```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
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
```

```
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,
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,
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'
```

```
data_dir = "/content/drive/MyDrive/cse498r"
data_dir =pathlib.Path(data_dir)
```

```
Total_Images1 = glob.glob('/content/drive/MyDrive/cse498r/*/*.JPG')
#print("Total number of images: ", len(Total_Images1))

Total_Images2 = glob.glob('/content/drive/MyDrive/cse498r/*/*.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:  629
```

```
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()
```

	Filename	ClassId
0	x-ray (651).JPG	pneumonia
1	x-ray (650).JPG	normal
2	x-ray (649).JPG	normal
3	x-ray (644).JPG	normal
4	x-ray (643).JPG	normal

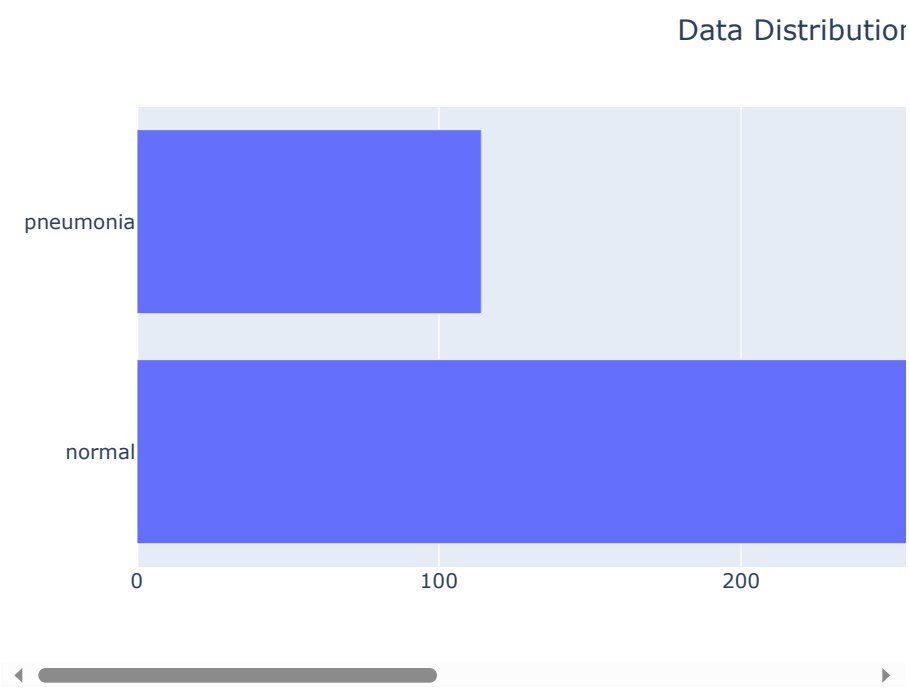
```
class_id_distributionTotal = total_df['ClassId'].value_counts()
class_id_distributionTotal.head(10)
```

```
normal       515
pneumonia    114
Name: ClassId, dtype: int64
```

