DATA698 - Analytics Master's Research Project

Data Collection and Analysis

Title: Integrate Image Recognition technology in Radiology Practice

Prof. Dr. Paul J. Bailo

CUNY -SPS

Date: Nov 11, 2022

By: Anjal Hussan

Table of Contents

Abstract

The purpose of this study is to explore some of the image recognition technology that exists currently and how we can take leverage this technology in radiology. The study seeks to answer various questions for example, how image recognition works? How can we integrate image recognition technology with radiology practice? Is image recognition technology reliable? Can Machine Learning (ML) applications help the Radiologist to detect anomalies as well as other abnormalities?

Introduction

Various clinical diagnosis requires medical imaging, such as early detection, monitoring, diagnosis, and treatment evaluation of various health conditions. In the health care system, there has been a significant increase in demand for medical image services, e.g., Radiography, endoscopy, Computed Tomography (CT), Mammography Images (MG), Ultrasound images, Magnetic Resonance Imaging (MRI), Magnetic Resonance Angiography (MRA), Nuclear medicine imaging, Positron Emission Tomography (PET) and pathological tests. Due to a shortage of radiologists, medical images can often be challenging to analyze and time-consuming. Machine Learning (ML) applications can help the Radiologist to detect anomalies as well as other abnormalities. ML applications can function without being specifically programmed, which learn from data and make predictions or decisions based on past data.

The term machine learning refers to a wide range of statistical techniques to analyze algorithms that iteratively improve in response to training data to build models for autonomous predictions. In other words, computer programs become more efficient as they learn from data. The objective of a machine learning algorithm is to develop a mathematical model that fits the data. Once this model fits known data, it can be used to predict the labels of new data. The field of radiology involves inherently interpreting data—extracting features from images and analyzing them with the help of a wide knowledge base that includes these features—which provides an excellent opportunity for implementing ML tools in practice.

Data

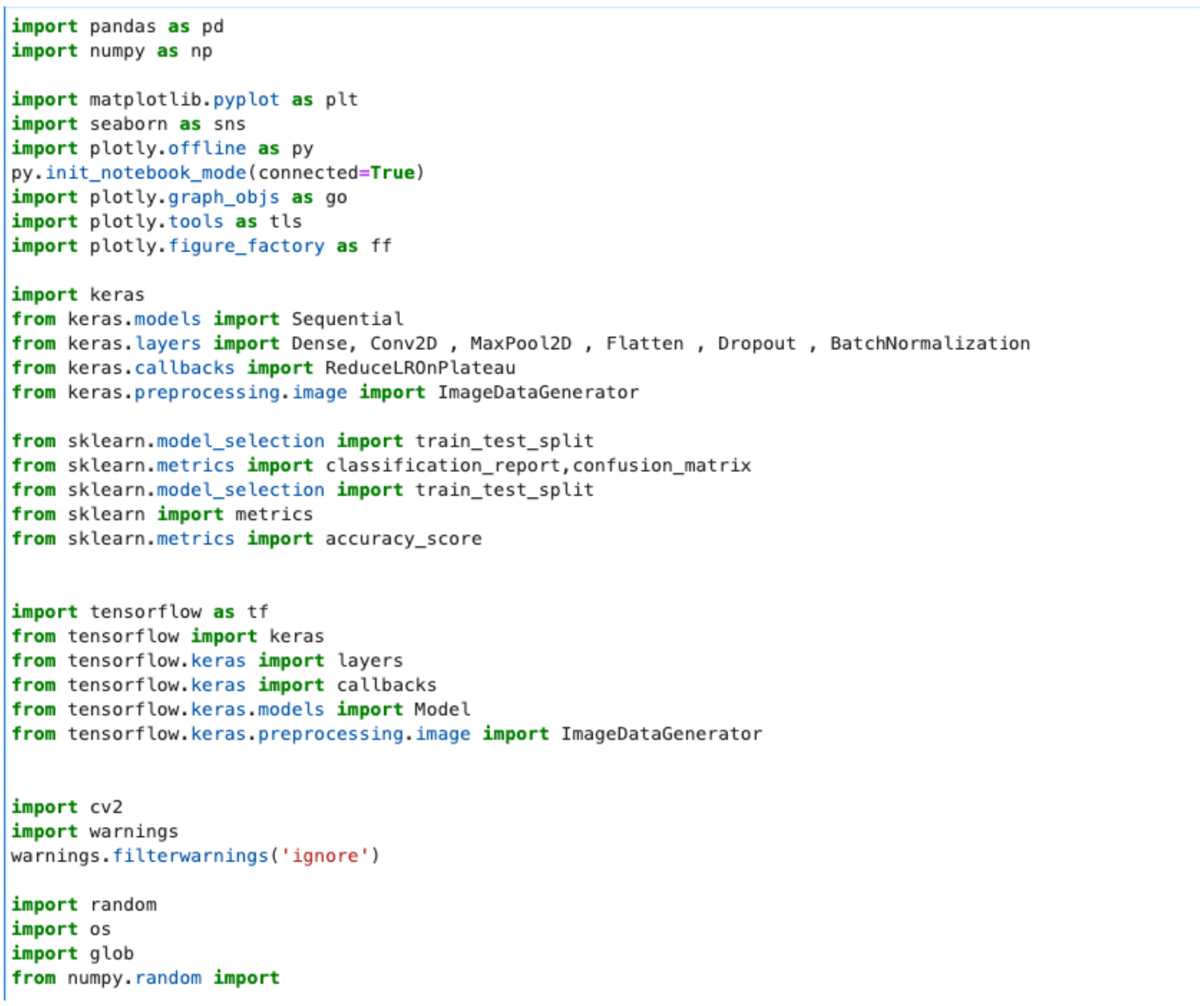
The data set we are going to use is the “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification” which is publicly available in the Mendeley.com (https://data.mendeley.com/datasets/rscbjbr9sj/2).

The dataset contains Chest X-Ray images and are already split into a training set and a testing set of independent patients. The dataset was used in many study and have been labelled properly.

