Pramathesh Shukla

CSC 578

Date: 11/22/2021

Project

Hybrid Model for detecting Lung Diseases

(1) Libraries

In [18]:
```
import numpy as np
import pandas as pd
from os import listdir

import matplotlib.pyplot as plt
from sklearn.feature_selection import chi2

import cv2
import numpy as np
import cv2
from keras.preprocessing import image
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
import pickle

from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import BatchNormalization, Conv2D, MaxPooling2D, MaxPool2D , Flatten, Dense, Activation, Dropout, SpatialDropout2D, ZeroPadding2D, Convolution2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import SGD

import tensorflow as tf

from tensorflow.keras.utils import to_categorical
```

2) Loading dataset & Visualizing

In [19]:
```
sdir=r'../CSC 578/COVID-19_Radiography_Database/COVID-19_Radiography_Dataset'
```

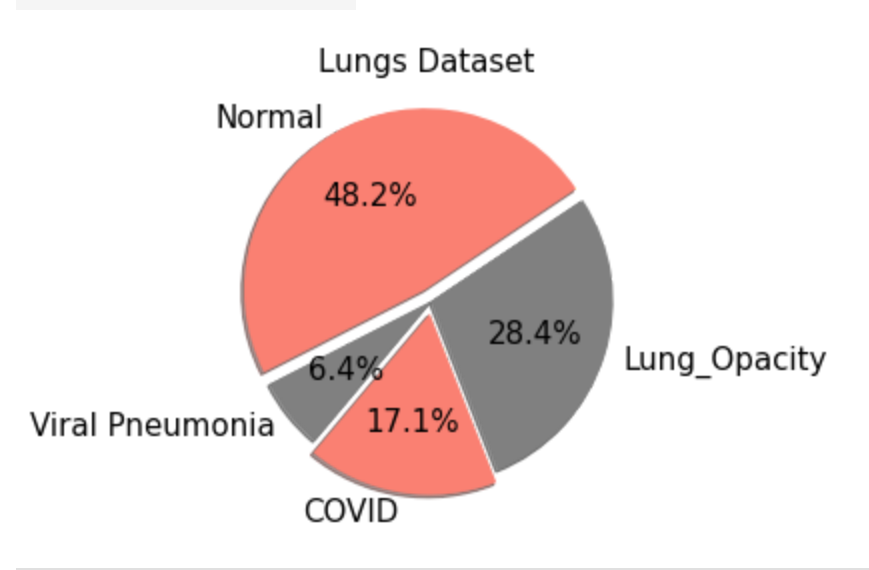
In [21]:
```
def getCount(directory_root):
    food_list = listdir(f"{directory_root}")
    dic_ = {}
    for label_folder in food_list:
      count = 1
      items = listdir(f"{directory_root}/{label_folder}")
      for label in items:
        count+=1
      dic_[label_folder] =  count
    return dic_
```

In [22]:
```
dic = getCount(sdir)
labels = list(dic.keys())
values = list(dic.values())
print(dic)
```

```
{'COVID': 3617, 'Lung_Opacity': 6013, 'Normal': 10193, 'Viral Pneumonia': 1346}
```

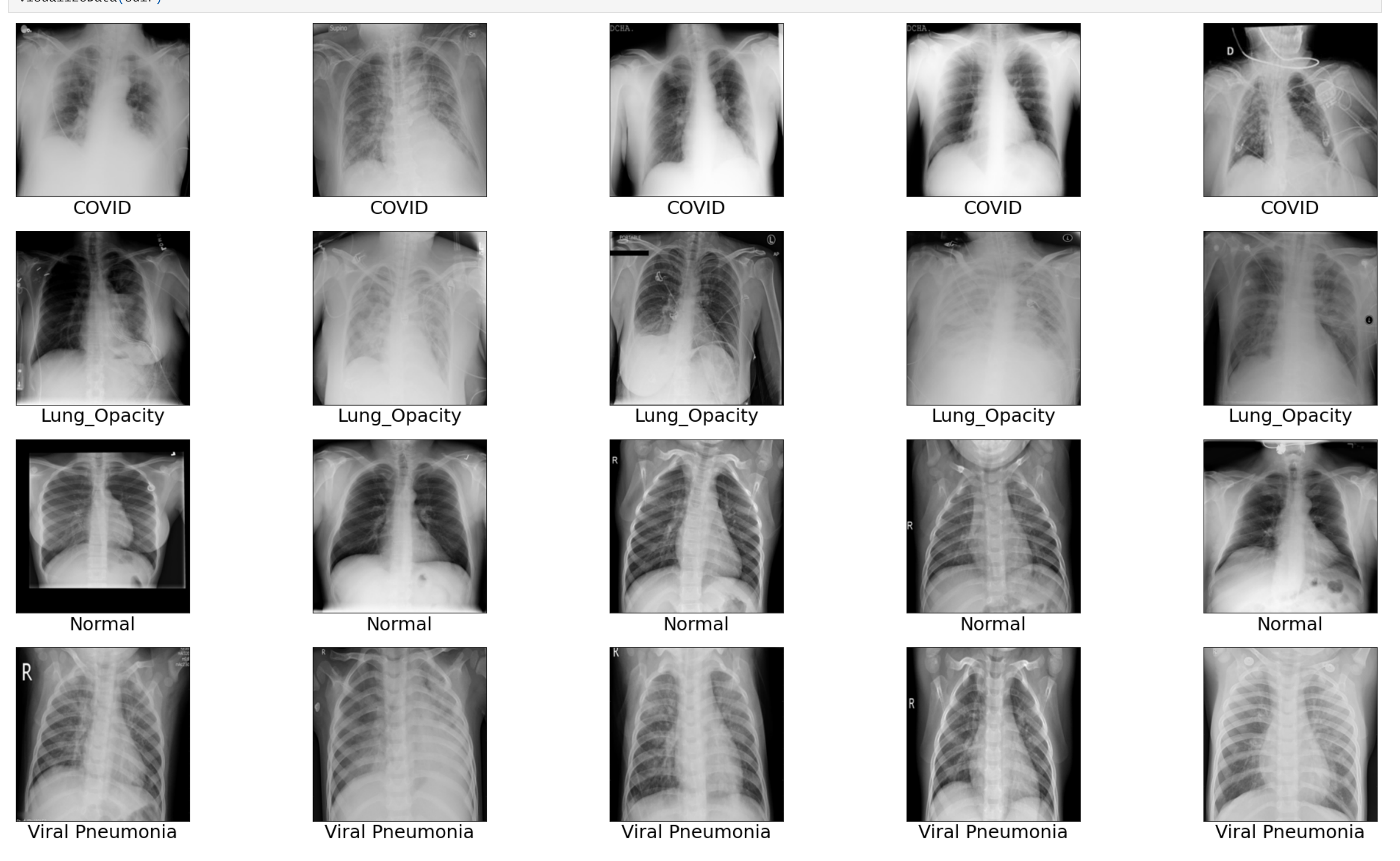
In [23]:
```
df = pd.DataFrame(columns=['Labels','Count'])

df['Labels']=values
df['Count']=labels
display(df)

textprops = {"fontsize":15} # Font size of text in pie chart
fig1, ax1 = plt.subplots()

colors = ['salmon','grey']

ax1.pie(values, labels=labels, colors= colors, shadow= True,explode=(0.1, 0,0.1,0), textprops =textprops, radius = 1.4,
        startangle=-130, autopct='%.1f%%')
ax1.axis('equal')
plt.title('Lungs Dataset', fontsize=15)
plt.show()
```