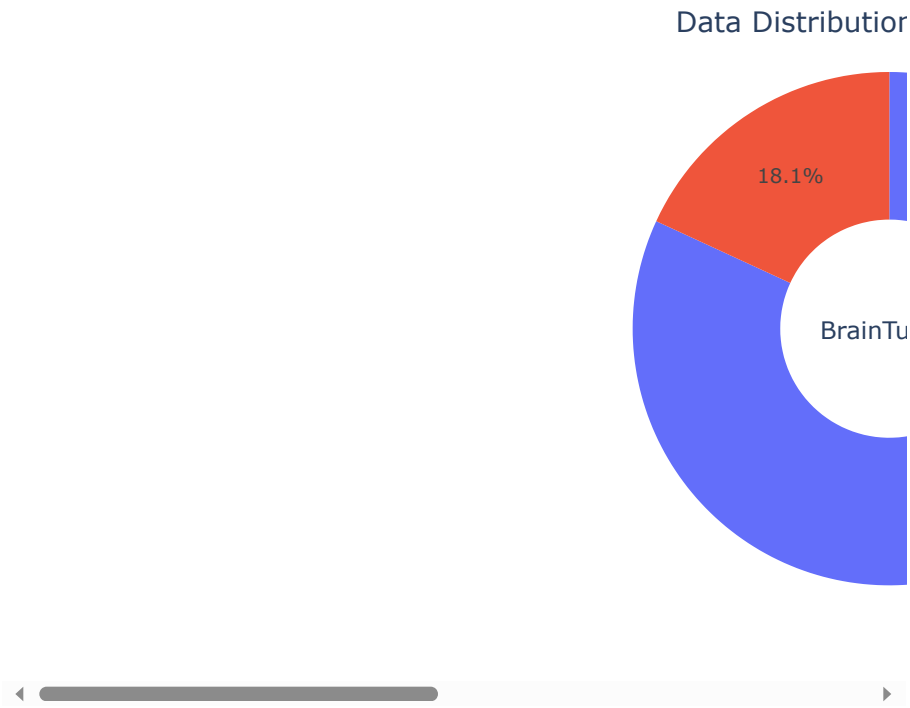
```
fig = go.Figure(go.Bar(
            x=  class_id_distributionTotal.values,
            y=class_id_distributionTotal.index,
            orientation='h'))

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


fig.show()
```




```
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=Fa
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```




```
splitfolders.ratio(data_dir, output="output", seed=101, ratio=(.8, .1, .1))
```

```
     Copying files: 629 files [00:12, 49.51 files/s]
```

```
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)
```

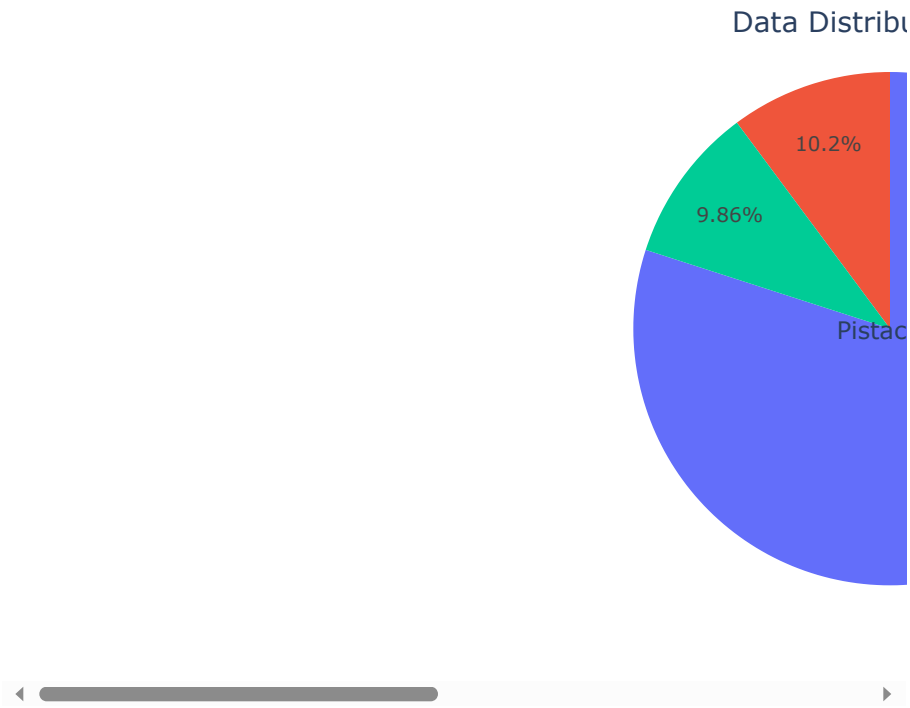
```
import glob
train_image1 = glob.glob('./output/train/*/*.jpg')
train_image2 = glob.glob('./output/train/*/*.JPG')
Total_TrainImages = train_image1 + train_image2
print("Total number of training images: ", len(Total_TrainImages))
```

```
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:  503
     Total number of test images:  64
     Total number of val images:  62
```

```
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,heigh
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```




```
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()
```

