(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

PNEUMONIA DETECTION

A Report for the Evaluation 3 of Project 2

Submitted by

MRIDUL MAZUMDAR

1613101418 / 16SCSE101124

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under the supervision of

Dr. D Rajesh Kumar

May 2020

TABLE OF CONTENTS

ABSTRACT

The risk of pneumonia is immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. The WHO estimates that over 4 million premature deaths occur annually from household air pollution-related diseases including pneumonia. Over 150 million people get infected with pneumonia on an annual basis especially children under 5 years old. In such regions, the problem can be further aggravated due to the dearth of medical resources and personnel. For example, in Africa’s 57 nations, a gap of 2.3 million doctors and nurses exists. For these populations, accurate and fast diagnosis means everything. It can guarantee timely access to treatment and save much needed time and money for those already experiencing poverty.

This model could help mitigate the reliability and interpretability challenges often faced when dealing with medical imagery.

INTRODUCTION

In recent years, a tremendous amount of enthusiasm for deep learning, a subfield of Machine Learning, where the algorithms are inspired by the working of human brain, has emerged among students,researchers and technologist from every different strata like medical science. All deep learning models use some form of Convolutional Neural Network [9] as their base algorithm. A Convolutional Neural Network(CNN) takes in data, train themselves to recognize the patterns in data and predict the output for a new set of similar data. CNN models exhibit a complex set of interconnections between inputs and outputs in order to preform a variety of tasks such as image recognition, natural language processing, music generation, even medical diagnosis and procedures. Pneumonia is a severe acute respiratory disease which is caused when the air sacks(alveoli) in lungs are infected by some bacteria, virus or fungi to which the patient's body has weak resistance. As a result the alveoli in lungs gets filled with pus and are unable to provide oxygen to the patient's blood. Although it can be treated by medical care and vaccinations , still it's one of the major causes of death worldwide. Figure 1 shows a radio-graph of a healthy pair of lungs.

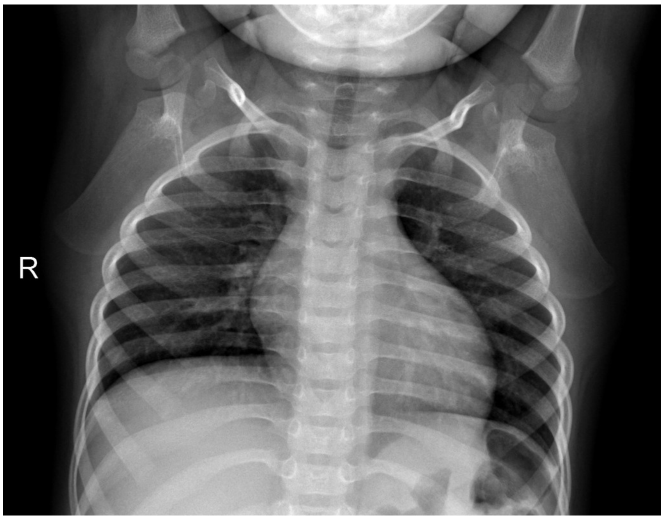


Figure1: Healthy Lungs

Now in an x-ray image the area of inflammation in lungs caused by pneumonia infection generally shows more translucency than the rest of the area as shown in **Figure 2.**

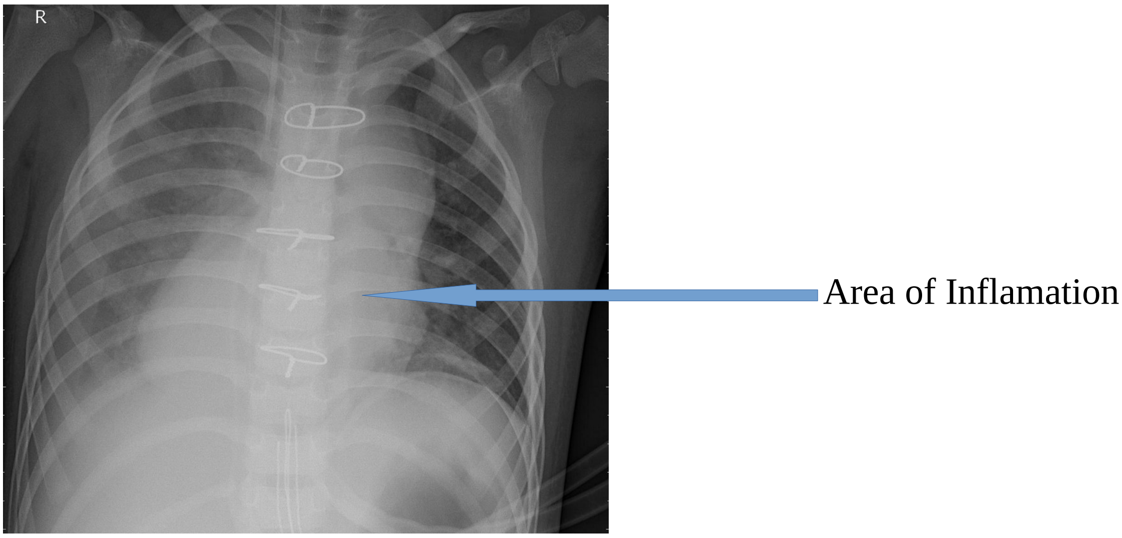


Figure2 Pneumonia Infected lungs.

Still for most radiologists determining whether a patient is suffering from pneumonia infection by analyzing an x-ray image alone can be very difficult given that an x-ray can often be effected by noise generated by radiation scattering, source leakage, sensor errors, electronic devices and implantation. Therefore, building a deep learning model by training a convolutional neural network with thousands of x-ray images to determine whether a patient has pneumonia with high accuracy while taking all the noise generated into the account can be an effective solution to this problem.

Popular Diagnosis Techniques of Pneumonia

Physical exam

Physical examination may sometimes reveal low blood pressure, high heart rate, or low oxygen saturation. The respiratory rate may be faster than normal, and this may occur a day or two before other signs. Examination of the chest may be normal, but it may show decreased chest expansion on the affected side. Harsh breath sounds from the larger airways that are transmitted through the inflamed lung are termed bronchial breathing and are heard on auscultation with a stethoscope. Crackles (rales) may be heard over the affected area during inspiration. Percussion may be dulled over the affected lung, and increased, rather than decreased, vocal resonance distinguishes pneumonia from a pleural effusion.

Imaging

A chest radiograph is frequently used in diagnosis. In people with mild disease, imaging is needed only in those with potential complications, those not having improved with treatment, or those in which the cause is uncertain. If a person is sufficiently sick to require hospitalization, a chest radiograph is recommended. Findings do not always match the severity of disease and do not reliably separate between bacterial infection and viral infection. X-ray presentations of pneumonia may be classified as lobar pneumonia, bronchopneumonia, lobular pneumonia, and interstitial pneumonia. Bacterial, community-acquired pneumonia classically show lung consolidation of one lung segmental lobe, which is known as lobar pneumonia. However, findings may vary, and other patterns are common in other types of pneumonia. Aspiration pneumonia may present with bilateral opacities primarily in the bases of the lungs and on the right side. Radiographs of viral pneumonia may appear normal, appear hyper- inflated, have bilateral patchy areas, or present similar to bacterial pneumonia with lobar consolidation. Radiologic findings may not be present in the early stages of the disease, especially in the presence of dehydration, or may be difficult to be interpreted in the obese or those with a history of lung disease. Complications such as pleural effusion may also be found on chest radiographs. Laterolateral chest radiograph can increase the diagnostic accuracy of lung consolidation and pleural effusion. A CT scan can give additional information in indeterminate cases. CT scan can also provide more details in those with an unclear chest radiograph (for example occult pneumonia in chronic obstructive pulmonary disease) and is able to exclude pulmonary embolism and fungal pneumonia and detecting lung abscess in those who are not responding to treatments. However, CT scan is more expensive, has a higher dose of radiation, and cannot be done at bedside.

Lung ultrasound may also be useful in helping to make the diagnosis. Ultrasound is radiation free and can be done at bedside. However, ultrasound requires specific skills to operate the machine and interpret the findings. It may be more accurate than chest X-ray.

PURPOSE OF THIS PROJECT

The purpose of this project is to study and construct a convolutional neural network model from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with pneumonia. This model could help mitigate the reliability and interpretability challenges often faced when dealing with medical imagery.