	Labels	Count
0	3617	COVID
1	6013	Lung_Opacity
2	10193	Normal
3	1346	Viral Pneumonia



In [24]:
```
def visualizeData(directory_root):
    food_list = listdir(f"{directory_root}")
    plt.figure(figsize = (35,20))
    dic_ = {}

    count = 1
    for label_folder in food_list:
        items = listdir(f"{directory_root}/{label_folder}")
        for label in items[:5]:
            plt.subplot(4,5,count)

            image_directory = f"{directory_root}/{label_folder}/{label}"
            img = cv2.imread(image_directory)
            Image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            plt.xticks([])
            plt.yticks([])
            plt.imshow(Image, cmap=plt.cm.bone)
            plt.xlabel(label_folder, fontsize=25)
            count+=1
    plt.show()
```

In [25]:
```
visualizeData(sdir)
```



3) Pre-Processing Image data & setting label

In [26]:
```
default_image_size = tuple((256, 256))
```

In [27]:
```
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None:
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print("Error :",e)
        return None
```

In [28]:
```
image_list, label_list = [], []
count = 0
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(sdir)
    for label_folder in root_dir:
        covid_folder_list = listdir(f"{sdir}/{label_folder}")
        print("Processing:",label_folder)
        for image in covid_folder_list[:1346]: # Execute Negative & Positive folder
            count+=1
            image_directory = f"{sdir}/{label_folder}/{image}"
            image_list.append(convert_image_to_array(image_directory))
            label_list.append(label_folder)

    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
```

```
[INFO] Loading images ...
Processing: COVID
Processing: Lung_Opacity
Processing: Normal
Processing: Viral Pneumonia
[INFO] Image loading completed
```

In [29]:
```
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
image_labels_2 = to_categorical(image_labels)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
```

In [32]:
```
print(label_binarizer.classes_)
```

```
['COVID' 'Lung_Opacity' 'Normal' 'Viral Pneumonia']
```

In [33]:
```
np_image_list = np.array(image_list, dtype=np.float16) / 255.0
```

In [34]:
```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels,#image_labels_2,#
                                                    test_size=0.2, random_state = 42, shuffle=True)
```

```
[INFO] Spliting data to train, test
```