	Filename	ClassId
0	x-ray (280).jpg	normal
1	x-ray (277).jpg	normal
2	x-ray (631).jpg	normal
3	x-ray (237).jpg	normal
4	x-ray (303).jpg	normal

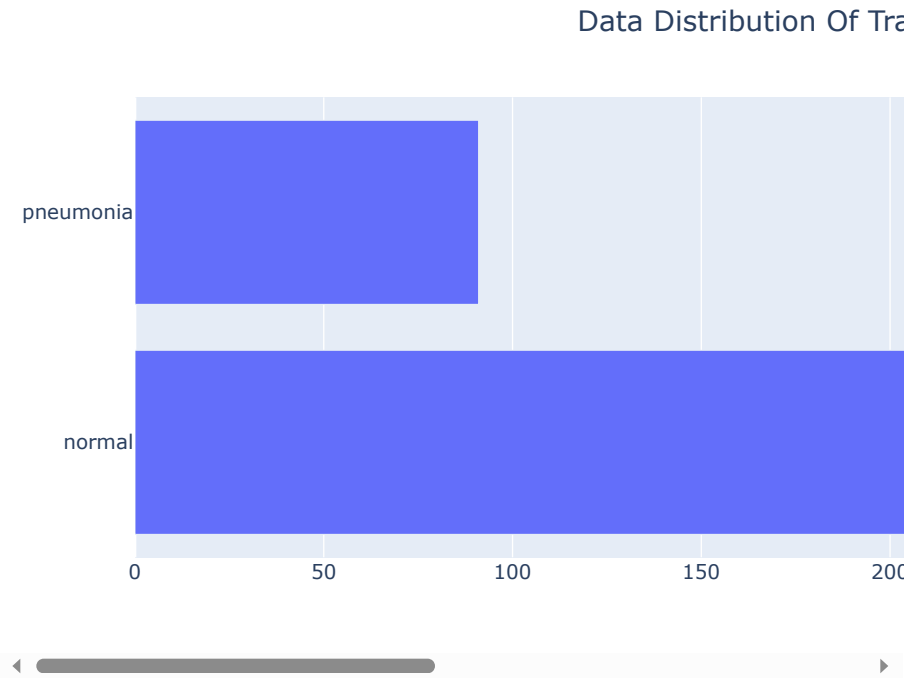
```
class_id_distribution_Train = train_df['ClassId'].value_counts()
class_id_distribution_Train.head(10)
```

```
normal       412
pneumonia     91
Name: ClassId, dtype: int64
```

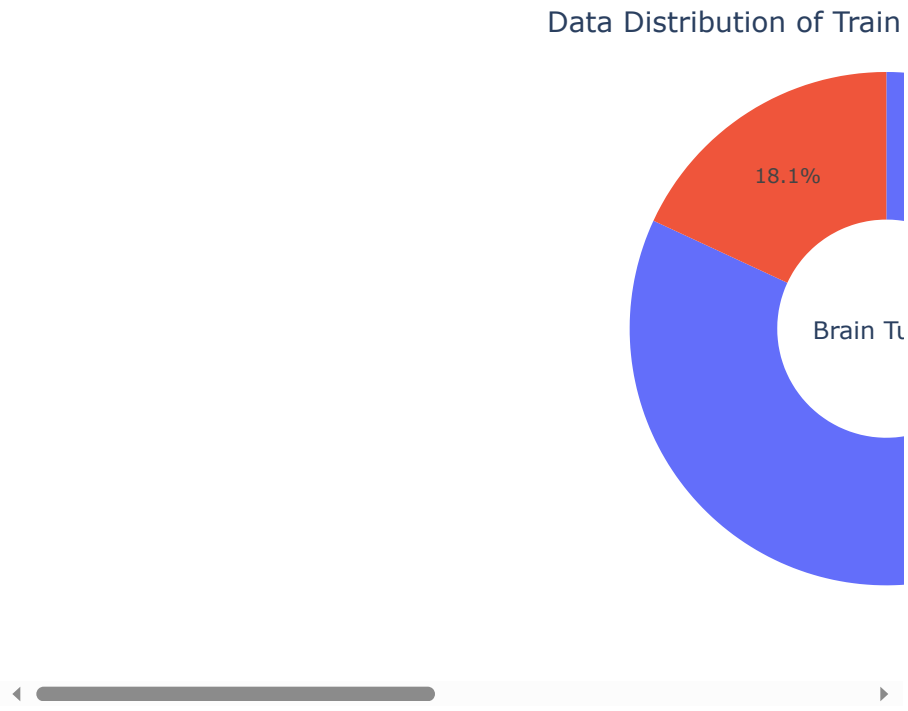
```
fig = go.Figure(go.Bar(
            x= class_id_distribution_Train.values,
            y=class_id_distribution_Train.index,
            orientation='h'))

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

fig.show()
```




```
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_s
fig.update_traces(textfont_size=15,textinfo='percent')
fig.show()
```




```
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()
```