Exploratory Data Analysis

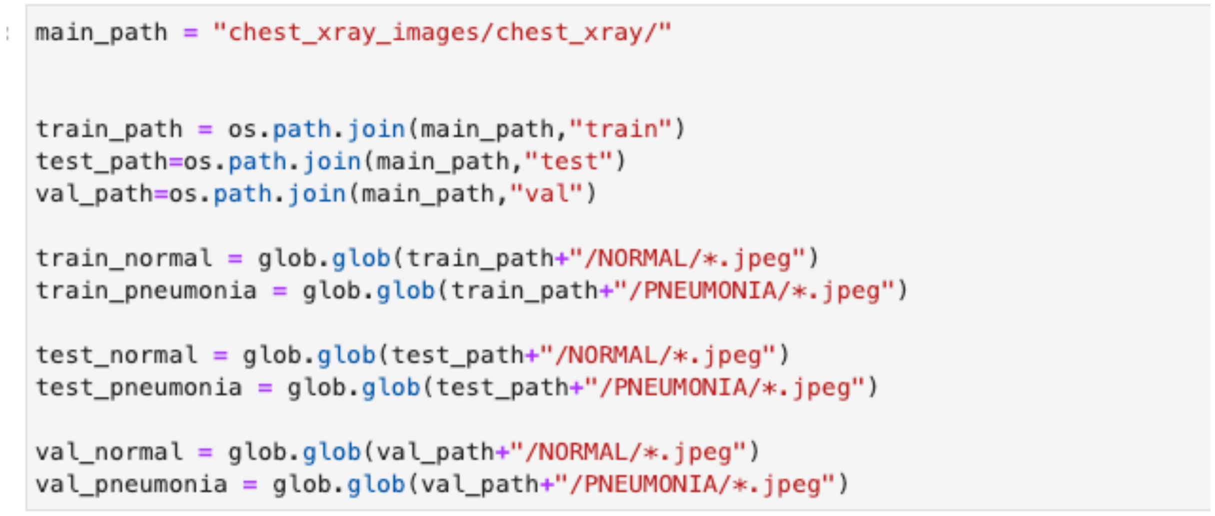
We will explore the data sets and perform the exploratory data analysis in python.

First, we start by importing libraries in the Python notebook:



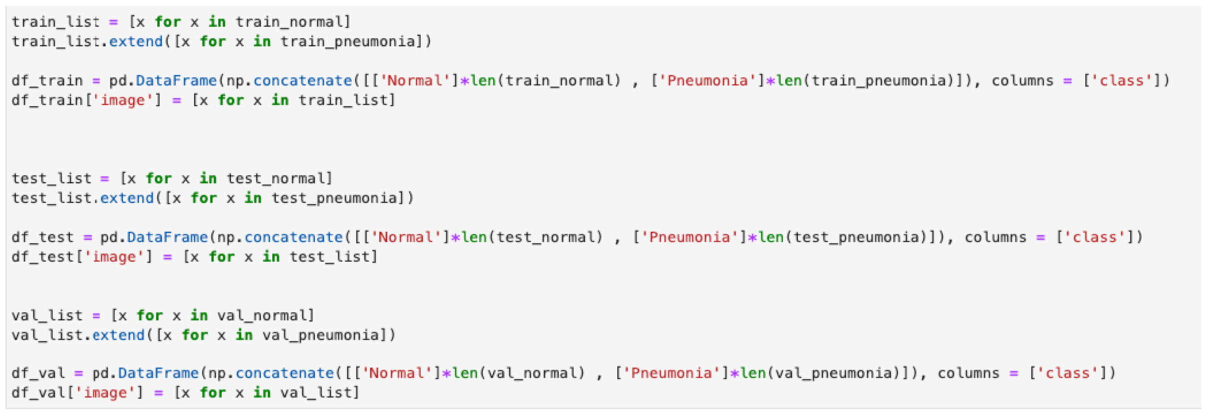
```python
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff

import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score


import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import callbacks
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator


import cv2
import warnings
warnings.filterwarnings('ignore')

import random
import os
import glob
from numpy.random import
```

Once we import the libraries it's time to read the data from our local directory. The data is already divided into train and test and val directory. Let's define the path for the directories.



```
main_path = "chest_xray_images/chest_xray/"


train_path = os.path.join(main_path,"train")
test_path=os.path.join(main_path,"test")
val_path=os.path.join(main_path,"val")

train_normal = glob.glob(train_path+"/NORMAL/*.jpeg")
train_pneumonia = glob.glob(train_path+"/PNEUMONIA/*.jpeg")

test_normal = glob.glob(test_path+"/NORMAL/*.jpeg")
test_pneumonia = glob.glob(test_path+"/PNEUMONIA/*.jpeg")

val_normal = glob.glob(val_path+"/NORMAL/*.jpeg")
val_pneumonia = glob.glob(val_path+"/PNEUMONIA/*.jpeg")
```

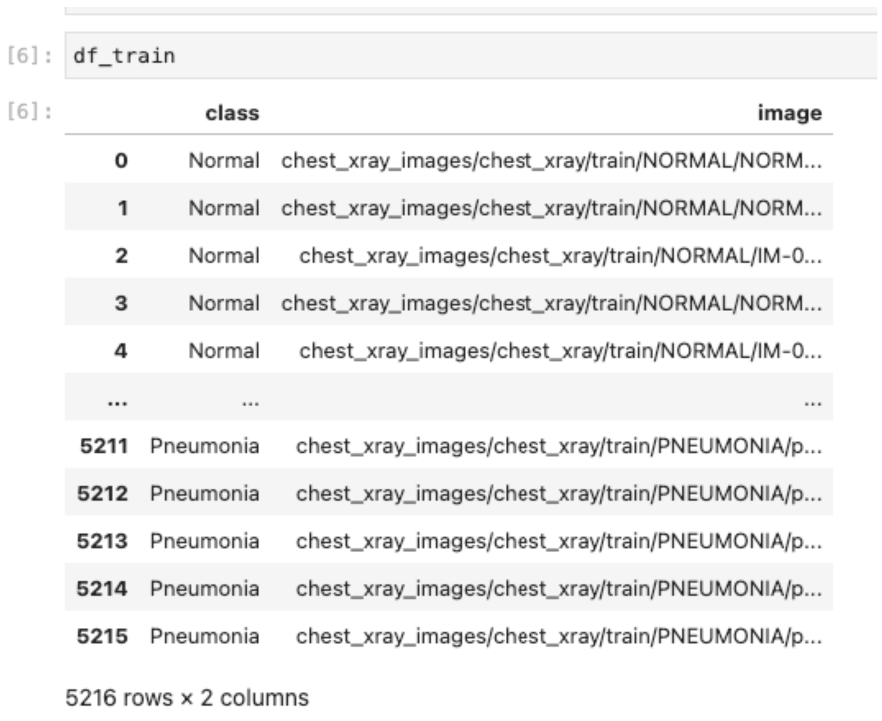
Now that we have the directories defined, we can combine the normal and pneumonia datasets for each of the data pool



```
train_list = [x for x in train_normal]
train_list.extend([x for x in train_pneumonia])

df_train = pd.DataFrame(np.concatenate([['Normal']*len(train_normal) , ['Pneumonia']*len(train_pneumonia)]), columns = ['class'])
df_train['image'] = [x for x in train_list]


test_list = [x for x in test_normal]
test_list.extend([x for x in test_pneumonia])

df_test = pd.DataFrame(np.concatenate([['Normal']*len(test_normal) , ['Pneumonia']*len(test_pneumonia)]), columns = ['class'])
df_test['image'] = [x for x in test_list]


val_list = [x for x in val_normal]
val_list.extend([x for x in val_pneumonia])

df_val = pd.DataFrame(np.concatenate([['Normal']*len(val_normal) , ['Pneumonia']*len(val_pneumonia)]), columns = ['class'])
df_val['image'] = [x for x in val_list]
```

