conv4_block6_1_relu[0][0]
conv4_block6_2_bn (BatchNormali	(None, 14, 14, 256)	1024	conv4_block6_2_conv[0][0]
conv4_block6_2_relu (Activation	(None, 14, 14, 256)	0	conv4_block6_2_bn[0][0]
conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263168	conv4_block6_2_relu[0][0]
conv4_block6_3_bn (BatchNormali	(None, 14, 14, 1024)	4096	conv4_block6_3_conv[0][0]
conv4_block6_add (Add)	(None, 14, 14, 1024)	0	conv4_block5_out[0][0] conv4_block6_3_bn[0][0]

conv4_block6_out (Activation)	(None, 14, 14, 1024)	0	conv4_block6_add[0][0]
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524800	conv4_block6_out[0][0]
conv5_block1_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block1_1_conv[0][0]
conv5_block1_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block1_1_bn[0][0]
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block1_1_relu[0][0]
conv5_block1_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block1_2_conv[0][0]
conv5_block1_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block1_2_bn[0][0]
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2099200	conv4_block6_out[0][0]
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block1_2_relu[0][0]
conv5_block1_0_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_0_conv[0][0]
conv5_block1_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block1_3_conv[0][0]
conv5_block1_add (Add)	(None, 7, 7, 2048)	0	conv5_block1_0_bn[0][0] conv5_block1_3_bn[0][0]
conv5_block1_out (Activation)	(None, 7, 7, 2048)	0	conv5_block1_add[0][0]
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block1_out[0][0]
conv5_block2_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_1_conv[0][0]
conv5_block2_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_1_bn[0][0]
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block2_1_relu[0][0]
conv5_block2_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block2_2_conv[0][0]
conv5_block2_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block2_2_bn[0][0]
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block2_2_relu[0][0]
conv5_block2_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block2_3_conv[0][0]
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 (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]

conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
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]
flatten (Flatten)	(None, 100352)	0	conv5_block3_out[0][0]
dropout (Dropout)	(None, 100352)	0	flatten[0][0]
dense (Dense)	(None, 1)	100353	dropout[0][0]

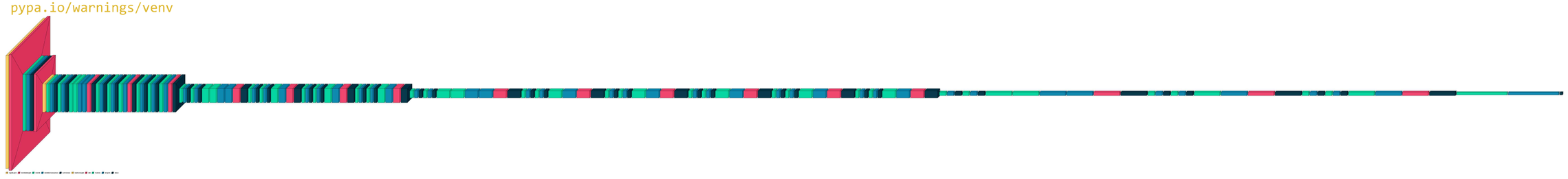
```
Total params: 23,688,065
Trainable params: 100,353
Non-trainable params: 23,587,712
```

Visual Respresntation of Model.

In [27]:
```
!pip install visualkeras
import visualkeras
visualkeras.layered_view(model, legend=True)
```

```
Collecting visualkeras
  Downloading visualkeras-0.0.2-py3-none-any.whl (12 kB)
Collecting aggdraw>=1.3.11
  Downloading aggdraw-1.3.16-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (989 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 989.8/989.8 kB 27.1 MB/s eta 0:00:0000:01
Requirement already satisfied: numpy>=1.18.1 in /opt/conda/lib/python3.7/site-packages (from visualkeras) (1.21.6)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from visualkeras) (9.1.0)
Installing collected packages: aggdraw, visualkeras
Successfully installed aggdraw-1.3.16 visualkeras-0.0.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

Out[27]:



In [ ]:
```
from tensorflow.keras.utils import plot_model
from IPython.display import Image
plot_model(model, to_file='convnet.png', show_shapes=True,show_layer_names=True)
Image(filename='convnet.png')
```

Tensor Flow Call Back

A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc)..

In [29]:
```
es=EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=20)
```

In [30]:
```
mc = ModelCheckpoint('./output/model.h5', monitor='val_accuracy', mode='max' )
```

Train The Model.

In [31]:
```
H = model.fit_generator(train_generator,validation_data=val_generator,epochs=150,verbose=1, callbacks=[mc,es])
```