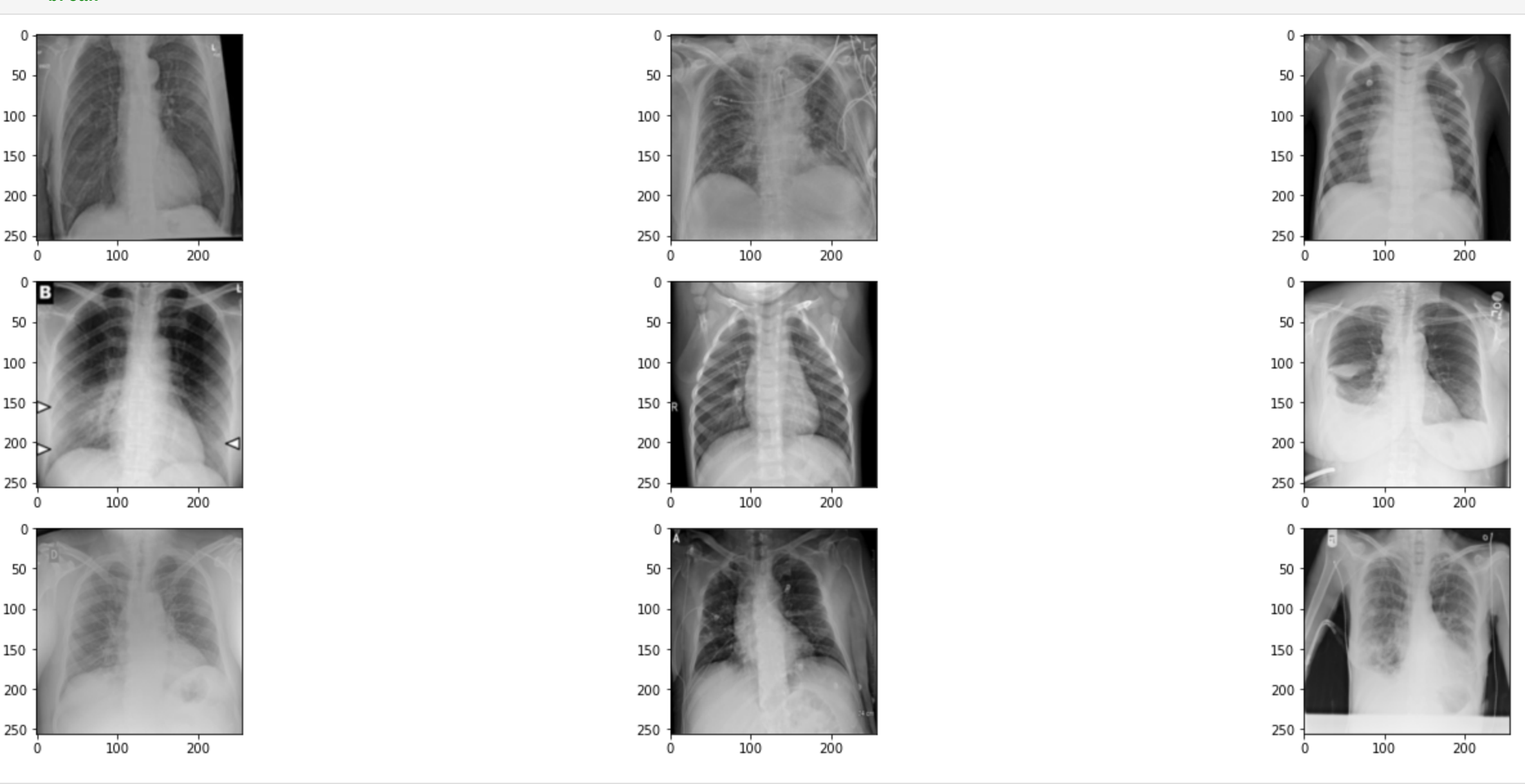
In [35]:
```
print(x_train.shape, y_train.shape, '\n',x_test.shape, y_test.shape)
```

```
(4306, 256, 256, 3) (4306, 4)
 (1077, 256, 256, 3) (1077, 4)
```

In [36]:
```
aug = ImageDataGenerator(
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    shear_range=0.0,
    zoom_range=0.0,
    horizontal_flip=False,
    fill_mode="nearest")
```

In [37]:
```
# fit parameters from data
aug.fit(x_train)
```

In [38]:
```
# Configure batch size and retrieve one batch of images
for x_batch, y_batch in aug.flow(x_train, y_train, batch_size=32):
    # Show 9 images
    plt.figure(figsize = (25,10))
    for i in range(0, 9):
        plt.subplot(330 + 1 + i)
        plt.imshow(x_batch[i])
    # show the plot
    plt.show()
    break
```



In [39]:
```
EPOCHS = 100#50
INIT_LR = 1e-2
BS = 32
width=256
height=256
depth=3
```

Custom Model (CNN)

In [51]:
```
model_custom = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model_custom.add(Conv2D(32, (3, 3), padding="same",activation="relu",input_shape=inputShape, strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(3, 3)))

model_custom.add(Conv2D(64, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(64, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Conv2D(128, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(128, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(Conv2D(512, (3, 3), padding="same",activation="relu", strides=(1,1)))
model_custom.add(MaxPooling2D(pool_size=(2, 2)))

model_custom.add(Flatten())
model_custom.add(Dense(1024, activation="relu"))
model_custom.add(Dropout(0.5))
model_custom.add(Dense(4,activation="softmax" ))
```

In [52]:
```
model_custom.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 85, 85, 64)        18496
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 85, 85, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 42, 42, 128)       73856
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 42, 42, 128)       147584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 21, 21, 128)       0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 21, 21, 512)       590336
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 21, 21, 512)       2359808
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 21, 21, 512)       2359808
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 10, 10, 512)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              52429824
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 4100
=================================================================
Total params: 58,021,636
Trainable params: 58,021,636
Non-trainable params: 0
_________________________________________________________________
```

In [53]:
```
opt = SGD(lr=INIT_LR)
model_custom.compile(loss='mean_squared_error',
              optimizer = opt,metrics=["accuracy"])
```

In [54]:
```
history = model_custom.fit_generator(aug.flow(x_train,y_train,batch_size=BS),
                                  validation_data= (x_test, y_test),
                                   steps_per_epoch=len(x_train) // BS,
                                   epochs=EPOCHS,
                                   verbose=1)
```