Now that we have our dataset combined. We will explore the dimension of the data frames:

Training data set:



```
[6]: df_train
```

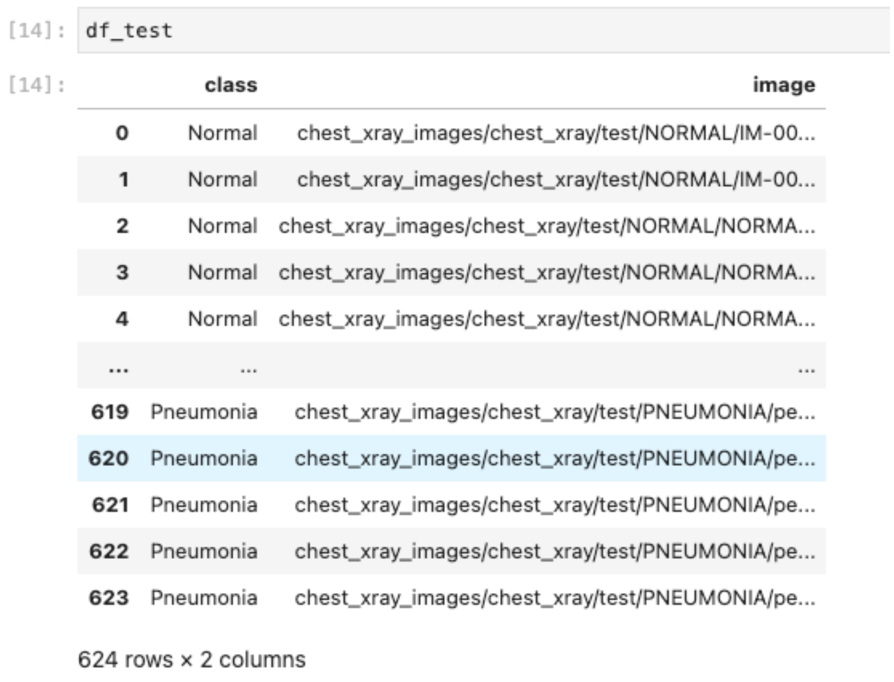
[6]:

	class	image
0	Normal	chest_xray_images/chest_xray/train/NORMAL/NORM...
1	Normal	chest_xray_images/chest_xray/train/NORMAL/NORM...
2	Normal	chest_xray_images/chest_xray/train/NORMAL/IM-0...
3	Normal	chest_xray_images/chest_xray/train/NORMAL/NORM...
4	Normal	chest_xray_images/chest_xray/train/NORMAL/IM-0...
...	...	...
5211	Pneumonia	chest_xray_images/chest_xray/train/PNEUMONIA/p...
5212	Pneumonia	chest_xray_images/chest_xray/train/PNEUMONIA/p...
5213	Pneumonia	chest_xray_images/chest_xray/train/PNEUMONIA/p...
5214	Pneumonia	chest_xray_images/chest_xray/train/PNEUMONIA/p...
5215	Pneumonia	chest_xray_images/chest_xray/train/PNEUMONIA/p...

5216 rows × 2 columns

Looks like the training data set has 5216 rows and 2 columns. One column is the label and the other column is the path to the image.

Testing data set:



```
[14]: df_test
```

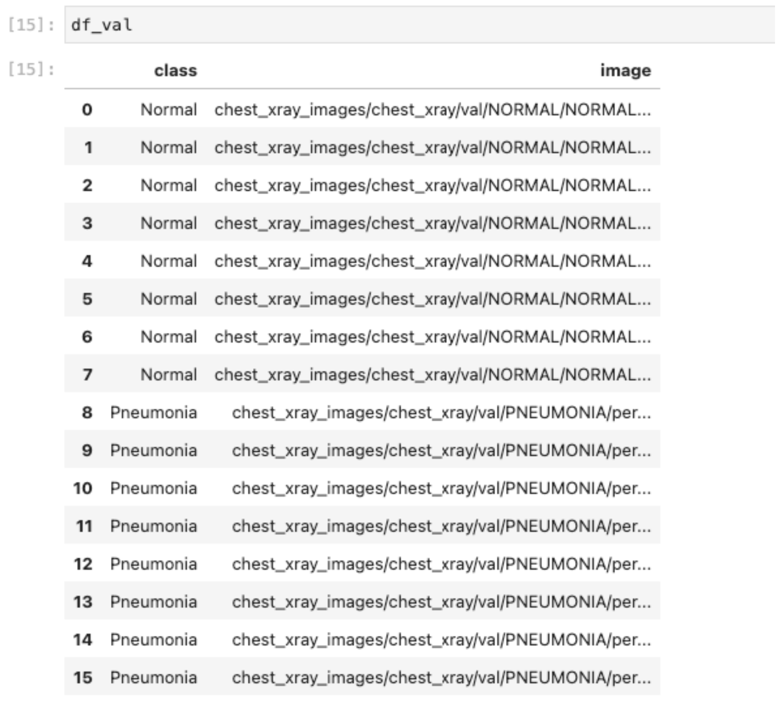
[14]:

	class	image
0	Normal	chest_xray_images/chest_xray/test/NORMAL/IM-00...
1	Normal	chest_xray_images/chest_xray/test/NORMAL/IM-00...
2	Normal	chest_xray_images/chest_xray/test/NORMAL/NORMA...
3	Normal	chest_xray_images/chest_xray/test/NORMAL/NORMA...
4	Normal	chest_xray_images/chest_xray/test/NORMAL/NORMA...
...	...	...
619	Pneumonia	chest_xray_images/chest_xray/test/PNEUMONIA/pe...
620	Pneumonia	chest_xray_images/chest_xray/test/PNEUMONIA/pe...
621	Pneumonia	chest_xray_images/chest_xray/test/PNEUMONIA/pe...
622	Pneumonia	chest_xray_images/chest_xray/test/PNEUMONIA/pe...
623	Pneumonia	chest_xray_images/chest_xray/test/PNEUMONIA/pe...