MOTIVATION AND SCOPE

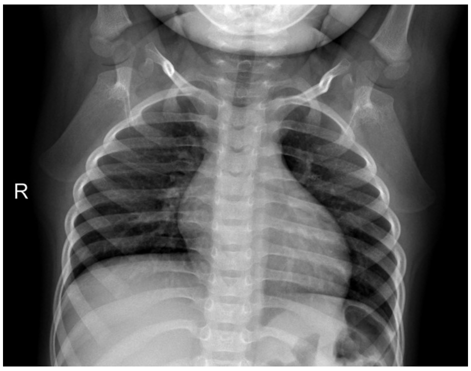
In recent times, CNN-motivated deep learning algorithms have become the standard choice for medical image classifications although the state-of-the-art CNN-based classification techniques pose similar fixated network architectures of the trial-and-error system which have been their designing principle. The proposed work will help doctors better predict pneumonia in minimal time with high efficiency. The aggregation of this will contribute to the health care system for better patient satisfaction and care. This work is in its early stages and can be improved by adding more images to the dataset, incorporating better architectures, training the model based on more transformations and orientations.

PROPOSED MODEL

Pneumonia in India accounts for 20 percent of the death worldwide caused by pneumonia. This project focuses to develop and train a deep learning neural network in order to determine if a person is infected with pneumonia with maximum accuracy.

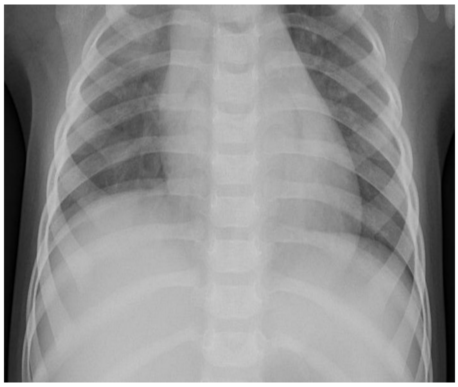
The aim is to allocated more than 3000 images to the training set and about 2000 to the validation set to improve validation accuracy. The primary task of this model is to detect pneumonic infection and classify the input X-ray image by analysing it, as given below:

NORMAL



This normal chest X-ray depicts clear lungs without any areas of abnormal opacification in the image.

PNEUMONIA



This chest X-ray depicts Bacterial pneumonia with partial translucency/opacification

EXISTING MODEL

The main aim of our model is to predict the class of an input image with highest accuracy possible. In order to achieve that objective we need to build a custom CNN (Convolutional Neural Network). Our model's convolutional neural network mainly consists four components:

1. **Convolutional Layer(Cov2D)**

   This layer performs a blend of linear and non-linear operations essential to perform feature extraction. Our model consists of 5 convolutionally interconnected layers which uses two dimensional kernel of size 3*3 and 'Relu' activation function . The first cov2d layer is the input layer which provides input data to the system to be processed further by the consecutive layers of neurons. The input shape of our training data is 150 pixels in height , 150 pixels in width and color gamut value as 1 for gray scale (for RGB color gamut the value is 3), represented as 'input_shape = (150,150,1) ' .

2. **Max-Pooling Layer**

   A pooling layer provides a typical downsampling operation which reduces the in-plane dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions, and decrease the number of subsequent learnable parameters. It is of note that there is no learnable parameter in any of the pooling layers, whereas filter size, stride, and padding are hyperparameters in pooling operations, similar to convolution operations.
   This layer typically reduces the dimensionality of the features and decreases the number of trainable parameters[10] to avoid over-fitting. Our model consists of 5 max-pooling layers, each of filter size 2*2 .

3. **Dense Layer**

   The feature maps[10] are converted into a 1D array after being extracted from the last convolutional or pooling layer, this process is also called flattening.

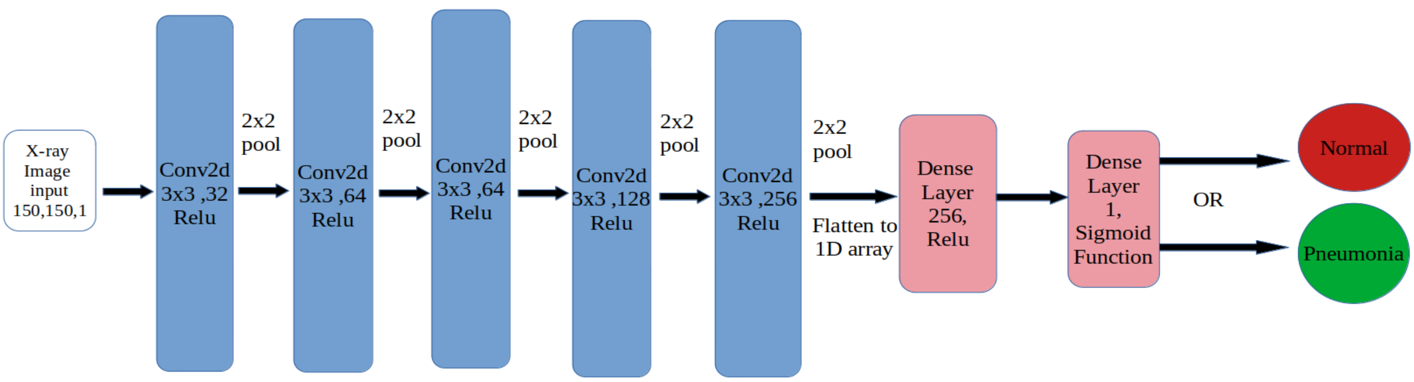
After the flattening process the output are connected to a fully connected layer called the dense layer which also uses ‘Relu’ as activation function.

4. Output Layer

This layer is the last dense layer of our model .It gives only one output at a time and uses ‘sigmoid function’ as activation function which gives a probability score from 0 to 1 to the output of the previous layer . The output with the highest probability score is produced by the output layer.

Layer(type)	**Output Shape**	**Param #**
Conv2D	(None,150,150,32)	320
MaxPooling2D	(None,75,75,32)	0
Conv2D	(None,75,75,64)	18496
MaxPooling2D	(None,38,38,64)	0
Conv2D	(None,38,38,64)	36928
MaxPooling2D	(None,19,19,64)	0
Conv2D	(None,19,19,128)	73856
MaxPooling2D	(None,10,10,128)	0
Conv2D	(None,10,10,256)	295168
MaxPooling2D	(None,5,5,256)	0
Flatten	(None,6400)	0
Dense	(None,128)	819328
Dropout	(None,128)	0
Dense	(None,1)	129

Neural Network’s Configuration

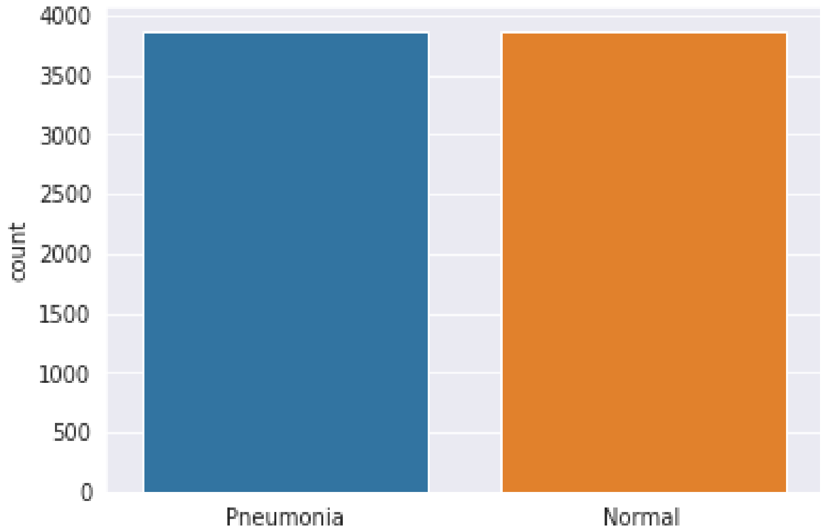


CNN Architecture of our model

METHODOLOGY ADOPTED

1. Dataset Distribution

The original dataset consists of two main folders (i.e., training and validation folders) and two subfolders containing pneumonia (P) and normal (N) chest X-ray images, respectively. A total of 7,750 X-ray images of anterior-posterior chests were carefully chosen from retrospective paediatric patients between 1 and 5 years old. The entire chest X-ray imaging was conducted as part of patients' routine medical care. To balance the proportion of data assigned to the training and validation set, the original data category is modified. It has been rearranged into training and validation set only. A total of 7,750 images are allocated to the training set and 468 images were assigned to the validation set to improve validation accuracy**.**



Balanced Dataset

Both 'Normal' and 'Pneumonia' class consist of 3875 images as training and 234 images as testing data.