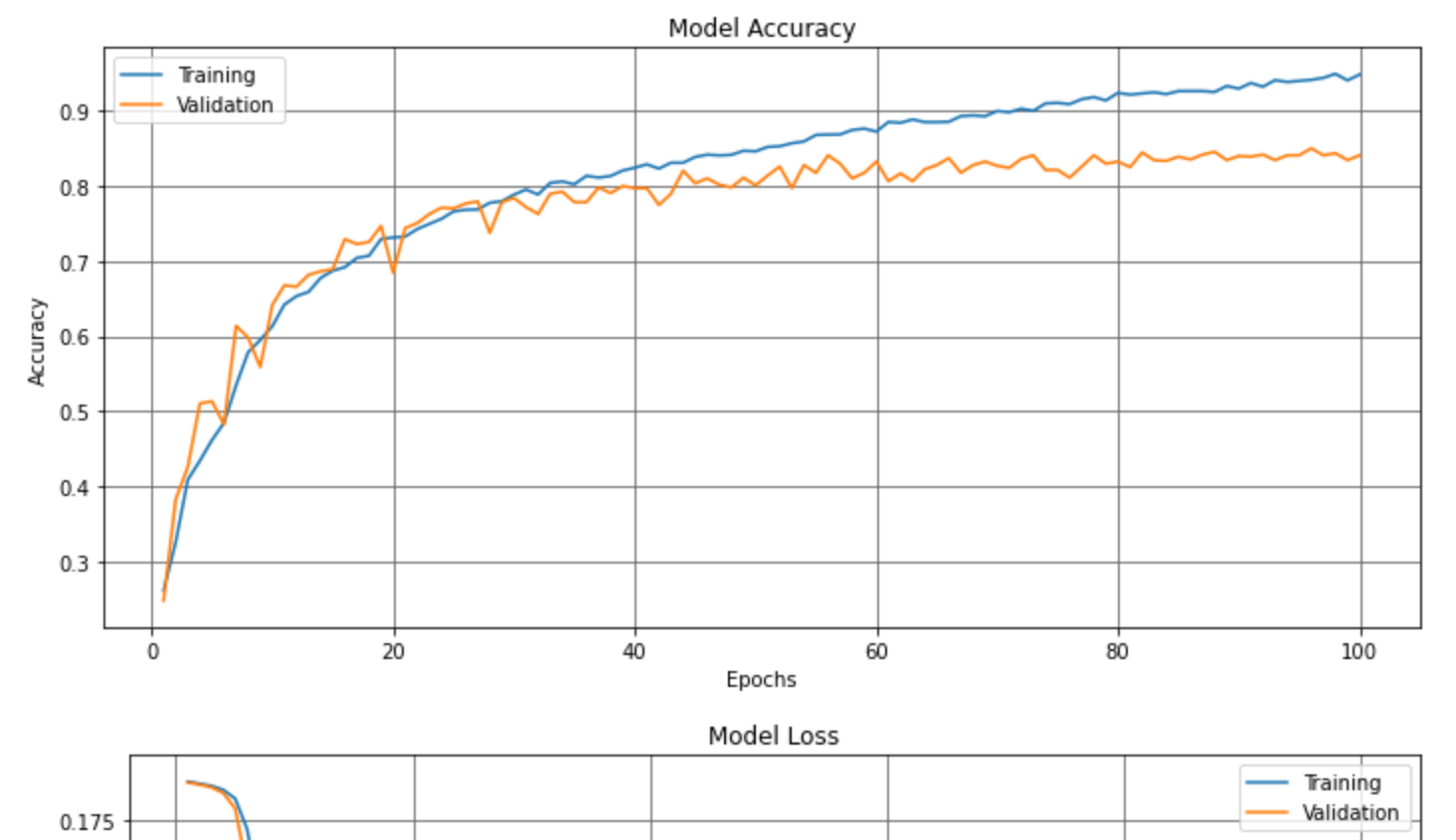
```
Epoch 1/100
134/134 [==============================] - 374s 3s/step - loss: 0.1872 - accuracy: 0.2620 - val_loss: 0.1869 - val_accuracy: 0.2488
Epoch 2/100
134/134 [==============================] - 353s 3s/step - loss: 0.1865 - accuracy: 0.3264 - val_loss: 0.1862 - val_accuracy: 0.3835
Epoch 3/100
134/134 [==============================] - 375s 3s/step - loss: 0.1858 - accuracy: 0.4092 - val_loss: 0.1854 - val_accuracy: 0.4262
Epoch 4/100
134/134 [==============================] - 379s 3s/step - loss: 0.1846 - accuracy: 0.4345 - val_loss: 0.1836 - val_accuracy: 0.5107
Epoch 5/100
134/134 [==============================] - 375s 3s/step - loss: 0.1818 - accuracy: 0.4619 - val_loss: 0.1786 - val_accuracy: 0.5135
Epoch 6/100
134/134 [==============================] - 372s 3s/step - loss: 0.1723 - accuracy: 0.4850 - val_loss: 0.1597 - val_accuracy: 0.4828
Epoch 7/100
134/134 [==============================] - 379s 3s/step - loss: 0.1518 - accuracy: 0.5349 - val_loss: 0.1366 - val_accuracy: 0.6137
Epoch 8/100
134/134 [==============================] - 372s 3s/step - loss: 0.1426 - accuracy: 0.5788 - val_loss: 0.1319 - val_accuracy: 0.5980
Epoch 9/100
134/134 [==============================] - 361s 3s/step - loss: 0.1346 - accuracy: 0.5950 - val_loss: 0.1409 - val_accuracy: 0.5590
Epoch 10/100
134/134 [==============================] - 379s 3s/step - loss: 0.1295 - accuracy: 0.6132 - val_loss: 0.1211 - val_accuracy: 0.6416
Epoch 11/100
134/134 [==============================] - 378s 3s/step - loss: 0.1220 - accuracy: 0.6420 - val_loss: 0.1175 - val_accuracy: 0.6676
Epoch 12/100
134/134 [==============================] - 392s 3s/step - loss: 0.1200 - accuracy: 0.6533 - val_loss: 0.1140 - val_accuracy: 0.6657
Epoch 13/100
134/134 [==============================] - 379s 3s/step - loss: 0.1164 - accuracy: 0.6589 - val_loss: 0.1131 - val_accuracy: 0.6815
Epoch 14/100
134/134 [==============================] - 382s 3s/step - loss: 0.1114 - accuracy: 0.6774 - val_loss: 0.1099 - val_accuracy: 0.6862
Epoch 15/100
134/134 [==============================] - 373s 3s/step - loss: 0.1080 - accuracy: 0.6869 - val_loss: 0.1058 - val_accuracy: 0.6890
Epoch 16/100
134/134 [==============================] - 377s 3s/step - loss: 0.1055 - accuracy: 0.6916 - val_loss: 0.0979 - val_accuracy: 0.7289
Epoch 17/100
134/134 [==============================] - 376s 3s/step - loss: 0.1021 - accuracy: 0.7040 - val_loss: 0.0952 - val_accuracy: 0.7224
Epoch 18/100
134/134 [==============================] - 361s 3s/step - loss: 0.0995 - accuracy: 0.7068 - val_loss: 0.0955 - val_accuracy: 0.7252
Epoch 19/100
134/134 [==============================] - 359s 3s/step - loss: 0.0945 - accuracy: 0.7291 - val_loss: 0.0900 - val_accuracy: 0.7465
Epoch 20/100
134/134 [==============================] - 356s 3s/step - loss: 0.0931 - accuracy: 0.7312 - val_loss: 0.1083 - val_accuracy: 0.6843
Epoch 21/100
134/134 [==============================] - 355s 3s/step - loss: 0.0916 - accuracy: 0.7323 - val_loss: 0.0856 - val_accuracy: 0.7437
Epoch 22/100
134/134 [==============================] - 367s 3s/step - loss: 0.0888 - accuracy: 0.7424 - val_loss: 0.0851 - val_accuracy: 0.7502
Epoch 23/100
134/134 [==============================] - 374s 3s/step - loss: 0.0859 - accuracy: 0.7492 - val_loss: 0.0822 - val_accuracy: 0.7623
Epoch 24/100
134/134 [==============================] - 376s 3s/step - loss: 0.0838 - accuracy: 0.7560 - val_loss: 0.0781 - val_accuracy: 0.7707
Epoch 25/100
134/134 [==============================] - 376s 3s/step - loss: 0.0825 - accuracy: 0.7658 - val_loss: 0.0799 - val_accuracy: 0.7697