```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:1972: UserWarning:

`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
```

```
Epoch 1/150
7/7 [==============================] - 13s 513ms/step - loss: 3.1115 - accuracy: 0.6139 - val_loss: 3.0477 - val_accuracy: 0.7917
```

```
/opt/conda/lib/python3.7/site-packages/keras/utils/generic_utils.py:497: CustomMaskWarning:

Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.
```

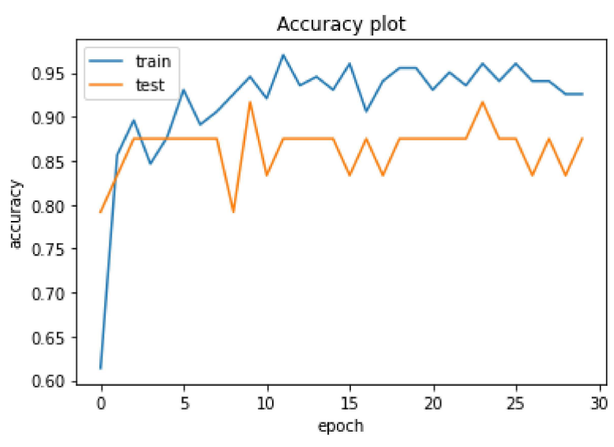
```
Epoch 2/150
7/7 [==============================] - 3s 377ms/step - loss: 1.0231 - accuracy: 0.8564 - val_loss: 1.8120 - val_accuracy: 0.8333
Epoch 3/150
7/7 [==============================] - 3s 372ms/step - loss: 0.8928 - accuracy: 0.8960 - val_loss: 1.9311 - val_accuracy: 0.8750
Epoch 4/150
7/7 [==============================] - 3s 405ms/step - loss: 1.1910 - accuracy: 0.8465 - val_loss: 1.7865 - val_accuracy: 0.8750
Epoch 5/150
7/7 [==============================] - 3s 387ms/step - loss: 0.7999 - accuracy: 0.8762 - val_loss: 1.4221 - val_accuracy: 0.8750
Epoch 6/150
7/7 [==============================] - 3s 435ms/step - loss: 0.5175 - accuracy: 0.9307 - val_loss: 1.3375 - val_accuracy: 0.8750
Epoch 7/150
7/7 [==============================] - 3s 388ms/step - loss: 0.5007 - accuracy: 0.8911 - val_loss: 1.6468 - val_accuracy: 0.8750
Epoch 8/150
7/7 [==============================] - 3s 385ms/step - loss: 0.4699 - accuracy: 0.9059 - val_loss: 1.3280 - val_accuracy: 0.8750
Epoch 9/150
7/7 [==============================] - 3s 380ms/step - loss: 0.4140 - accuracy: 0.9257 - val_loss: 1.2188 - val_accuracy: 0.7917
Epoch 10/150
7/7 [==============================] - 3s 384ms/step - loss: 0.2995 - accuracy: 0.9455 - val_loss: 1.5919 - val_accuracy: 0.9167
Epoch 11/150
7/7 [==============================] - 3s 387ms/step - loss: 0.4452 - accuracy: 0.9208 - val_loss: 1.2875 - val_accuracy: 0.8333
Epoch 12/150
7/7 [==============================] - 3s 432ms/step - loss: 0.1246 - accuracy: 0.9703 - val_loss: 1.4626 - val_accuracy: 0.8750
Epoch 13/150
7/7 [==============================] - 3s 401ms/step - loss: 0.3283 - accuracy: 0.9356 - val_loss: 1.9050 - val_accuracy: 0.8750
Epoch 14/150
7/7 [==============================] - 3s 375ms/step - loss: 0.2449 - accuracy: 0.9455 - val_loss: 1.5585 - val_accuracy: 0.8750
Epoch 15/150
7/7 [==============================] - 3s 370ms/step - loss: 0.3023 - accuracy: 0.9307 - val_loss: 1.6733 - val_accuracy: 0.8750
Epoch 16/150
7/7 [==============================] - 3s 388ms/step - loss: 0.2489 - accuracy: 0.9604 - val_loss: 1.2752 - val_accuracy: 0.8333
Epoch 17/150