624 rows × 2 columns

Looks like the testing data set has 624 rows.

Val data set:



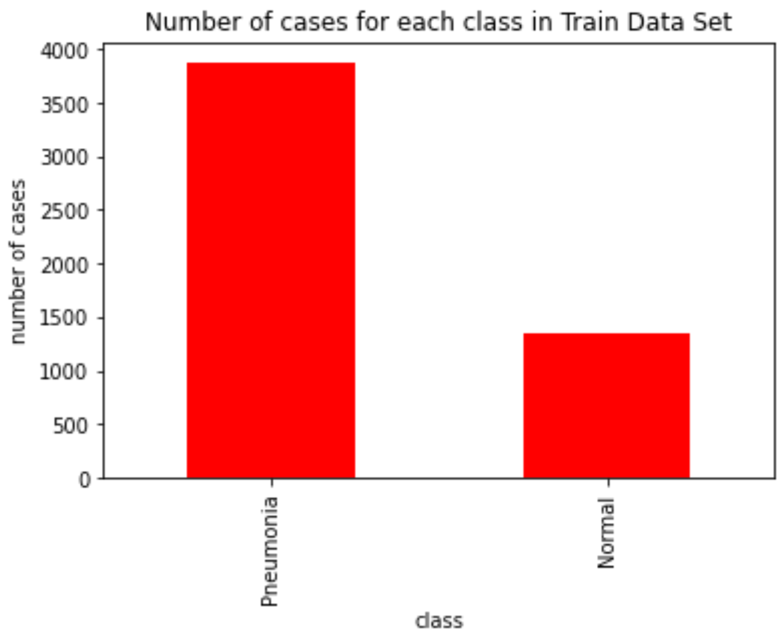
```
[15]: df_val
```

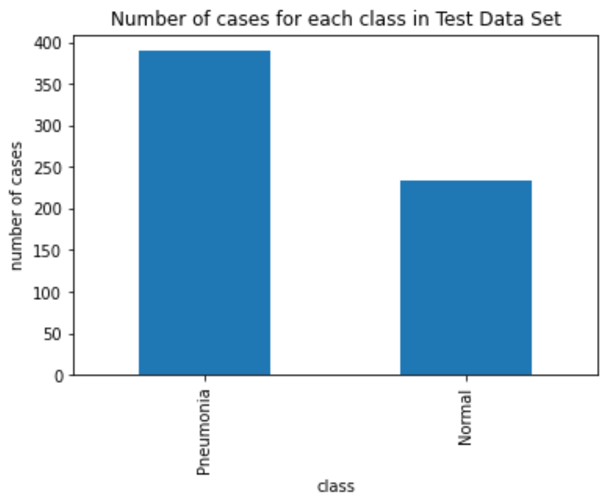
[15]:

	class	image
0	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
1	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
2	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
3	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
4	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
5	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
6	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
7	Normal	chest_xray_images/chest_xray/val/NORMAL/NORMAL...
8	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
9	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
10	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
11	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
12	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
13	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
14	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...
15	Pneumonia	chest_xray_images/chest_xray/val/PNEUMONIA/per...

The Val data set has only 15 rows of data. Among the 15 rows, there 7 rows with normal x-ray image and 8 rows with x-ray image of the patient who had pneumonia.

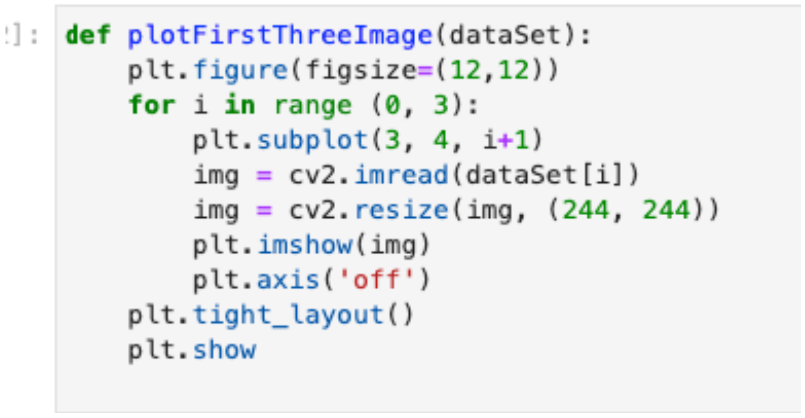
Let's explore more with some visualization:





From the bar charts above, it appears that the number of pneumonia cases are higher than normal cases in both the training and testing data set.

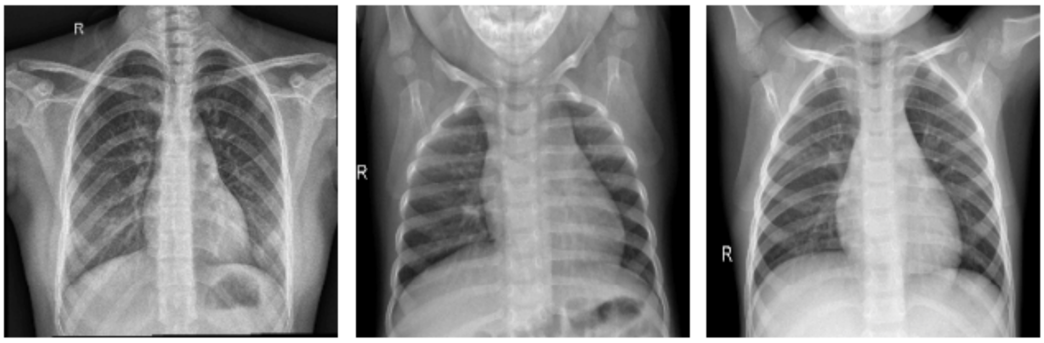
Using Python, we are now going to create a function which will essentially read the data from the given path and plot the image.



```
]: def plotFirstThreeImage(dataSet):
       plt.figure(figsize=(12,12))
       for i in range (0, 3):
           plt.subplot(3, 4, i+1)
           img = cv2.imread(dataSet[i])
           img = cv2.resize(img, (244, 244))
           plt.imshow(img)
           plt.axis('off')
       plt.tight_layout()
       plt.show
```