Epoch 26/100
134/134 [==============================] - 363s 3s/step - loss: 0.0813 - accuracy: 0.7679 - val_loss: 0.0771 - val_accuracy: 0.7762
Epoch 27/100
134/134 [==============================] - 340s 3s/step - loss: 0.0802 - accuracy: 0.7684 - val_loss: 0.0750 - val_accuracy: 0.7790
Epoch 28/100
134/134 [==============================] - 339s 3s/step - loss: 0.0777 - accuracy: 0.7773 - val_loss: 0.0895 - val_accuracy: 0.7372
Epoch 29/100
134/134 [==============================] - 339s 3s/step - loss: 0.0768 - accuracy: 0.7794 - val_loss: 0.0742 - val_accuracy: 0.7781
Epoch 30/100
134/134 [==============================] - 340s 3s/step - loss: 0.0752 - accuracy: 0.7880 - val_loss: 0.0748 - val_accuracy: 0.7837
Epoch 31/100
134/134 [==============================] - 339s 3s/step - loss: 0.0739 - accuracy: 0.7948 - val_loss: 0.0783 - val_accuracy: 0.7716
Epoch 32/100
134/134 [==============================] - 341s 3s/step - loss: 0.0733 - accuracy: 0.7883 - val_loss: 0.0837 - val_accuracy: 0.7623
Epoch 33/100
134/134 [==============================] - 341s 3s/step - loss: 0.0701 - accuracy: 0.8037 - val_loss: 0.0742 - val_accuracy: 0.7892
Epoch 34/100
134/134 [==============================] - 340s 3s/step - loss: 0.0706 - accuracy: 0.8056 - val_loss: 0.0732 - val_accuracy: 0.7920
Epoch 35/100
134/134 [==============================] - 340s 3s/step - loss: 0.0696 - accuracy: 0.8018 - val_loss: 0.0735 - val_accuracy: 0.7781
Epoch 36/100
134/134 [==============================] - 340s 3s/step - loss: 0.0675 - accuracy: 0.8131 - val_loss: 0.0796 - val_accuracy: 0.7781
Epoch 37/100
134/134 [==============================] - 341s 3s/step - loss: 0.0671 - accuracy: 0.8107 - val_loss: 0.0696 - val_accuracy: 0.7976
Epoch 38/100
134/134 [==============================] - 340s 3s/step - loss: 0.0663 - accuracy: 0.8128 - val_loss: 0.0718 - val_accuracy: 0.7902
Epoch 39/100
134/134 [==============================] - 339s 3s/step - loss: 0.0651 - accuracy: 0.8201 - val_loss: 0.0705 - val_accuracy: 0.7994
Epoch 40/100
134/134 [==============================] - 342s 3s/step - loss: 0.0640 - accuracy: 0.8238 - val_loss: 0.0708 - val_accuracy: 0.7967
Epoch 41/100
134/134 [==============================] - 340s 3s/step - loss: 0.0623 - accuracy: 0.8283 - val_loss: 0.0727 - val_accuracy: 0.7967
Epoch 42/100
134/134 [==============================] - 339s 3s/step - loss: 0.0622 - accuracy: 0.8226 - val_loss: 0.0793 - val_accuracy: 0.7744
Epoch 43/100
134/134 [==============================] - 341s 3s/step - loss: 0.0607 - accuracy: 0.8304 - val_loss: 0.0758 - val_accuracy: 0.7892
Epoch 44/100
134/134 [==============================] - 340s 3s/step - loss: 0.0607 - accuracy: 0.8304 - val_loss: 0.0647 - val_accuracy: 0.8199
Epoch 45/100
134/134 [==============================] - 340s 3s/step - loss: 0.0587 - accuracy: 0.8381 - val_loss: 0.0685 - val_accuracy: 0.8032
Epoch 46/100
134/134 [==============================] - 341s 3s/step - loss: 0.0573 - accuracy: 0.8411 - val_loss: 0.0693 - val_accuracy: 0.8097
Epoch 47/100
134/134 [==============================] - 339s 3s/step - loss: 0.0578 - accuracy: 0.8402 - val_loss: 0.0665 - val_accuracy: 0.8013
Epoch 48/100
134/134 [==============================] - 342s 3s/step - loss: 0.0568 - accuracy: 0.8409 - val_loss: 0.0705 - val_accuracy: 0.7976
Epoch 49/100
134/134 [==============================] - 339s 3s/step - loss: 0.0557 - accuracy: 0.8463 - val_loss: 0.0652 - val_accuracy: 0.8106
Epoch 50/100
134/134 [==============================] - 339s 3s/step - loss: 0.0548 - accuracy: 0.8456 - val_loss: 0.0700 - val_accuracy: 0.8004