7/7 [==============================] - 3s 392ms/step - loss: 0.4995 - accuracy: 0.9059 - val_loss: 1.6391 - val_accuracy: 0.8750
Epoch 18/150
7/7 [==============================] - 3s 371ms/step - loss: 0.3481 - accuracy: 0.9406 - val_loss: 1.0059 - val_accuracy: 0.8333
Epoch 19/150
7/7 [==============================] - 3s 386ms/step - loss: 0.1085 - accuracy: 0.9554 - val_loss: 1.7063 - val_accuracy: 0.8750
Epoch 20/150
7/7 [==============================] - 3s 366ms/step - loss: 0.2391 - accuracy: 0.9554 - val_loss: 0.9274 - val_accuracy: 0.8750
Epoch 21/150
7/7 [==============================] - 3s 374ms/step - loss: 0.3246 - accuracy: 0.9307 - val_loss: 1.9062 - val_accuracy: 0.8750
Epoch 22/150
7/7 [==============================] - 3s 390ms/step - loss: 0.1766 - accuracy: 0.9505 - val_loss: 2.0436 - val_accuracy: 0.8750
Epoch 23/150
7/7 [==============================] - 3s 392ms/step - loss: 0.2661 - accuracy: 0.9356 - val_loss: 2.0789 - val_accuracy: 0.8750
Epoch 24/150
7/7 [==============================] - 3s 390ms/step - loss: 0.2438 - accuracy: 0.9604 - val_loss: 2.0888 - val_accuracy: 0.9167
Epoch 25/150
7/7 [==============================] - 3s 376ms/step - loss: 0.3348 - accuracy: 0.9406 - val_loss: 2.0826 - val_accuracy: 0.8750
Epoch 26/150
7/7 [==============================] - 3s 375ms/step - loss: 0.2028 - accuracy: 0.9604 - val_loss: 2.0548 - val_accuracy: 0.8750
Epoch 27/150
7/7 [==============================] - 3s 392ms/step - loss: 0.4647 - accuracy: 0.9406 - val_loss: 1.5354 - val_accuracy: 0.8333
Epoch 28/150
7/7 [==============================] - 3s 373ms/step - loss: 0.2684 - accuracy: 0.9406 - val_loss: 1.6350 - val_accuracy: 0.8750
Epoch 29/150
7/7 [==============================] - 3s 391ms/step - loss: 0.3612 - accuracy: 0.9257 - val_loss: 1.4791 - val_accuracy: 0.8333
Epoch 30/150
7/7 [==============================] - 3s 385ms/step - loss: 0.4451 - accuracy: 0.9257 - val_loss: 1.3958 - val_accuracy: 0.8750
Epoch 00030: early stopping
```

In [32]:
```
hist = H.history
```

Plotting the History

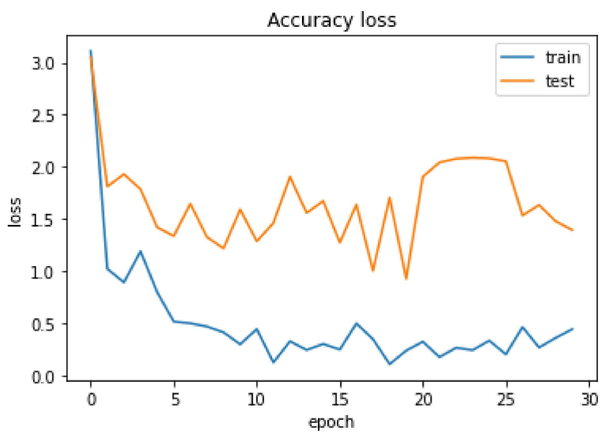
In [33]:
```
plt.plot(hist["accuracy"])
plt.plot(hist["val_accuracy"])
plt.title("Accuracy plot")
plt.legend(["train","test"])
plt.xlabel("epoch")
plt.ylabel("accuracy")
```

Out[33]:
```
Text(0, 0.5, 'accuracy')
```



In [34]:
```
plt.plot(hist["loss"])
plt.plot(hist["val_loss"])
plt.title("Accuracy loss")
plt.legend(["train","test"])
plt.xlabel("epoch")
plt.ylabel("loss")
```

Out[34]:
```
Text(0, 0.5, 'loss')
```



Model Evaluation

In [35]:
```
test_loss, test_acc = model.evaluate(test_generator, steps=len(test_generator), verbose=1)
print('Loss: %.3f' % (test_loss * 100.0))
print('Accuracy: %.3f' % (test_acc * 100.0))
```