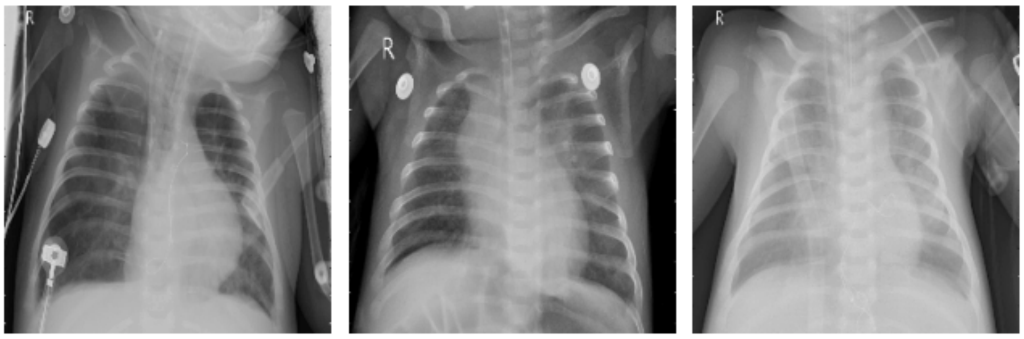
Now that we have the function ready, let's plot first three images of the train_normal data set:

```
[13]: plotFirstThreeImage(train_normal)
```



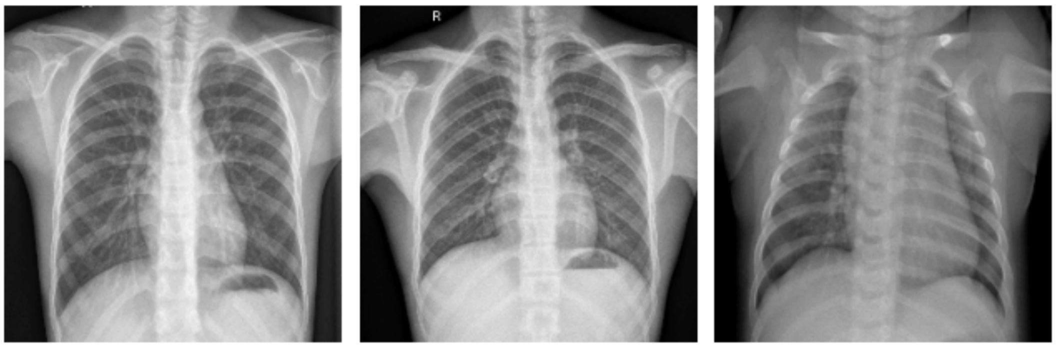
Plot first three images from train_pneumonia data set:

```
[14]: plotFirstThreeImage(train_pneumonia)
```



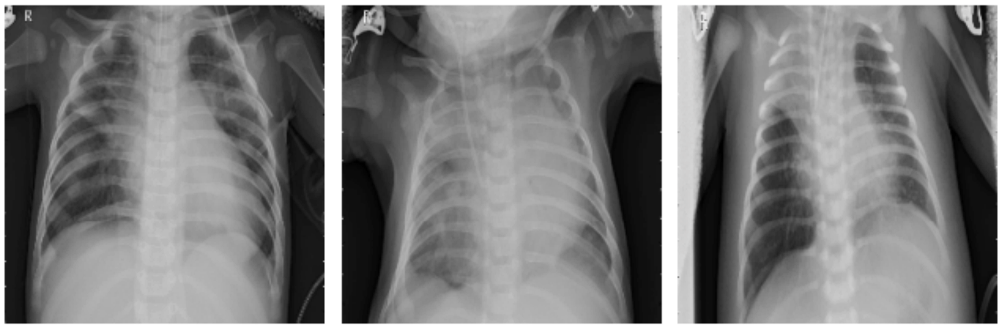
Plot first three images from test_normal data set:

```
[58]: plotFirstThreeImage(test_normal)
```



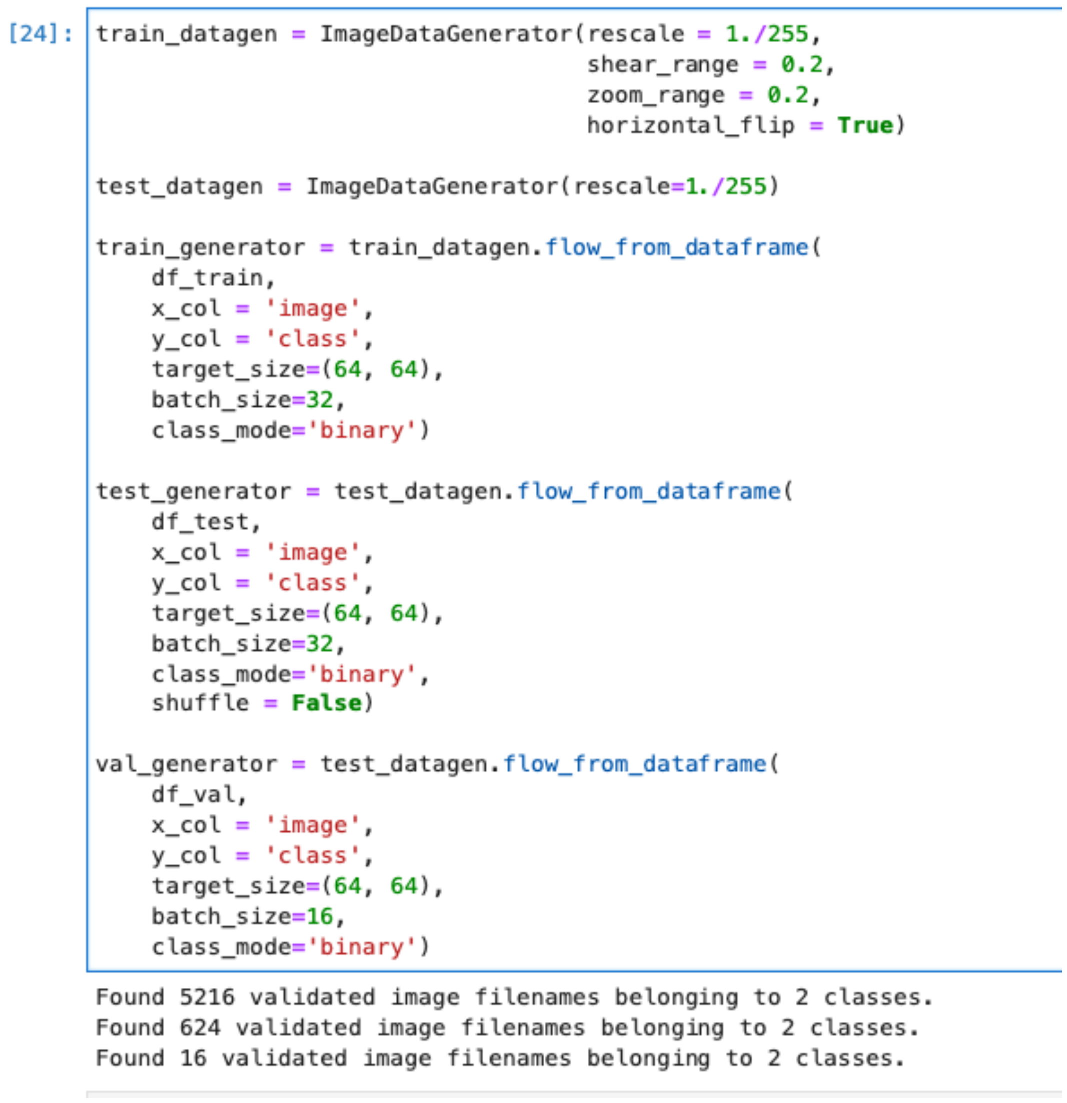
Plot first three images from test_pneumonia data set:

```
[59]: plotFirstThreeImage(test_pneumonia)
```