Epoch 51/100
134/134 [==============================] - 340s 3s/step - loss: 0.0540 - accuracy: 0.8514 - val_loss: 0.0672 - val_accuracy: 0.8134
Epoch 52/100
134/134 [==============================] - 339s 3s/step - loss: 0.0532 - accuracy: 0.8524 - val_loss: 0.0626 - val_accuracy: 0.8254
Epoch 53/100
134/134 [==============================] - 340s 3s/step - loss: 0.0516 - accuracy: 0.8563 - val_loss: 0.0716 - val_accuracy: 0.7967
Epoch 54/100
134/134 [==============================] - 341s 3s/step - loss: 0.0515 - accuracy: 0.8587 - val_loss: 0.0616 - val_accuracy: 0.8273
Epoch 55/100
134/134 [==============================] - 339s 3s/step - loss: 0.0498 - accuracy: 0.8673 - val_loss: 0.0666 - val_accuracy: 0.8171
Epoch 56/100
134/134 [==============================] - 340s 3s/step - loss: 0.0485 - accuracy: 0.8678 - val_loss: 0.0595 - val_accuracy: 0.8403
Epoch 57/100
134/134 [==============================] - 340s 3s/step - loss: 0.0483 - accuracy: 0.8680 - val_loss: 0.0645 - val_accuracy: 0.8292
Epoch 58/100
134/134 [==============================] - 339s 3s/step - loss: 0.0478 - accuracy: 0.8739 - val_loss: 0.0658 - val_accuracy: 0.8097
Epoch 59/100
134/134 [==============================] - 342s 3s/step - loss: 0.0472 - accuracy: 0.8758 - val_loss: 0.0681 - val_accuracy: 0.8171
Epoch 60/100
134/134 [==============================] - 340s 3s/step - loss: 0.0465 - accuracy: 0.8718 - val_loss: 0.0624 - val_accuracy: 0.8319
Epoch 61/100
134/134 [==============================] - 340s 3s/step - loss: 0.0442 - accuracy: 0.8847 - val_loss: 0.0681 - val_accuracy: 0.8059
Epoch 62/100
134/134 [==============================] - 340s 3s/step - loss: 0.0441 - accuracy: 0.8835 - val_loss: 0.0680 - val_accuracy: 0.8162
Epoch 63/100
134/134 [==============================] - 339s 3s/step - loss: 0.0432 - accuracy: 0.8879 - val_loss: 0.0714 - val_accuracy: 0.8059
Epoch 64/100
134/134 [==============================] - 340s 3s/step - loss: 0.0427 - accuracy: 0.8844 - val_loss: 0.0660 - val_accuracy: 0.8217
Epoch 65/100
134/134 [==============================] - 340s 3s/step - loss: 0.0429 - accuracy: 0.8844 - val_loss: 0.0623 - val_accuracy: 0.8273
Epoch 66/100
134/134 [==============================] - 339s 3s/step - loss: 0.0424 - accuracy: 0.8849 - val_loss: 0.0600 - val_accuracy: 0.8366
Epoch 67/100
134/134 [==============================] - 340s 3s/step - loss: 0.0396 - accuracy: 0.8924 - val_loss: 0.0686 - val_accuracy: 0.8171
Epoch 68/100
134/134 [==============================] - 339s 3s/step - loss: 0.0400 - accuracy: 0.8933 - val_loss: 0.0623 - val_accuracy: 0.8273
Epoch 69/100
134/134 [==============================] - 341s 3s/step - loss: 0.0402 - accuracy: 0.8921 - val_loss: 0.0614 - val_accuracy: 0.8319
Epoch 70/100
134/134 [==============================] - 342s 3s/step - loss: 0.0388 - accuracy: 0.8992 - val_loss: 0.0629 - val_accuracy: 0.8264
Epoch 71/100
134/134 [==============================] - 340s 3s/step - loss: 0.0385 - accuracy: 0.8975 - val_loss: 0.0670 - val_accuracy: 0.8236
Epoch 72/100
134/134 [==============================] - 359s 3s/step - loss: 0.0370 - accuracy: 0.9022 - val_loss: 0.0602 - val_accuracy: 0.8357
Epoch 73/100
134/134 [==============================] - 365s 3s/step - loss: 0.0378 - accuracy: 0.8994 - val_loss: 0.0597 - val_accuracy: 0.8403
Epoch 74/100
134/134 [==============================] - 374s 3s/step - loss: 0.0352 - accuracy: 0.9092 - val_loss: 0.0668 - val_accuracy: 0.8208
Epoch 75/100
134/134 [==============================] - 370s 3s/step - loss: 0.0344 - accuracy: 0.9099 - val_loss: 0.0630 - val_accuracy: 0.8208