2. Data Preprocessing and Augmentation

1.) **Rescale** - The rescale operation represents image reduction or magnification during the augmentation process.

2.) **Rotation Range -** The rotation range denotes the range in which the images were randomly rotated during training. For example: 40 degrees.

3.) **Width Shift** - Width shift is the horizontal translation of the image.

4.) **Height Shift -** Height shift is the vertical translation of the image.

5.) **Zoom Range** - Zoom range is the difference in magnification from one end of the zoom range to the other.

6.) **Horizontal Flip** - Flipping the image along vertical axis.

7.) **Vertical Flip –** Flipping the image along horizontal axis.

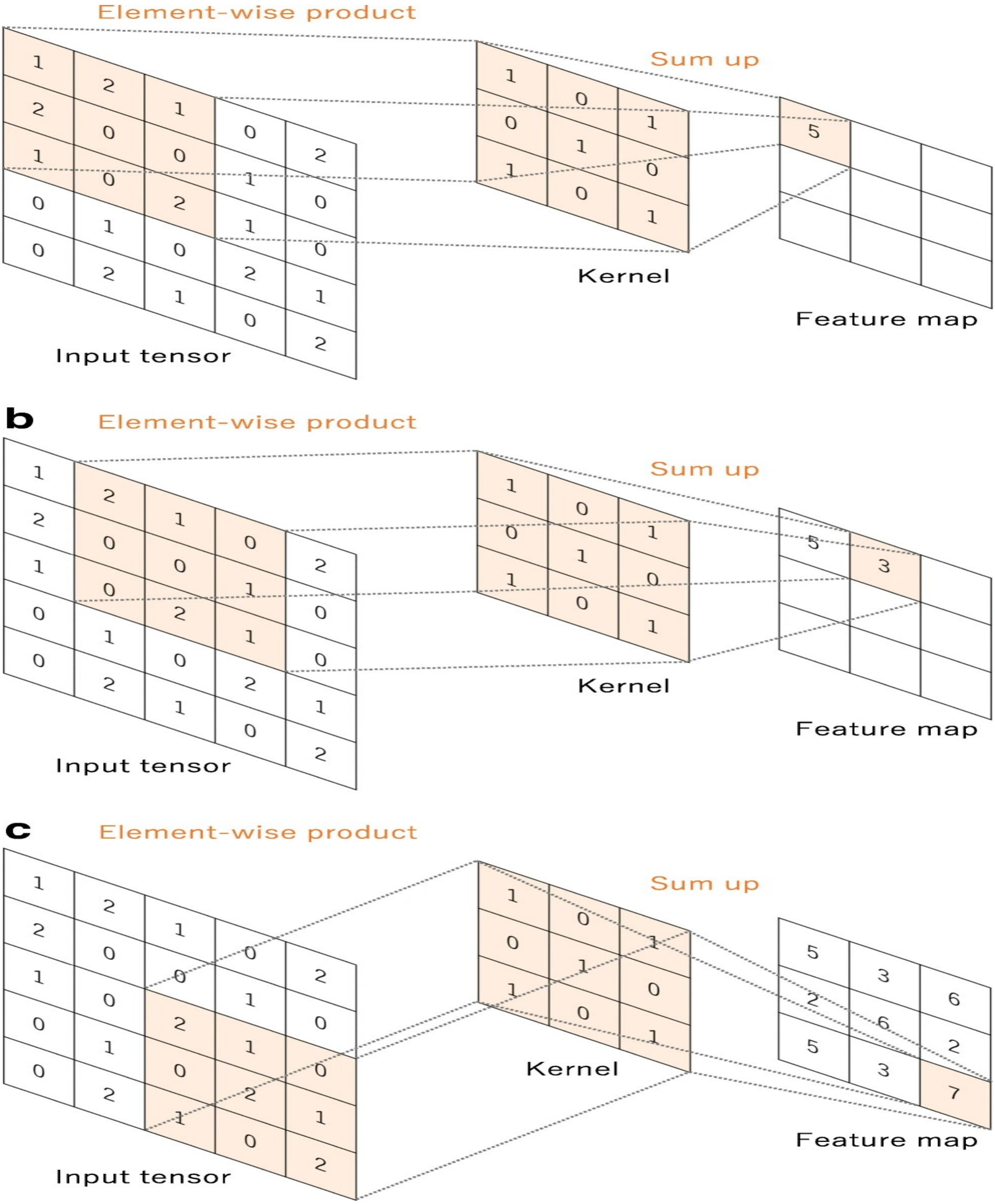
Method	**Settings**
Rotation range	30 degrees
Zoom range	0.2px
Width shift range	0.1px
Height shift range	0.1px
Horizontal flip	False
Vertical flip	False

Augmentation Parameters

3. Feature Extraction:

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (Fig. 3a–c). This procedure is repeated applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can, thus, be considered as different feature extractors (Fig. 3d). Two key hyperparameters that define the convolution operation are size and number of kernels. The former is typically 3×3, but sometimes 5×5 or 7×7. The latter is arbitrary, and determines the depth of output feature maps.

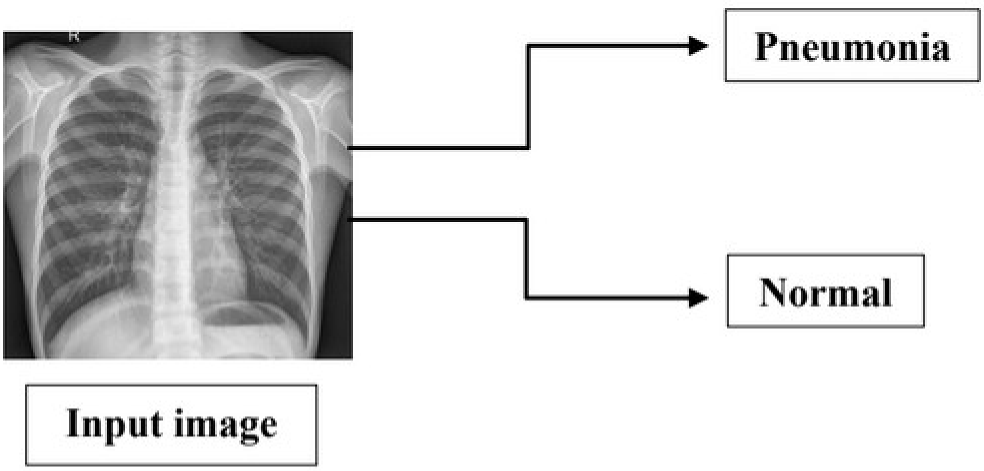


Element-wise product
Sum up
Input tensor
Kernel
Feature map
b
Element-wise product
Sum up
Input tensor
Kernel
Feature map
c
Element-wise product
Sum up
Input tensor
Kernel
Feature map

The convolution operation described above does not allow the center of each kernel to overlap the outermost element of the input tensor, and reduces the height and width of the output feature map compared to the input tensor. Padding, typically zero padding, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, so as to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation. Modern CNN architectures usually employ zero padding to retain in-plane dimensions in order to apply more layers. Without zero padding, each successive feature map would get smaller after the convolution operation.

4. Classification

Classification is a process of categorizing a given set of data into classes, it can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.