```
1/1 [==============================] - 0s 354ms/step - loss: 4.0267 - accuracy: 0.8148
Loss: 402.668
Accuracy: 81.481
```

Classifiaction Report

In [36]:
```
from sklearn.metrics import classification_report
```

In [37]:
```
y_val = test_generator.classes
y_pred = model.predict(test_generator)
y_pred = np.argmax(y_pred,axis=1)
```

In [38]:
```
print(classification_report(y_val,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.41      1.00      0.58        11
           1       0.00      0.00      0.00        16

    accuracy                           0.41        27
   macro avg       0.20      0.50      0.29        27
weighted avg       0.17      0.41      0.24        27
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

In [39]:
```
class_indices = test_generator.class_indices
indices = {v:k for k,v in class_indices.items()}
```

In [40]:
```
filenames = test_generator.filenames
```

In [41]:
```
val_df = pd.DataFrame()
val_df['filename'] = filenames
val_df['actual'] = y_val
val_df['predicted'] = y_pred
val_df['actual'] = val_df['actual'].apply(lambda x: indices[x])
val_df['predicted'] = val_df['predicted'].apply(lambda x: indices[x])
val_df.loc[val_df['actual']==val_df['predicted'],'Same'] = True
val_df.loc[val_df['actual']!=val_df['predicted'],'Same'] = False
val_df.head(10)
```

Out[41]:

	filename	actual	predicted	Same
0	no/15 no.jpg	no	no	True
1	no/32 no.jpg	no	no	True
2	no/35 no.jpg	no	no	True
3	no/36 no.jpg	no	no	True
4	no/6 no.jpg	no	no	True
5	no/N26.JPG	no	no	True
6	no/No12.jpg	no	no	True
7	no/No17.jpg	no	no	True
8	no/No22.jpg	no	no	True
9	no/no 100.jpg	no	no	True

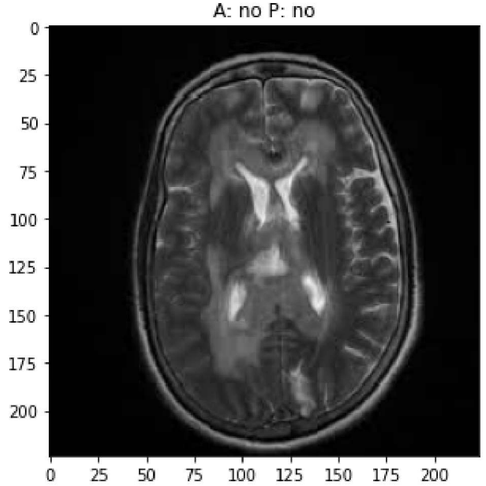
In [42]:
```
val_df = val_df.sample(frac=1).reset_index(drop=True)
```

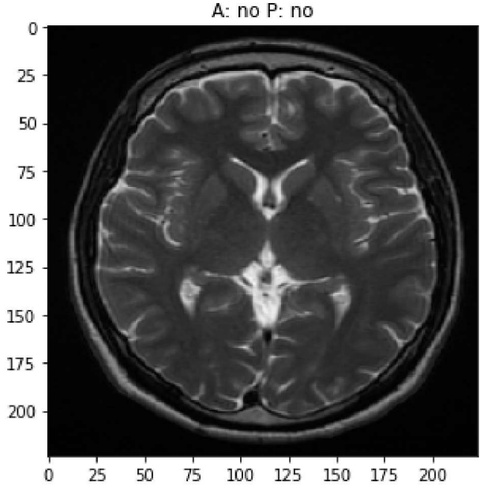
Prediction Comparison

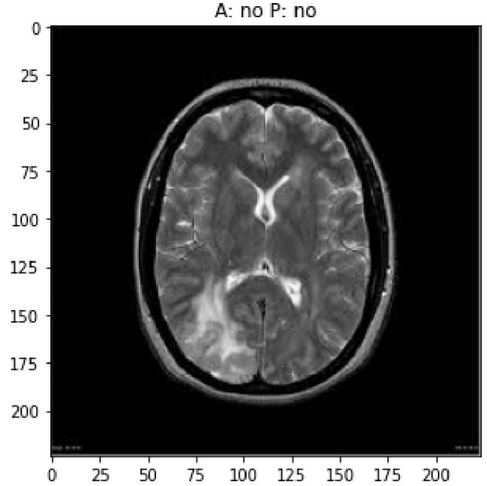
In [43]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
img_size = 224
def readImage(path):
    img = load_img(path,color_mode='rgb',target_size=(img_size,img_size))
    img = img_to_array(img)
    img = img/255.