Data Augmentation

Keras ImageDataGenerator is used for getting the input of the original data and further, it makes the transformation of this data on a random basis and gives the output resultant containing only the data that is newly transformed. It does not add the data. Keras image data generator class is also used to carry out data augmentation where we aim to gain the overall increment in the generalization of the model. Operations such as rotations, translations, shearin, scale changes, and horizontal flips are carried out randomly in data augmentation using an image data generator. Let's take a look how the ImageDataGenerator code looks like:



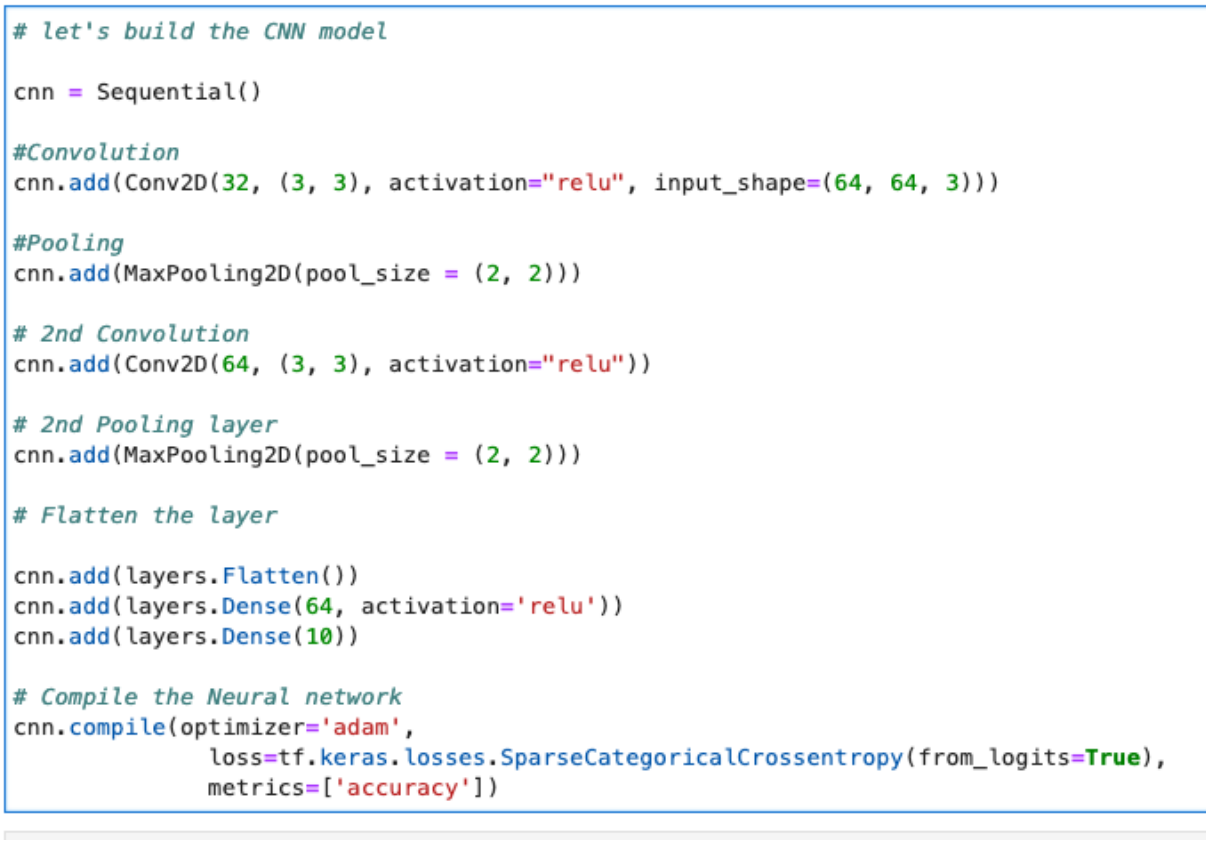
```
[24]: train_datagen = ImageDataGenerator(rescale = 1./255,
                                         shear_range = 0.2,
                                         zoom_range = 0.2,
                                         horizontal_flip = True)

      test_datagen = ImageDataGenerator(rescale=1./255)

      train_generator = train_datagen.flow_from_dataframe(
          df_train,
          x_col = 'image',
          y_col = 'class',
          target_size=(64, 64),
          batch_size=32,
          class_mode='binary')

      test_generator = test_datagen.flow_from_dataframe(
          df_test,
          x_col = 'image',
          y_col = 'class',
          target_size=(64, 64),
          batch_size=32,
          class_mode='binary',
          shuffle = False)

      val_generator = test_datagen.flow_from_dataframe(
          df_val,
          x_col = 'image',
          y_col = 'class',
          target_size=(64, 64),
          batch_size=16,
          class_mode='binary')

      Found 5216 validated image filenames belonging to 2 classes.
      Found 624 validated image filenames belonging to 2 classes.
      Found 16 validated image filenames belonging to 2 classes.
```

Model

A convolutional neural network (CNN) is a network architecture for deep learning which learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects. A great way to use deep learning to classify images is to build a convolutional neural network (CNN). The Keras library in Python makes it simple to build a CNN. Keras CNN is the neural network that makes the use of convolution, which is a mathematical operation carried out on matrices. The neural network consists of convolution layers containing the filters, like the matrices containing numbers.