RESULTS

To measure and validate the performance of the model, we evaluated it's effectiveness on various parameters given as:

1) **Precision:** Precision is the percentage of positive identifications and total identifications.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

2) **Recall:** Recall is the percentage of relevant identifications predicted by the algorithm.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

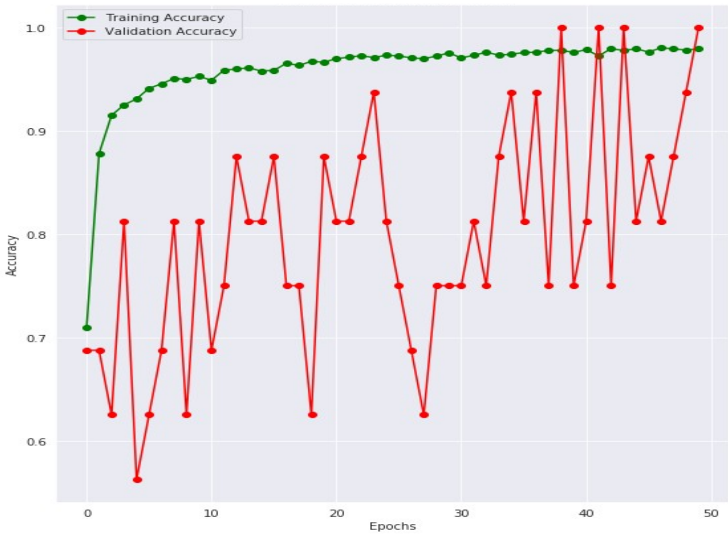
3) **F1-score:** F1-score represents the harmonic mean of precision and recall.

$$F1\,score = \frac{2 * Precision * Recall}{Precision + Recall}$$

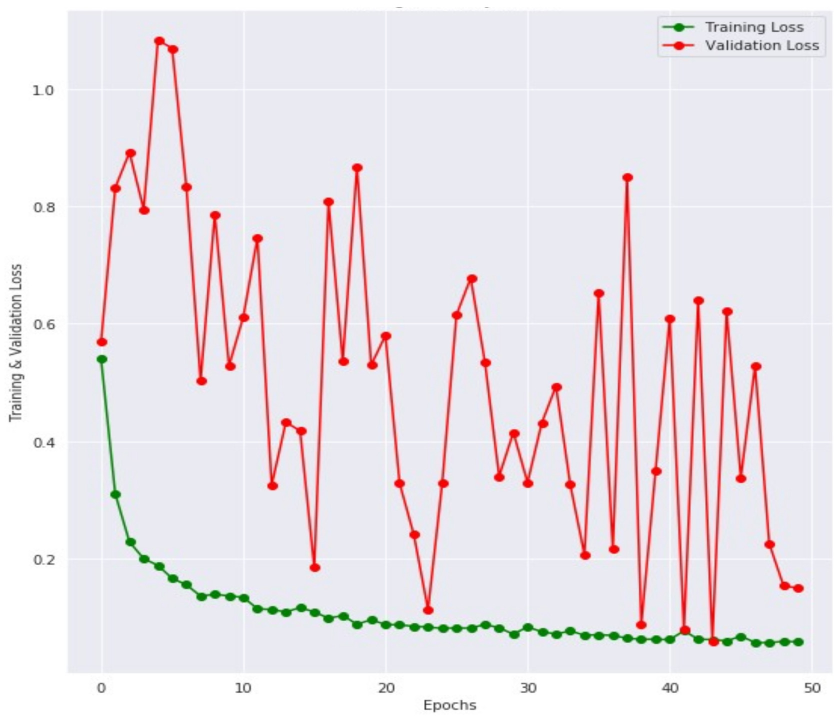
The evaluated scores based parameters mentioned above are give in table below:

Class	Precision	Recall	F1-score
Pneumonia(Class 0)	0.94	0.94	0.94
Normal(Class 1)	0.90	0.89	0.90

The model has been trained till 50 epochs, the training and validation accuracy on every epoch can be seen in graph1 and training and validation loss on graph 2.



Graph1: Training and validation accuracy on each epoch.



Graph2:Training and validation loss on every epoch.

In graph1, we can observe that the average validation accuracy increases with every epoch. In graph 2, we can observe that the average validation loss decreases with every epoch . Minor fluctuations in validation accuracy and loss does not effect the overall performance of the model .

LIMITATIONS

1. Processing and building the model requires fast and efficient processors which is time and cost consuming.

2. 100% accuracy is not achievable.

3. The challenges of large variation in sensing modality which is complicated by human anatomy are faced in medical image analysis.

4. Parameters affecting medical images fluctuate from organ to organ.

5. Medical Images are affected by noise due to sensors, device implantation, electronics leading to inefficiency while detection.

6. The model doesn't take other diagnosis parameters like Blood tests, Pulse Oximetry, Sputum tests into account.

PRACTICAL IMPLEMENTATION

May 7, 2020

[2]:
```
import numpy as np
import pandas as pd


import os

import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D , MaxPool2D , Flatten ,␣
↪Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
import cv2
import os
```

[3]:
```
data_dir='./data/'
labels = ['PNEUMONIA', 'NORMAL']

img_size = 150

def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.
↪IMREAD_GRAYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) #␣
↪Reshaping images to preferred size
                data.append([resized_arr, class_num])
            except Exception as e:
```

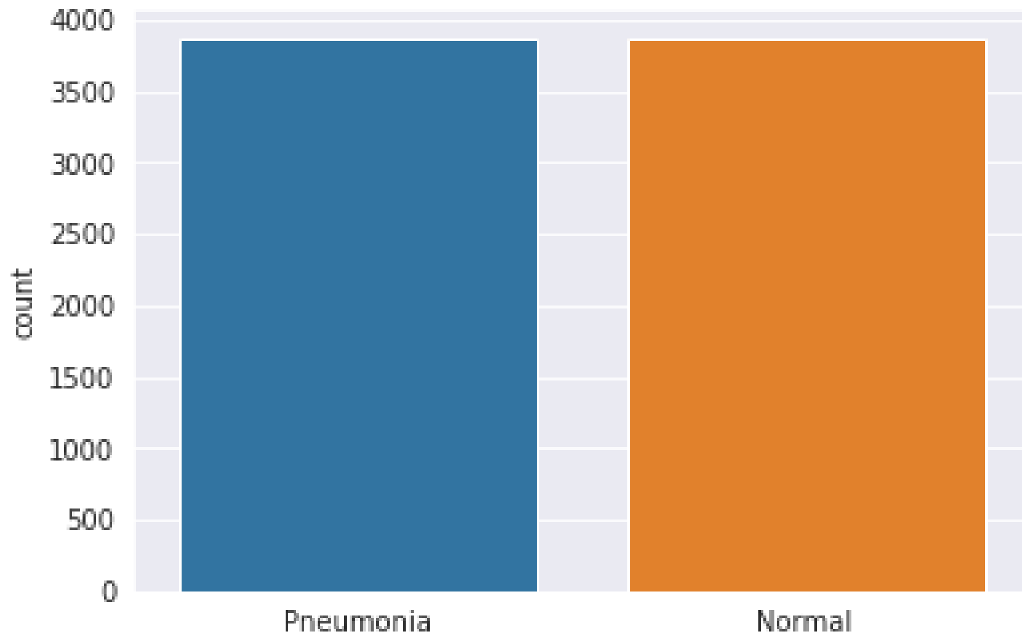
```
                print(e)
    return np.array(data)
```

[4]:
```
train = get_training_data('./data/train')
test = get_training_data('./data/test')
val = get_training_data('./data/val')
```

```
OpenCV(3.4.2) /tmp/build/80754af9/opencv-
suite_1535558553474/work/modules/imgproc/src/resize.cpp:4044: error:
(-215:Assertion failed) !ssize.empty() in function 'resize'

OpenCV(3.4.2) /tmp/build/80754af9/opencv-
suite_1535558553474/work/modules/imgproc/src/resize.cpp:4044: error:
(-215:Assertion failed) !ssize.empty() in function 'resize'

OpenCV(3.4.2) /tmp/build/80754af9/opencv-
suite_1535558553474/work/modules/imgproc/src/resize.cpp:4044: error:
(-215:Assertion failed) !ssize.empty() in function 'resize'

OpenCV(3.4.2) /tmp/build/80754af9/opencv-
suite_1535558553474/work/modules/imgproc/src/resize.cpp:4044: error:
(-215:Assertion failed) !ssize.empty() in function 'resize'
```