	Filename	ClassId
0	x-ray (134).jpg	normal
1	x-ray (81).jpg	normal
2	x-ray (151).jpg	normal
3	x-ray (407).jpg	normal
4	x-ray (341).jpg	normal

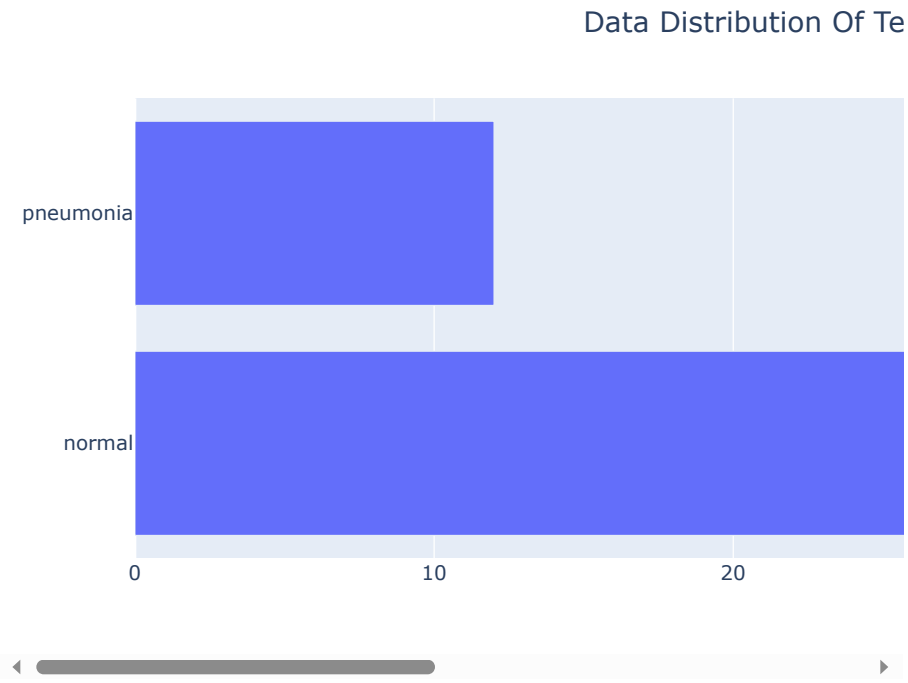
```
class_id_distribution_test = test_df['ClassId'].value_counts()
class_id_distribution_test.head(10)
```

```
normal       52
pneumonia    12
Name: ClassId, dtype: int64
```