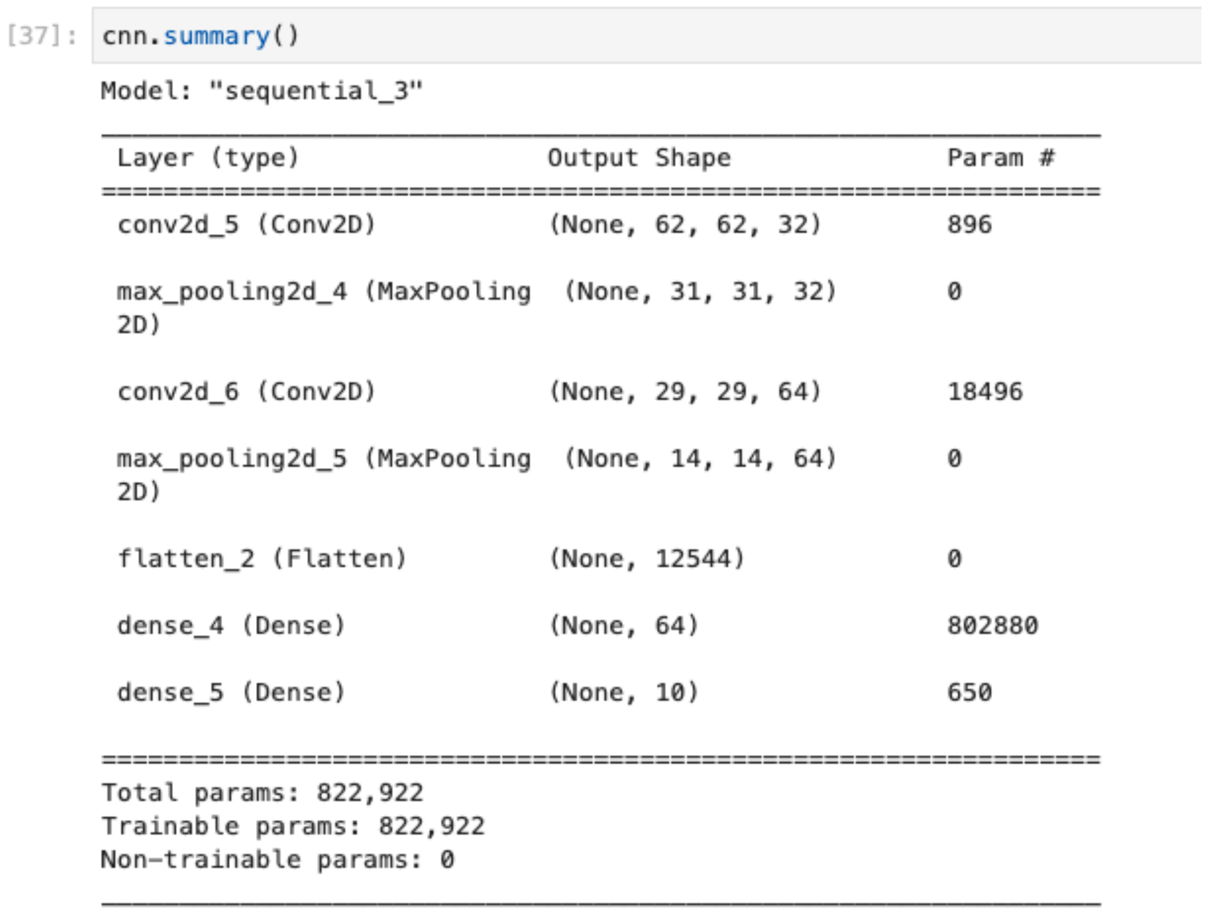
The model we are going to use is the Sequential model. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

Our Model looks like this code:



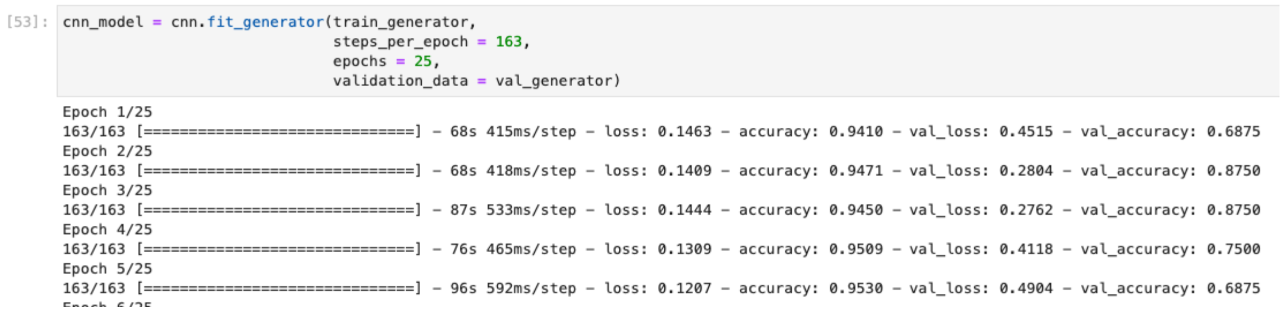
```
# let's build the CNN model

cnn = Sequential()

#Convolution
cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(64, 64, 3)))

#Pooling
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# 2nd Convolution
cnn.add(Conv2D(64, (3, 3), activation="relu"))

# 2nd Pooling layer
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# Flatten the layer

cnn.add(layers.Flatten())
cnn.add(layers.Dense(64, activation='relu'))
cnn.add(layers.Dense(10))

# Compile the Neural network
cnn.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

When we compile the model and print the summary, the summary prints pertinent information of the model:



```
[37]: cnn.summary()

Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_4 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_2 (Flatten)         (None, 12544)             0

 dense_4 (Dense)             (None, 64)                802880

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 822,922
Trainable params: 822,922
Non-trainable params: 0
_________________________________________________________________
```

Now that we have a model and our data is ready to be provided as input, let's try to train the model.