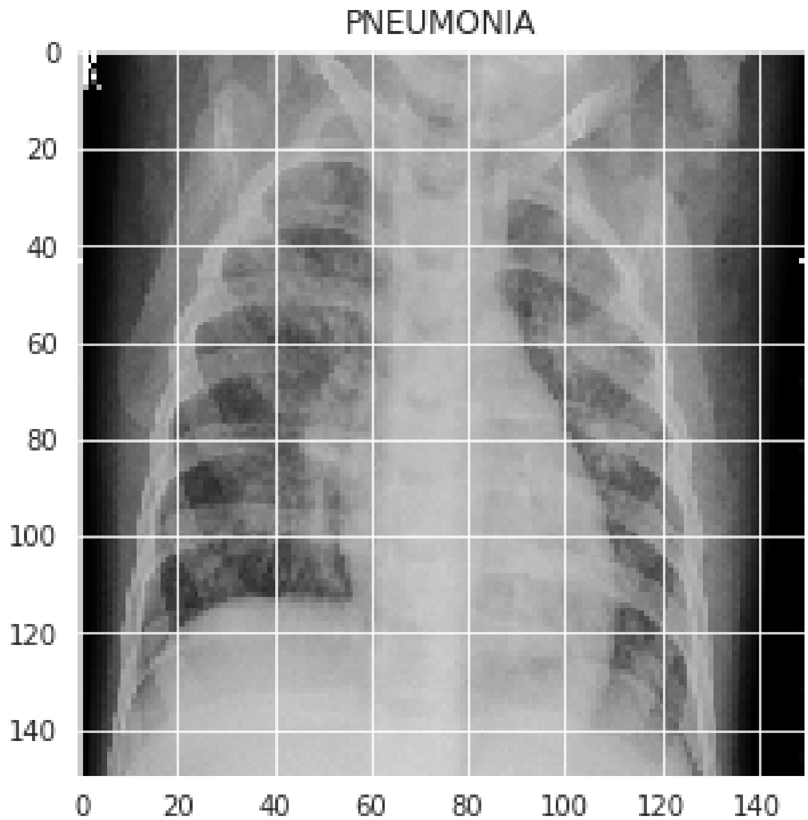
[5]:
```
l = []
for i in train:
    if(i[1] == 0):
        l.append("Pneumonia")
    else:
        l.append("Normal")
sns.set_style('darkgrid')
sns.countplot(l)
```

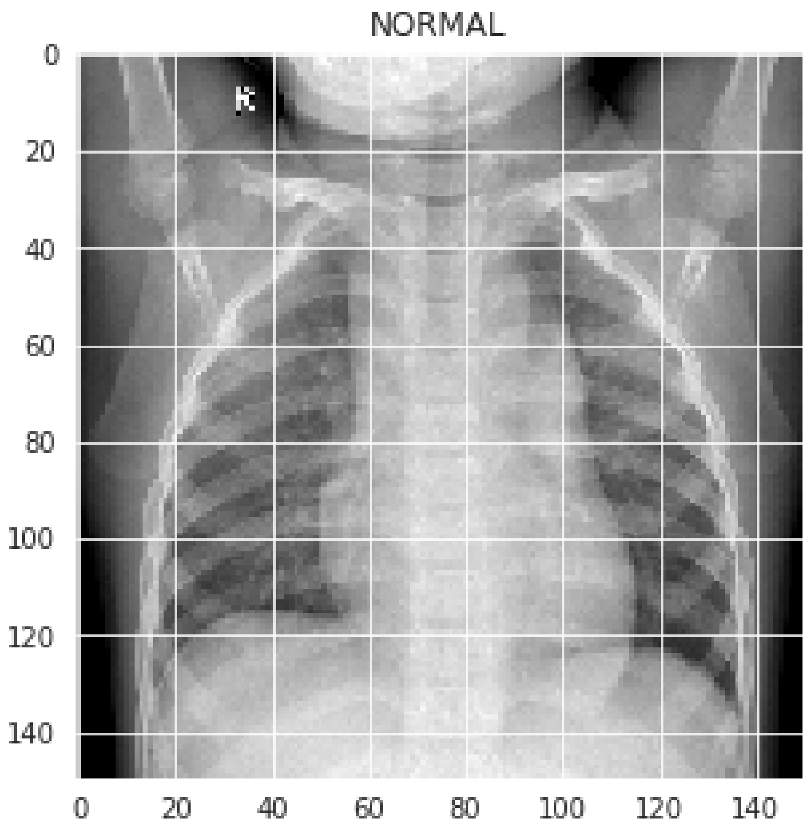
[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f87787b2290>
```



[6]:
```
plt.figure(figsize = (5,5))
plt.imshow(train[0][0], cmap='gray')
plt.title(labels[train[0][1]])

plt.figure(figsize = (5,5))
plt.imshow(train[-1][0], cmap='gray')
plt.title(labels[train[-1][1]])
```

[6]:
```
Text(0.5, 1.0, 'NORMAL')
```


PNEUMONIA
0
20
40
60
80
100
120
140



[7]:
```
x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)
```

[8]:
```
# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255
x_train.shape
```

[8]: (7750, 150, 150)

[9]:
```
x_train = x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 1)
y_test = np.array(y_test)
```

[13]:
```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the␣
 ↪dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range (degrees, 0␣
 ↪to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction␣
 ↪of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction␣
 ↪of total height)
        horizontal_flip = False,  # randomly flip images
        vertical_flip=False)  # randomly flip images


datagen.fit(x_train)
```

[22]:
```
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation =␣
 ↪'relu' , input_shape = (150,150,1)))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation =␣
 ↪'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
```

```
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation =␣
↪'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation =␣
↪'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation =␣
↪'relu'))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Flatten())

model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = 'adam' , loss = 'binary_crossentropy' , metrics =␣
↪['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 150, 150, 32)      320
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 75, 75, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 38, 38, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 38, 38, 64)        36928
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 19, 19, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 19, 19, 128)       73856
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 10, 10, 256)       295168
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 5, 5, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6400)              0
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 128)               819328
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,244,225
Trainable params: 1,244,225
Non-trainable params: 0
_________________________________________________________________
```

[23]:
```
history = model.fit(datagen.flow(x_train,y_train, batch_size = 64) ,epochs = 50␣
↪, validation_data = datagen.flow(x_val, y_val))
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 122 steps, validate for 1 steps
Epoch 1/50
122/122 [==============================] - 14s 113ms/step - loss: 0.5422 -
accuracy: 0.7097 - val_loss: 0.5690 - val_accuracy: 0.6875
Epoch 2/50
122/122 [==============================] - 13s 104ms/step - loss: 0.3128 -
accuracy: 0.8779 - val_loss: 0.8320 - val_accuracy: 0.6875
Epoch 3/50
122/122 [==============================] - 13s 105ms/step - loss: 0.2300 -
accuracy: 0.9150 - val_loss: 0.8922 - val_accuracy: 0.6250
Epoch 4/50
122/122 [==============================] - 13s 106ms/step - loss: 0.2017 -
accuracy: 0.9250 - val_loss: 0.7951 - val_accuracy: 0.8125
Epoch 5/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1884 -
accuracy: 0.9308 - val_loss: 1.0835 - val_accuracy: 0.5625
Epoch 6/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1689 -
accuracy: 0.9410 - val_loss: 1.0698 - val_accuracy: 0.6250
Epoch 7/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1546 -
accuracy: 0.9454 - val_loss: 0.8340 - val_accuracy: 0.6875
Epoch 8/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1350 -
accuracy: 0.9506 - val_loss: 0.5024 - val_accuracy: 0.8125
```