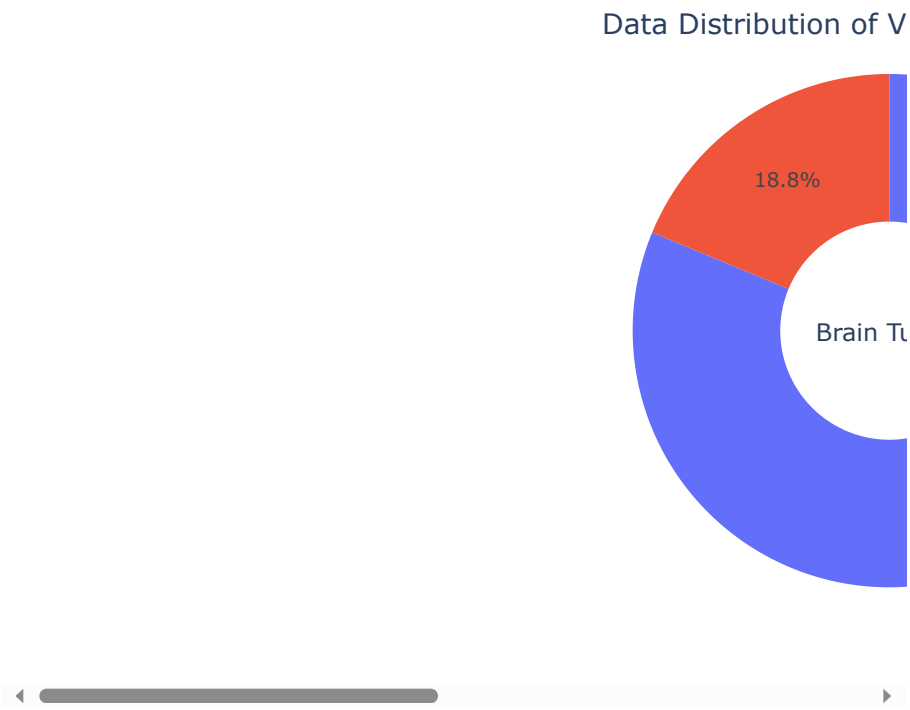
```
fig = go.Figure(go.Bar(
            x=class_id_distribution_test.values,
            y=class_id_distribution_test.index,
            orientation='h'))

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

fig.show()
```

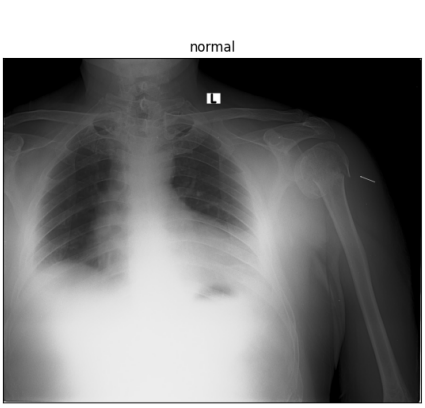


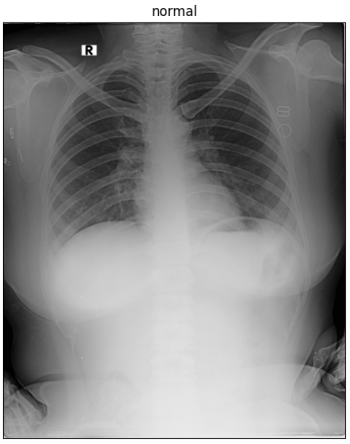

```
fig=px.pie(class_id_distribution_test.head(10),values= 'ClassId', names=test_df['ClassId'].unique(),hole=0.425)
fig.update_layout(title='Data Distribution of Validation Data',font_size=15,title_x=0.45,annotations=[dict(text='Brain Tumor',font_size=18,
fig.update_traces(textfont_size=15,textinfo='percent')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

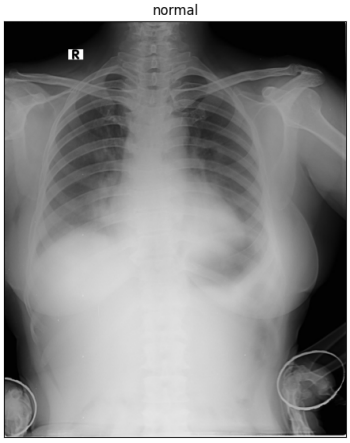


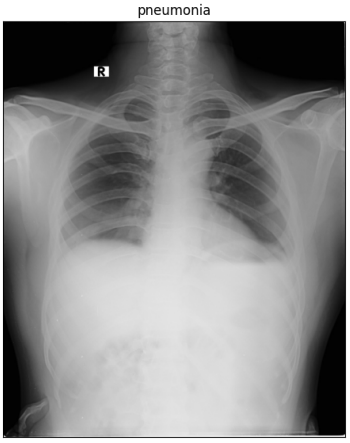