Epoch 76/100
134/134 [==============================] - 385s 3s/step - loss: 0.0350 - accuracy: 0.9083 - val_loss: 0.0638 - val_accuracy: 0.8106
Epoch 77/100
134/134 [==============================] - 363s 3s/step - loss: 0.0326 - accuracy: 0.9148 - val_loss: 0.0663 - val_accuracy: 0.8254
Epoch 78/100
134/134 [==============================] - 367s 3s/step - loss: 0.0324 - accuracy: 0.9176 - val_loss: 0.0615 - val_accuracy: 0.8403
Epoch 79/100
134/134 [==============================] - 370s 3s/step - loss: 0.0333 - accuracy: 0.9132 - val_loss: 0.0613 - val_accuracy: 0.8292
Epoch 80/100
134/134 [==============================] - 367s 3s/step - loss: 0.0301 - accuracy: 0.9233 - val_loss: 0.0606 - val_accuracy: 0.8319
Epoch 81/100
134/134 [==============================] - 364s 3s/step - loss: 0.0309 - accuracy: 0.9209 - val_loss: 0.0686 - val_accuracy: 0.8245
Epoch 82/100
134/134 [==============================] - 366s 3s/step - loss: 0.0308 - accuracy: 0.9226 - val_loss: 0.0595 - val_accuracy: 0.8440
Epoch 83/100
134/134 [==============================] - 368s 3s/step - loss: 0.0291 - accuracy: 0.9240 - val_loss: 0.0604 - val_accuracy: 0.8338
Epoch 84/100
134/134 [==============================] - 406s 3s/step - loss: 0.0312 - accuracy: 0.9216 - val_loss: 0.0612 - val_accuracy: 0.8329
Epoch 85/100
134/134 [==============================] - 432s 3s/step - loss: 0.0298 - accuracy: 0.9256 - val_loss: 0.0609 - val_accuracy: 0.8384
Epoch 86/100
134/134 [==============================] - 407s 3s/step - loss: 0.0281 - accuracy: 0.9256 - val_loss: 0.0621 - val_accuracy: 0.8347
Epoch 87/100
134/134 [==============================] - 404s 3s/step - loss: 0.0285 - accuracy: 0.9256 - val_loss: 0.0599 - val_accuracy: 0.8412
Epoch 88/100
134/134 [==============================] - 400s 3s/step - loss: 0.0279 - accuracy: 0.9244 - val_loss: 0.0585 - val_accuracy: 0.8449
Epoch 89/100
134/134 [==============================] - 368s 3s/step - loss: 0.0264 - accuracy: 0.9324 - val_loss: 0.0624 - val_accuracy: 0.8338
Epoch 90/100
134/134 [==============================] - 352s 3s/step - loss: 0.0276 - accuracy: 0.9289 - val_loss: 0.0592 - val_accuracy: 0.8394
Epoch 91/100
134/134 [==============================] - 350s 3s/step - loss: 0.0240 - accuracy: 0.9364 - val_loss: 0.0613 - val_accuracy: 0.8384
Epoch 92/100
134/134 [==============================] - 347s 3s/step - loss: 0.0262 - accuracy: 0.9314 - val_loss: 0.0579 - val_accuracy: 0.8412
Epoch 93/100
134/134 [==============================] - 352s 3s/step - loss: 0.0237 - accuracy: 0.9401 - val_loss: 0.0636 - val_accuracy: 0.8338
Epoch 94/100
134/134 [==============================] - 349s 3s/step - loss: 0.0241 - accuracy: 0.9378 - val_loss: 0.0596 - val_accuracy: 0.8403
Epoch 95/100
134/134 [==============================] - 347s 3s/step - loss: 0.0227 - accuracy: 0.9392 - val_loss: 0.0602 - val_accuracy: 0.8403
Epoch 96/100
134/134 [==============================] - 356s 3s/step - loss: 0.0232 - accuracy: 0.9406 - val_loss: 0.0586 - val_accuracy: 0.8496
Epoch 97/100
134/134 [==============================] - 352s 3s/step - loss: 0.0221 - accuracy: 0.9431 - val_loss: 0.0627 - val_accuracy: 0.8403
Epoch 98/100
134/134 [==============================] - 357s 3s/step - loss: 0.0210 - accuracy: 0.9485 - val_loss: 0.0603 - val_accuracy: 0.8431
Epoch 99/100
134/134 [==============================] - 352s 3s/step - loss: 0.0226 - accuracy: 0.9399 - val_loss: 0.0621 - val_accuracy: 0.8338
Epoch 100/100
134/134 [==============================] - 378s 3s/step - loss: 0.0210 - accuracy: 0.9476 - val_loss: 0.0600 - val_accuracy: 0.8403
```