```
Epoch 9/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1397 -
accuracy: 0.9498 - val_loss: 0.7865 - val_accuracy: 0.6250
Epoch 10/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1373 -
accuracy: 0.9526 - val_loss: 0.5283 - val_accuracy: 0.8125
Epoch 11/50
122/122 [==============================] - 13s 107ms/step - loss: 0.1342 -
accuracy: 0.9489 - val_loss: 0.6110 - val_accuracy: 0.6875
Epoch 12/50
122/122 [==============================] - 13s 105ms/step - loss: 0.1137 -
accuracy: 0.9586 - val_loss: 0.7464 - val_accuracy: 0.7500
Epoch 13/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1122 -
accuracy: 0.9599 - val_loss: 0.3248 - val_accuracy: 0.8750
Epoch 14/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1088 -
accuracy: 0.9609 - val_loss: 0.4320 - val_accuracy: 0.8125
Epoch 15/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1165 -
accuracy: 0.9578 - val_loss: 0.4174 - val_accuracy: 0.8125
Epoch 16/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1081 -
accuracy: 0.9585 - val_loss: 0.1841 - val_accuracy: 0.8750
Epoch 17/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0973 -
accuracy: 0.9653 - val_loss: 0.8082 - val_accuracy: 0.7500
Epoch 18/50
122/122 [==============================] - 13s 106ms/step - loss: 0.1018 -
accuracy: 0.9632 - val_loss: 0.5360 - val_accuracy: 0.7500
Epoch 19/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0874 -
accuracy: 0.9671 - val_loss: 0.8666 - val_accuracy: 0.6250
Epoch 20/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0946 -
accuracy: 0.9663 - val_loss: 0.5295 - val_accuracy: 0.8750
Epoch 21/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0866 -
accuracy: 0.9698 - val_loss: 0.5800 - val_accuracy: 0.8125
Epoch 22/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0858 -
accuracy: 0.9712 - val_loss: 0.3279 - val_accuracy: 0.8125
Epoch 23/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0840 -
accuracy: 0.9726 - val_loss: 0.2403 - val_accuracy: 0.8750
Epoch 24/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0823 -
accuracy: 0.9708 - val_loss: 0.1125 - val_accuracy: 0.9375
```

```
Epoch 25/50
122/122 [==============================] - 13s 105ms/step - loss: 0.0803 -
accuracy: 0.9732 - val_loss: 0.3287 - val_accuracy: 0.8125
Epoch 26/50
122/122 [==============================] - 13s 105ms/step - loss: 0.0809 -
accuracy: 0.9724 - val_loss: 0.6152 - val_accuracy: 0.7500
Epoch 27/50
122/122 [==============================] - 13s 105ms/step - loss: 0.0809 -
accuracy: 0.9706 - val_loss: 0.6779 - val_accuracy: 0.6875
Epoch 28/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0869 -
accuracy: 0.9698 - val_loss: 0.5346 - val_accuracy: 0.6250
Epoch 29/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0813 -
accuracy: 0.9723 - val_loss: 0.3399 - val_accuracy: 0.7500
Epoch 30/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0709 -
accuracy: 0.9754 - val_loss: 0.4146 - val_accuracy: 0.7500
Epoch 31/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0841 -
accuracy: 0.9706 - val_loss: 0.3292 - val_accuracy: 0.7500
Epoch 32/50
122/122 [==============================] - 13s 105ms/step - loss: 0.0742 -
accuracy: 0.9730 - val_loss: 0.4310 - val_accuracy: 0.8125
Epoch 33/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0709 -
accuracy: 0.9760 - val_loss: 0.4935 - val_accuracy: 0.7500
Epoch 34/50
122/122 [==============================] - 13s 108ms/step - loss: 0.0758 -
accuracy: 0.9732 - val_loss: 0.3261 - val_accuracy: 0.8750
Epoch 35/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0697 -
accuracy: 0.9741 - val_loss: 0.2066 - val_accuracy: 0.9375
Epoch 36/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0691 -
accuracy: 0.9756 - val_loss: 0.6529 - val_accuracy: 0.8125
Epoch 37/50
122/122 [==============================] - 13s 105ms/step - loss: 0.0680 -
accuracy: 0.9760 - val_loss: 0.2156 - val_accuracy: 0.9375
Epoch 38/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0633 -
accuracy: 0.9777 - val_loss: 0.8503 - val_accuracy: 0.7500
Epoch 39/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0618 -
accuracy: 0.9778 - val_loss: 0.0871 - val_accuracy: 1.0000
Epoch 40/50
122/122 [==============================] - 13s 108ms/step - loss: 0.0615 -
accuracy: 0.9757 - val_loss: 0.3484 - val_accuracy: 0.7500
```

```
Epoch 41/50
122/122 [==============================] - 13s 108ms/step - loss: 0.0610 -
accuracy: 0.9785 - val_loss: 0.6102 - val_accuracy: 0.8125
Epoch 42/50
122/122 [==============================] - 13s 108ms/step - loss: 0.0770 -
accuracy: 0.9725 - val_loss: 0.0790 - val_accuracy: 1.0000
Epoch 43/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0619 -
accuracy: 0.9794 - val_loss: 0.6408 - val_accuracy: 0.7500
Epoch 44/50
122/122 [==============================] - 13s 108ms/step - loss: 0.0609 -
accuracy: 0.9777 - val_loss: 0.0591 - val_accuracy: 1.0000
Epoch 45/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0588 -
accuracy: 0.9792 - val_loss: 0.6212 - val_accuracy: 0.8125
Epoch 46/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0668 -
accuracy: 0.9761 - val_loss: 0.3373 - val_accuracy: 0.8750
Epoch 47/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0565 -
accuracy: 0.9803 - val_loss: 0.5278 - val_accuracy: 0.8125
Epoch 48/50
122/122 [==============================] - 13s 107ms/step - loss: 0.0556 -
accuracy: 0.9794 - val_loss: 0.2250 - val_accuracy: 0.8750
Epoch 49/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0583 -
accuracy: 0.9779 - val_loss: 0.1534 - val_accuracy: 0.9375
Epoch 50/50
122/122 [==============================] - 13s 106ms/step - loss: 0.0573 -
accuracy: 0.9795 - val_loss: 0.1490 - val_accuracy: 1.0000
```

[24]:

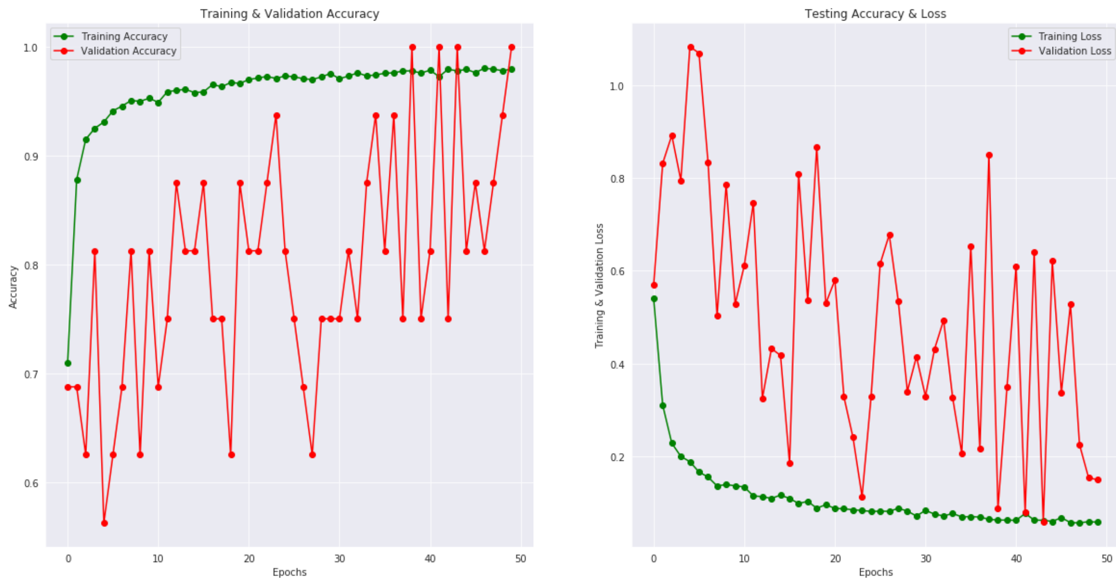
```
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0]*100 , "%")
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 ,␣
 ↪"%")
```

```
624/624 [==============================] - 1s 840us/sample - loss: 0.3202 -
accuracy: 0.9215
Loss of the model is -  32.02381273492789 %
624/624 [==============================] - 0s 655us/sample - loss: 0.3202 -
accuracy: 0.9215
Accuracy of the model is -  92.14743375778198 %
```

[39]:

```
epochs = [i for i in range(50)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
```

```
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()
```



[27]:
```
predictions = model.predict_classes(x_test)
predictions = predictions.reshape(1,-1)[0]
predictions[:15]
```

[27]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)

[28]:
```
print(classification_report(y_test, predictions, target_names = ['Pneumonia␣
↪(Class 0)','Normal (Class 1)']))
```