```
plot_df = train_df.sample(6).reset_index()
plt.figure(figsize=(15, 15))

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([])
```










```
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 503 images belonging to 2 classes.
Found 64 images belonging to 2 classes.
Found 62 images belonging to 2 classes.
```

```
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_no
94765736/94765736 [==============================] - 1s 0us/step
```

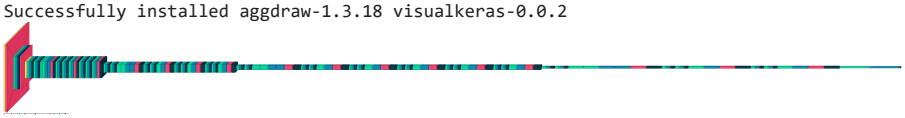
```
for layers in model.layers:
    layers.trainable = False


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()
```

```
!pip install visualkeras
import visualkeras
visualkeras.layered_view(model, legend=True)
```

```
Collecting visualkeras
  Downloading visualkeras-0.0.2-py3-none-any.whl (12 kB)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy>=1.18.1 in /usr/local/lib/python3.10/dist-packages
Collecting aggdraw>=1.3.11 (from visualkeras)
  Downloading aggdraw-1.3.18-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 993.7/993.7 kB 9.2 MB/s eta 0:00:00
Installing collected packages: aggdraw, visualkeras
Successfully installed aggdraw-1.3.18 visualkeras-0.0.2
```



```
es=EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=20)
```

```
mc = ModelCheckpoint('./output/model.h5', monitor='val_accuracy', mode='max' )
```

```
H = model.fit_generator(train_generator,validation_data=val_generator,epochs=10,verbose=1, callbacks=[mc,es])
```

```
<ipython-input-31-874fcdec34fe>:1: UserWarning:

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

Epoch 1/10
16/16 [==============================] - ETA: 0s - loss: 4.3514 - accuracy: 0.7376/usr/local/lib/python3.10/dist-packages/keras/src/engi

You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the nati

16/16 [==============================] - 32s 1s/step - loss: 4.3514 - accuracy: 0.7376 - val_loss: 2.8570 - val_accuracy: 0.8387
Epoch 2/10
16/16 [==============================] - 17s 1s/step - loss: 1.4214 - accuracy: 0.7913 - val_loss: 1.4757 - val_accuracy: 0.7097
Epoch 3/10
16/16 [==============================] - 17s 1s/step - loss: 1.0108 - accuracy: 0.8231 - val_loss: 1.5177 - val_accuracy: 0.7742
Epoch 4/10
16/16 [==============================] - 21s 1s/step - loss: 1.2912 - accuracy: 0.8072 - val_loss: 2.8147 - val_accuracy: 0.6290
Epoch 5/10
16/16 [==============================] - 18s 1s/step - loss: 1.6279 - accuracy: 0.8171 - val_loss: 2.2814 - val_accuracy: 0.8387
Epoch 6/10
16/16 [==============================] - 18s 1s/step - loss: 1.2705 - accuracy: 0.8509 - val_loss: 2.4906 - val_accuracy: 0.7742
Epoch 7/10
16/16 [==============================] - 17s 1s/step - loss: 1.4647 - accuracy: 0.8151 - val_loss: 2.2527 - val_accuracy: 0.7419
Epoch 8/10
16/16 [==============================] - 18s 1s/step - loss: 0.9608 - accuracy: 0.8708 - val_loss: 3.8246 - val_accuracy: 0.8226
Epoch 9/10
16/16 [==============================] - 18s 1s/step - loss: 1.7000 - accuracy: 0.8211 - val_loss: 2.5941 - val_accuracy: 0.7903
Epoch 10/10
16/16 [==============================] - 19s 1s/step - loss: 1.2342 - accuracy: 0.8529 - val_loss: 2.5782 - val_accuracy: 0.8065
```

```
hist = H.history
```

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

```
Text(0, 0.5, 'accuracy')
```