    return img

def display_images(temp_df):
    temp_df = temp_df.reset_index(drop=True)
    plt.figure(figsize = (20 , 20))
    n = 0
    for i in range(6):
        n+=1
        plt.subplot(3 , 3, n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.3)
        image = readImage(f"../input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset/{temp_df.filename[i]}")
        plt.imshow(image)
        plt.title(f'A: {temp_df.actual[i]} P: {temp_df.predicted[i]}')
```

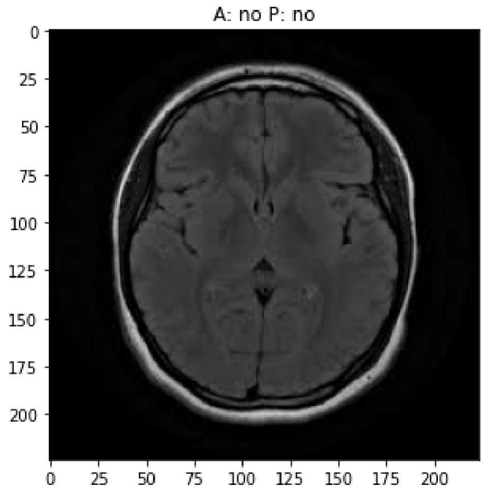
Correctly Classified

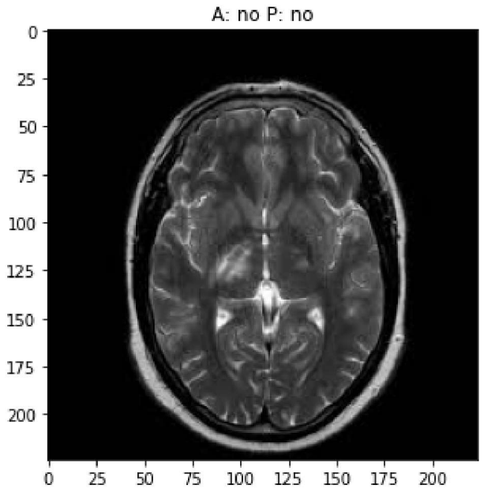
In [44]:
```
display_images(val_df[val_df['Same']==True])
```

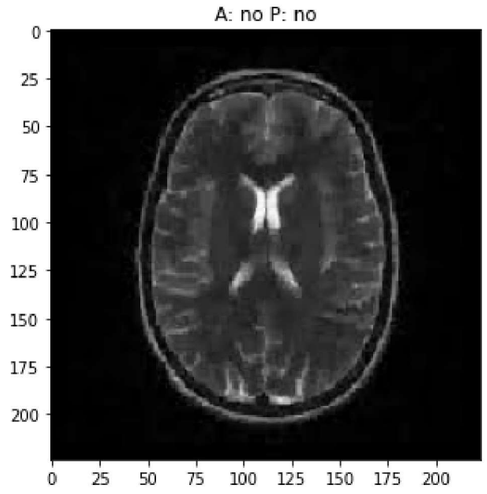






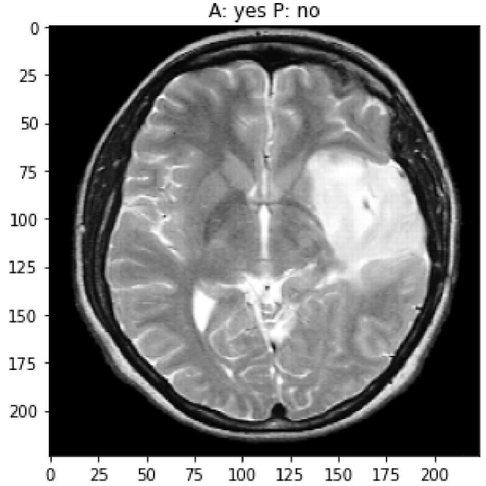


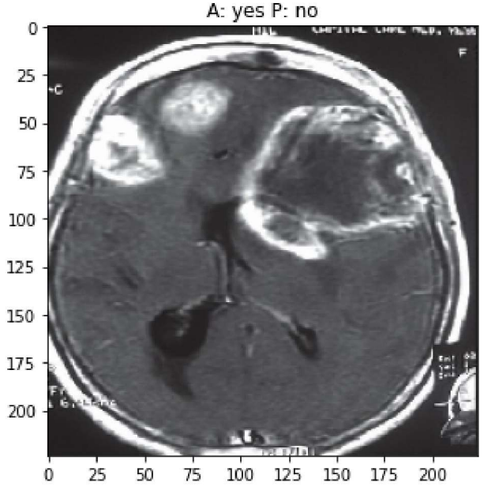


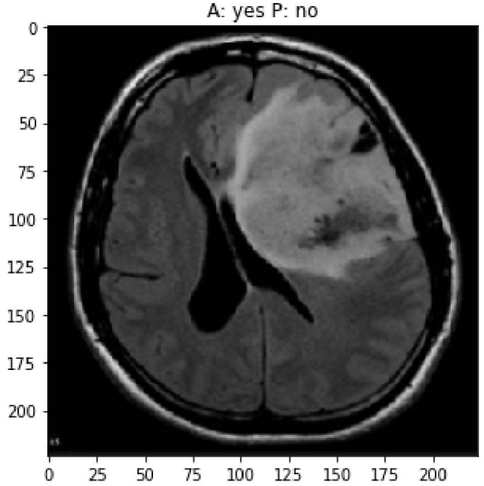