```
              precision    recall  f1-score   support
```

```
Pneumonia (Class 0)       0.94      0.94      0.94       390
   Normal (Class 1)       0.90      0.89      0.90       234

           accuracy                           0.92       624
          macro avg       0.92      0.92      0.92       624
       weighted avg       0.92      0.92      0.92       624
```

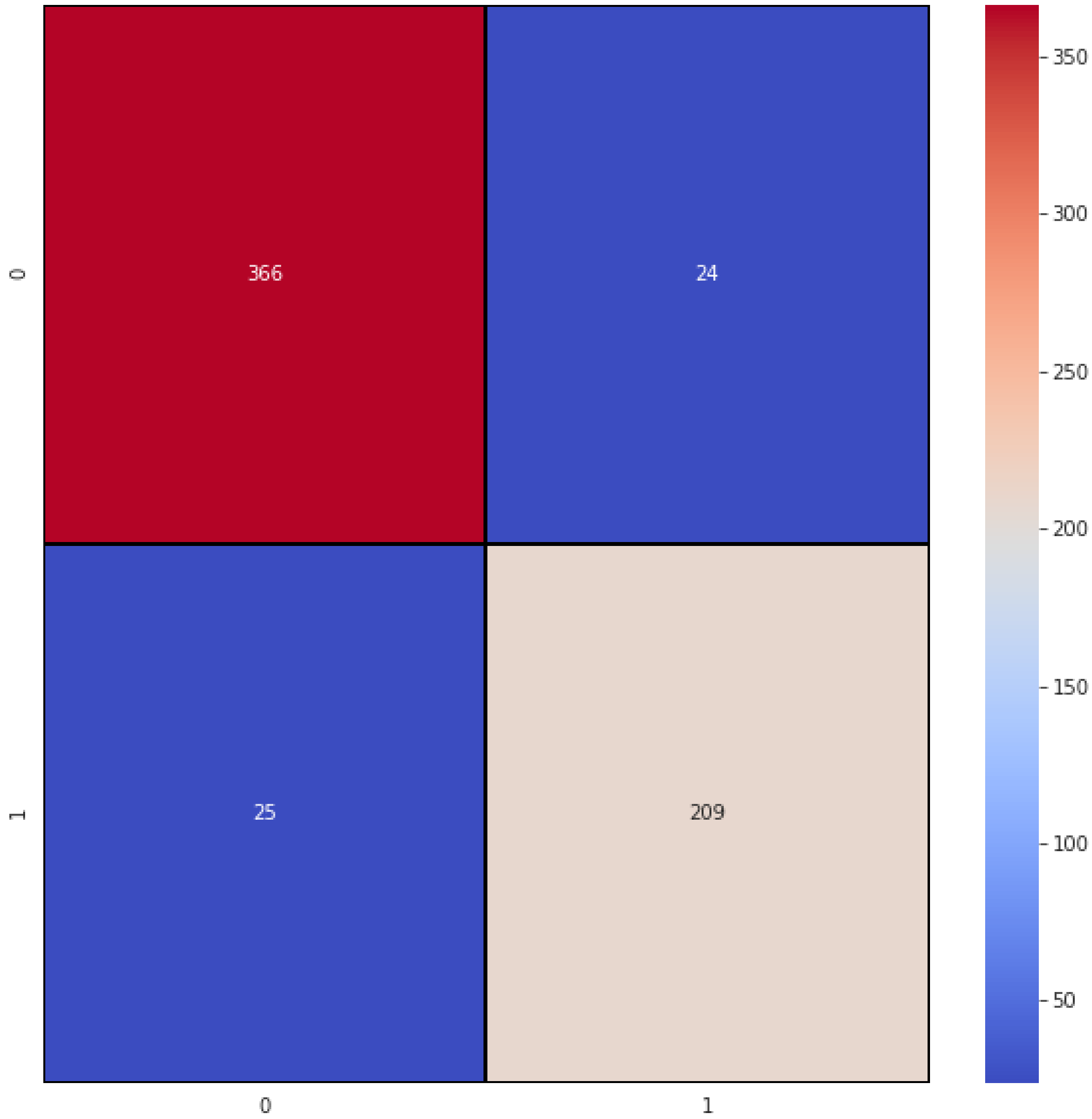
[29]:
```python
cm = confusion_matrix(y_test,predictions)
cm
```

[29]:
```
array([[366,  24],
       [ 25, 209]])
```

[30]:
```python
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
```

[31]:
```python
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "coolwarm", linecolor = 'black' , linewidth = 1 , annot =␣
 ↪True, fmt='')
```

[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8680092b50>
```

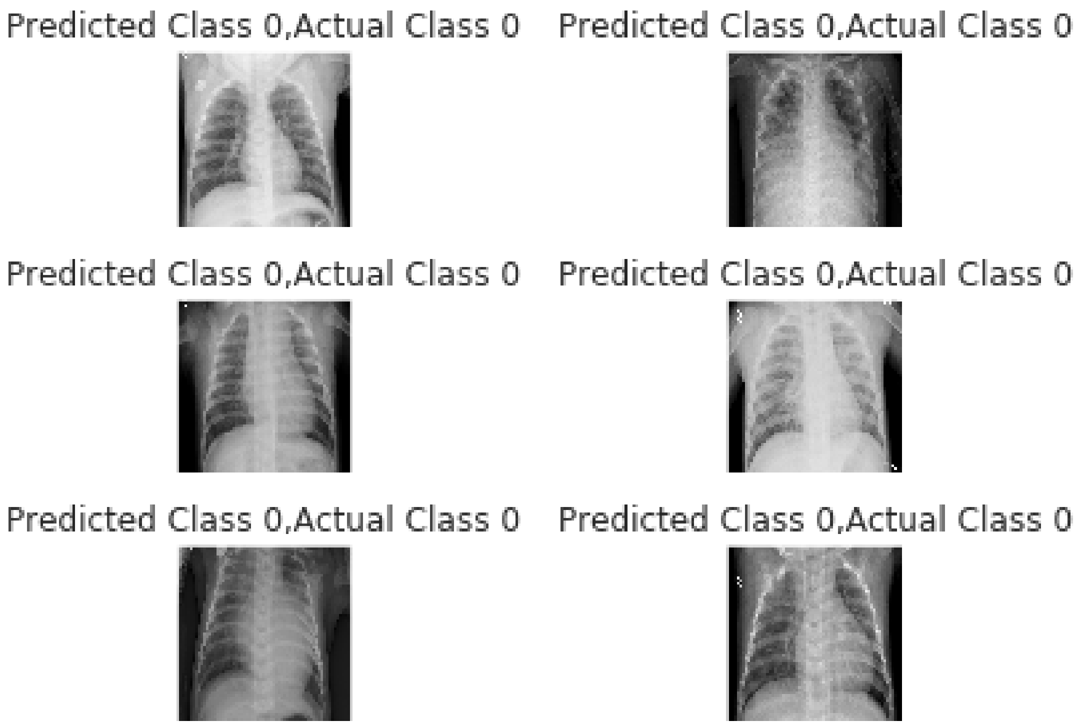


[32]:
```
correct = np.nonzero(predictions == y_test)[0]
incorrect = np.nonzero(predictions != y_test)[0]
print(len(correct))
print(len(incorrect))
```