```
[53]: cnn_model = cnn.fit_generator(train_generator,
                                    steps_per_epoch = 163,
                                    epochs = 25,
                                    validation_data = val_generator)

Epoch 1/25
163/163 [==============================] - 68s 415ms/step - loss: 0.1463 - accuracy: 0.9410 - val_loss: 0.4515 - val_accuracy: 0.6875
Epoch 2/25
163/163 [==============================] - 68s 418ms/step - loss: 0.1409 - accuracy: 0.9471 - val_loss: 0.2804 - val_accuracy: 0.8750
Epoch 3/25
163/163 [==============================] - 87s 533ms/step - loss: 0.1444 - accuracy: 0.9450 - val_loss: 0.2762 - val_accuracy: 0.8750
Epoch 4/25
163/163 [==============================] - 76s 465ms/step - loss: 0.1309 - accuracy: 0.9509 - val_loss: 0.4118 - val_accuracy: 0.7500
Epoch 5/25
163/163 [==============================] - 96s 592ms/step - loss: 0.1207 - accuracy: 0.9530 - val_loss: 0.4904 - val_accuracy: 0.6875
```

After 25th iteration, the accuracy seems to be consistent around 96.70% and the accuracy for validation data is around 93.75%

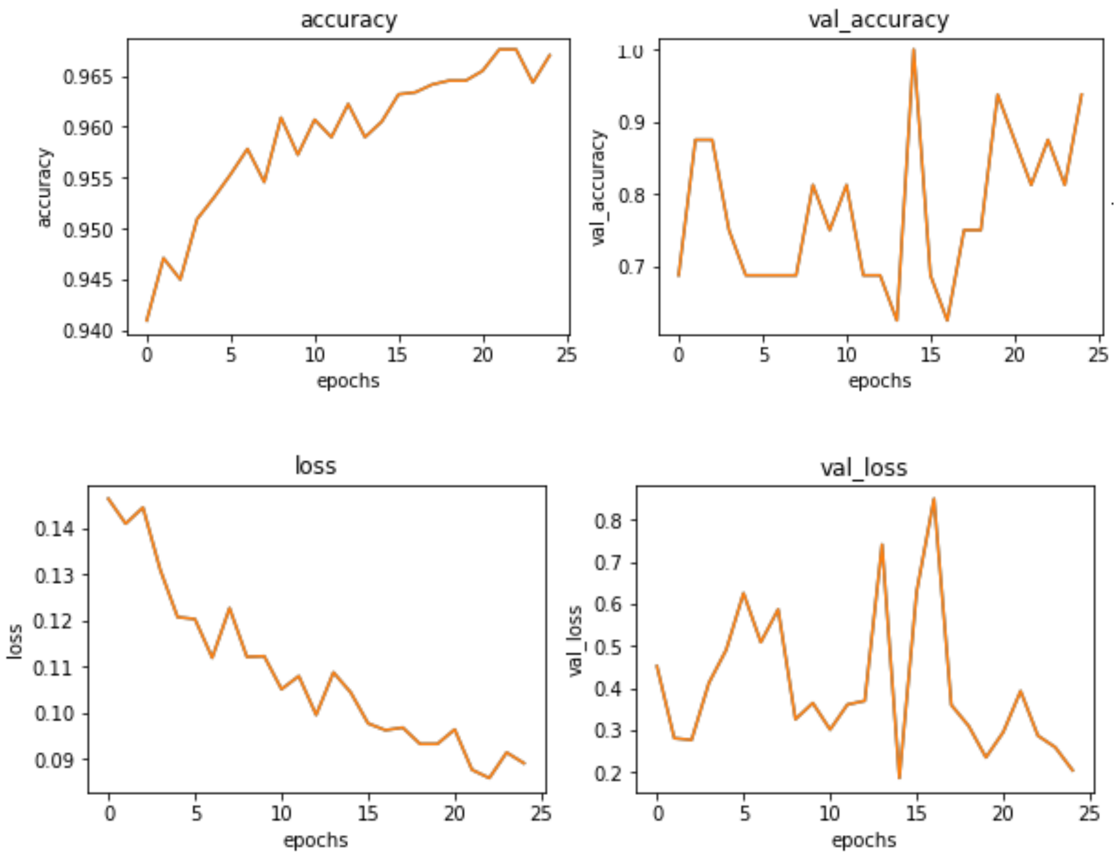
```
Epoch 22/25
163/163 [==============================] - 73s 449ms/step - loss: 0.0876 - accuracy: 0.9676 - val_loss: 0.3924 - val_accuracy: 0.8125
Epoch 23/25
163/163 [==============================] - 129s 793ms/step - loss: 0.0858 - accuracy: 0.9676 - val_loss: 0.2863 - val_accuracy: 0.8750
Epoch 24/25
163/163 [==============================] - 119s 729ms/step - loss: 0.0913 - accuracy: 0.9643 - val_loss: 0.2588 - val_accuracy: 0.8125
Epoch 25/25
163/163 [==============================] - 122s 750ms/step - loss: 0.0890 - accuracy: 0.9670 - val_loss: 0.2041 - val_accuracy: 0.9375
```

Visualizing Model Performance

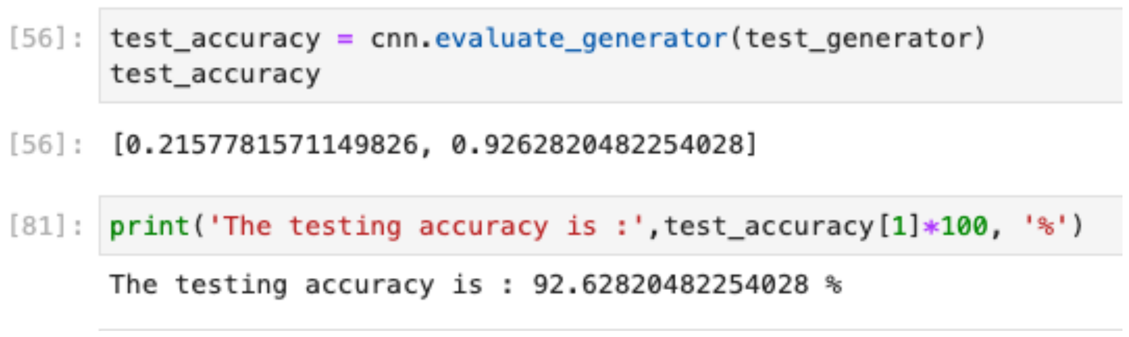
Now that we have a model which is trained, let's visualize the model performance

```
fig, ax = plt.subplots(1, 4, figsize=(20, 3))
ax = ax.ravel()

for i, x in enumerate(['accuracy', 'val_accuracy', 'loss', 'val_loss']):
    ax[i].plot(cnn_model.history[x])
    ax[i].plot(cnn_model.history[x])
    ax[i].set_title(x)
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(x)
```

The plots look like this:



Let’s evaluate our model against our test data:



```
[56]: test_accuracy = cnn.evaluate_generator(test_generator)
      test_accuracy

[56]: [0.2157781571149826, 0.9262820482254028]

[81]: print('The testing accuracy is :',test_accuracy[1]*100, '%')

      The testing accuracy is : 92.62820482254028 %
```

Conclusion

The model we created is a convolutional neural network model and it learned from the input data which we provided. The model was able to classify our test data and accuracy is almost 93%. We can use this kind of model to get help when it comes to integrate machine learning in our radiology technology.

Reference:

Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification”, Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.

https://keras.io/guides/sequential_model/