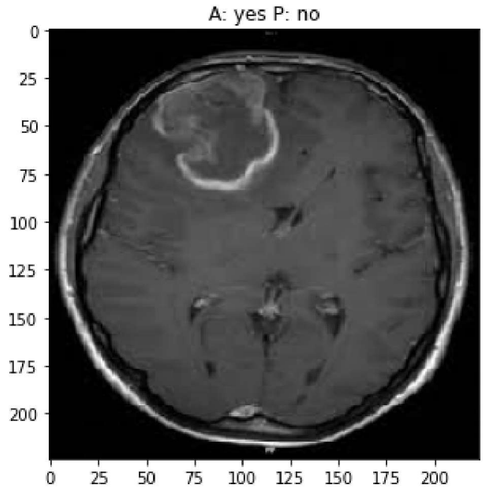
Miss Classified

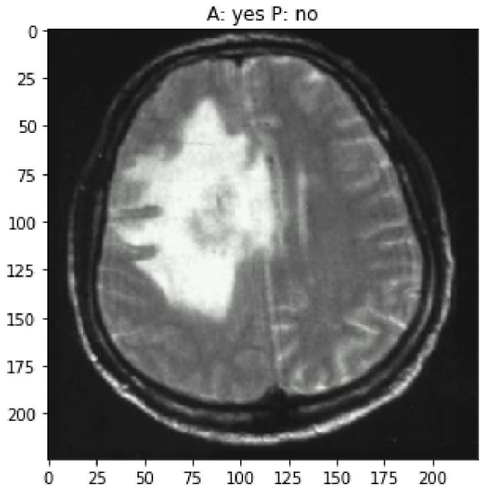
In [45]:
```
display_images(val_df[val_df['Same']!=True])
```

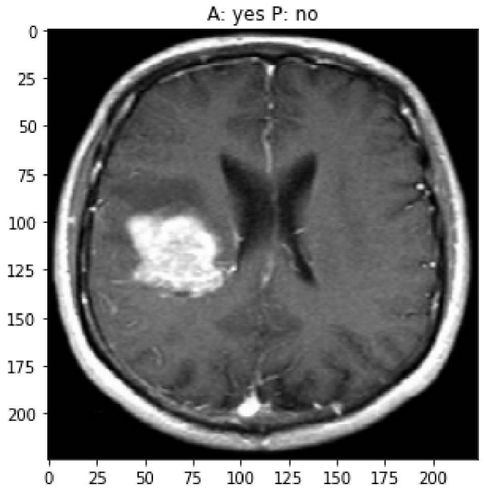












Confusion Matrix

In [46]:
```
cm = confusion_matrix(y_true=y_val, y_pred=y_pred)
```

In [47]:
```
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix',cmap=plt.cm.Blues):

    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)
```

```
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
        print(cm)

    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')
```

In [48]:
```
cm_plot_labels = ['1_Yes', '2_No']

plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='Confusion Matrix')
```

```
Confusion matrix, without normalization
[[11  0]
 [16  0]]
```