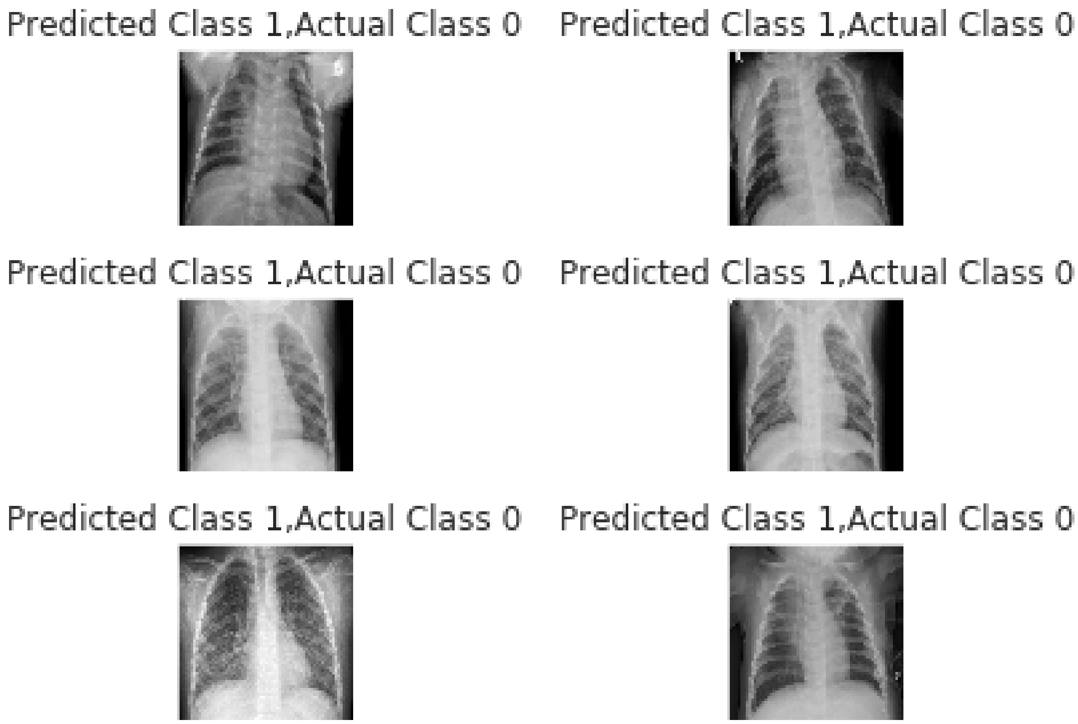
575
49

[33]:
```
i = 0
for c in correct[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
```

```
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150), cmap="gray", interpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(predictions[c],␣
↪y_test[c]))
    plt.tight_layout()
    i += 1
```



[34]:
```
i = 0
for c in incorrect[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150), cmap="gray", interpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(predictions[c],␣
↪y_test[c]))
    plt.tight_layout()
    i += 1
```


Predicted Class 1,Actual Class 0
Predicted Class 1,Actual Class 0
Predicted Class 1,Actual Class 0
Predicted Class 1,Actual Class 0
Predicted Class 1,Actual Class 0
Predicted Class 1,Actual Class 0

[ ]:

[ ]:

REFERENCES

[1] Q. Wang, D. Yang, Z. Li, X. Zhang and C. Liu, "Deep Regression via Multi-Channel Multi-Modal Learning for Pneumonia Screening," in IEEE Access,2020.

[2] D. Odaibo, Z. Zhang, F. Skidmore and M. Tanik, "Detection of Visual Signals for Pneumonia in Chest Radiographs using Weak Supervision," 2019 SoutheastCon, Huntsville, AL, USA, 2019.

[3] K. Jakhar and N. Hooda, "Big Data Deep Learning Framework using Keras: A Case Study of Pneumonia Prediction," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018.

[4] Kalyani Kadam, Dr.Swati Ahirrao,Harbir Kaur ,Dr. Shraddha Phansalkar , Dr. Ambika Pawar, "Deep Learning Approach For Prediction of Pneumonia," International Journal of Scientific & Technology Research Volume 8, issue 10, October 2019.

[5] Paras Lakhani and Baskaran Sundaram, "Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks," *Radiology 2017 284:2, 574-582.*

[6] Arcadu, F., Benmansour, F., Maunz, A, "Deep learning algorithm predicts diabetic retinopathy progression in individual patients," *npj Digit. Med.* **2,** 92 (2019)

[7] Jia Ding, Aoxue Li, Zhiqiang Hu, Liwei Wang, "Accurate Pulmonary Nodule Detection in Computed Tomography Images Using Deep Convolutional Neural Networks," Medical Image Computing and Computer Assisted Intervention − MICCAI 2017.

[8] Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen and H. Greenspan, "Chest pathology detection using deep learning with non-medical training," *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, New York, NY, 2015, pp. 294-297.

[9] Keiron O`Shea, Ryan Nash "An Introduction to Convolutional Neural Networks," Cornell University, Dec 2015.

[10] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9,** 611–629 ,2018.

[11] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng and M. Chen, "Medical image classification with convolutional neural network," *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, Singapore, 2014, pp. 844-848, doi: 10.1109/ICARCV.2014.7064414.

[12] Y. Zhang and H. Yu, "Convolutional Neural Network Based Metal Artifact Reduction in X-Ray Computed Tomography," in IEEE Transactions on Medical Imaging, vol. 37, no. 6, pp. 1370-1381, June 2018, doi: 10.1109/TMI.2018.2823083.

[13] Pranav Rajpurkar and Jeremy Irvin and Kaylie Zhu and Brandon Yang and Hershel Mehta and Tony Duan and Daisy Ding and Aarti Bagul and Curtis Langlotz and Katie Shpanskaya and Matthew P. Lungren and Andrew Y. Ng, "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning," arXiv:1711.05225v3 ,Cornell University, Dec 2017

[14] Rajaraman, S.; Candemir, S.; Kim, I.; Thoma, G.; Antani, S. "Visualization and Interpretation of Convolutional Neural Network Predictions in Detecting Pneumonia in Pediatric Chest Radiographs". *Appl. Sci.* **2018**, *8*, 1715.

[15] Lin Li, Lixin Qin, Zeguo Xu, Youbing Yin, Xin Wang, Bin Kong, Junjie Bai, Yi Lu, Zhenghan Fang, Qi Song, Kunlin Cao, Daliang Liu, Guisheng Wang, Qizhong Xu, Xisheng Fang, Shiqin Zhang, Juan Xia, and Jun Xia " Artificial Intelligence Distinguishes COVID-19 from Community Acquired Pneumonia on Chest CT," RSNA Journals, March 2020.

[16] Ilyas Sirazitdinov, Maksym Kholiavchenko, Tamerlan Mustafaev, Yuan Yixuan, Ramil Kuleev,Bulat Ibragimov, "Deep neural network ensemble for pneumonia localization from a large-scale chest x-ray database," Computers & Electrical Engineering, Volume 78,Pages 388-399, ISSN0045-7906, 2019.

[17] Okeke Stephen, **Mangal Sain**, Uchenna Joseph Maduh, and **Do-Un Jeong,** "An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare," Department of Computer Engineering, Dongseo University, Busan, Republic of Korea, March 2019.

[18] Arata Andrade Saraiva, Nuno M. Fonseca Ferreira, Luciano Lopes de Sousa, Nator Junior C. Costa, José Vigno M. Sousa, D. B. S. Santos, António Valente, Salviano F. S. P. Soares,"Classification of Images of Childhood Pneumonia using Convolutional Neural Networks," BIOIMAGING 2019.

[19] Zech, John R et al. "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study." *PLoS medicine* vol. 15,11 e1002683. 6 Nov. 2018.

[20] Rastegari M., Ordonez V., Redmon J., Farhadi A. (2016) XNOR-Net, "ImageNet Classification Using Binary Convolutional Neural Networks," Computer Vision – ECCV 2016.

[21] Marcelo Fiszman, MD, Wendy W. Chapman, Dominik Aronsky, MD, R. Scott Evans, PhD, Peter J. Haug, MD, "Automatic Detection of Acute Bacterial Pneumonia from Chest X-ray Reports," Journal of the American Medical Informatics Association, Volume 7, Issue 6, November 2000.

[22] Gregory F. Cooper, Constantin F. Aliferis, Richard Ambrosino, John Aronis, Bruce G. Buchanan, Richard Caruana, Michael J. Fine, Clark Glymour, Geoffrey Gordon, Barbara H. Hanusa, Janine E. Janosky,

Christopher Meek, Tom Mitchell, Thomas Richardson, Peter Spirtes, "An evaluation of machine-learning methods for predicting pneumonia mortality," Artificial Intelligence in Medicine,Volume 9, Issue 2,Pages 107-138, ISSN 0933-3657, 1997.

[23] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission," *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD '15*). Association for Computing Machinery, New York, NY, USA, 1721–1730, 2015.

[24] Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C.S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K. Prasadha, Jacqueline Pei, Magdalene Y.L. Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A.N. Huu, Cindy Wen, Edward D. Zhang, Charlotte L. Zhang, Oulan Li, Xiaobo Wang, Michael A. Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M. Anthony Lewis, Huimin Xia, Kang Zhang, "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," Volume 172, Issue 5,Pages 1122-1131.e9,ISSN 0092-8674, 2018.