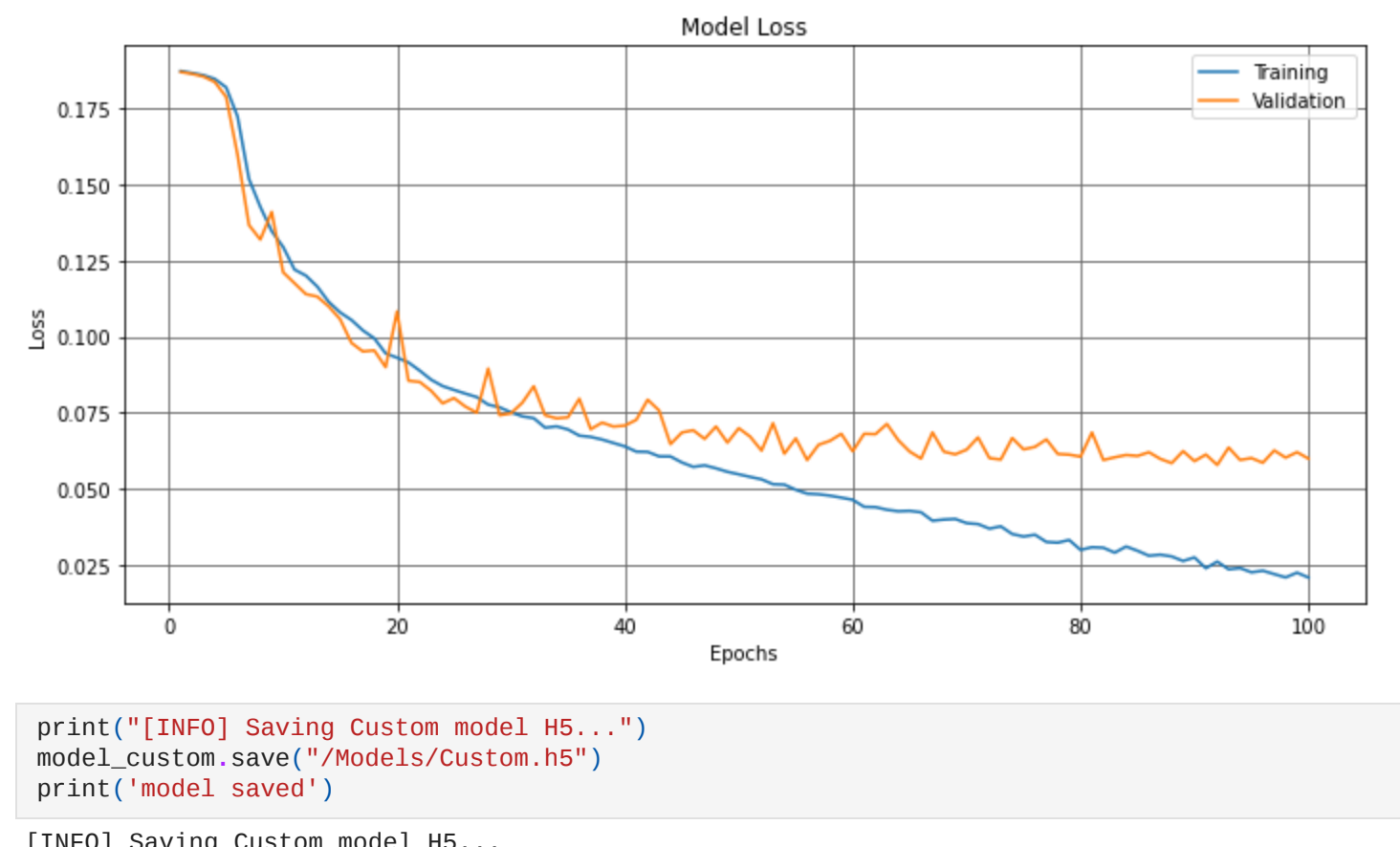
4) Accuracy check of Custom model

In [56]:
```
print("[INFO] Calculating Custom model accuracy")
scores = model_custom.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
[INFO] Calculating Custom model accuracy
34/34 [==============================] - 22s 637ms/step - loss: 0.0608 - accuracy: 0.8422
Test Accuracy: 84.21541452407837
```

In [57]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(1, len(history.epoch) + 1)


plt.figure(figsize=(10,5))

plt.plot(epochs_range, acc, label='Training')
plt.plot(epochs_range, val_acc, label='Validation')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10,5))

plt.plot(epochs_range, loss, label='Training')
plt.plot(epochs_range, val_loss, label='Validation')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.tight_layout()
plt.show()
```





In [58]:
```
print("[INFO] Saving Custom model H5...")
model_custom.save("/Models/Custom.h5")
print('model saved')
```

```
[INFO] Saving Custom model H5...
model saved
```