



**NEW HORIZON  
COLLEGE OF ENGINEERING**



# **“PNEUMONIA DETECTION USING DEEP LEARNING”**

## **A MINI PROJECT REPORT**

*Submitted by*

**DHANUSH BILIGIRI N H[1NH18IS030]**

*Under the guidance of,*

**Dr. K. Saravanan**

**Professor, ISE, NHCE**

*In partial fulfilment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION SCIENCE AND ENGINEERING**

**FOR**

**COURSE NAME: MINI PROJECT**

**COURSE CODE: 20ISE68**



## **CERTIFICATE**

Certified that the project work entitled “PNEUMONIA DETECTION USING DEEP LEARNING” carried out by Mr. DHANUSH BILIGIRI N H, bearing USN 1NH18IS030, a bonafide student of 6<sup>th</sup> semester in partial fulfilment for the award of Bachelor of Engineering in Information Science & Engineering of the Visveswaraiah Technological University, Belagavi during the year 2020-21. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated. The project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree.

**Name & Signature of Guide**

Dr. K.Saravanan

**Name & Signature of HOD**

Dr. Anandhii R J

**Name & Signature of Principal**

Dr. Manjunatha

### **Examiners:**

**Name**

**Signature**

1. ....

.....

2. ....

.....

---

## ORIGINALITY REPORT

---

26%

SIMILARITY INDEX

19%

INTERNET SOURCES

14%

PUBLICATIONS

20%

STUDENT PAPERS

---

## PRIMARY SOURCES

---

1

[en.wikipedia.org](https://en.wikipedia.org)

Internet Source

4%

2

[www.researchsquare.com](https://www.researchsquare.com)

Internet Source

4%

3

[www.theseus.fi](https://www.theseus.fi)

Internet Source

3%

4

[www.hindawi.com](https://www.hindawi.com)

Internet Source

2%

5

Submitted to Angeles University Foundation

Student Paper

2%

6

Submitted to Universidad Del Magdalena

Student Paper

2%

7

Submitted to Nanyang Polytechnic

Student Paper

2%

8

[python.mrkishorekumar.com](https://python.mrkishorekumar.com)

Internet Source

2%

9

Submitted to British University in Egypt

Student Paper

1%

---

# ABSTRACT

Deep learning is an artificial intelligence(AI) function that imitates the working of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in AI that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network.

Pneumonia is a form of acute respiratory infection that affects the lungs. The lungs are made up of small sacs called alveoli, which fill with air when a healthy person breathes. When an individual has pneumonia, the alveoli are filled with pus and fluid, which makes breathing painful and limits oxygen intake.

Over 150 million people get infected by pneumonia on an annual basis especially children under the age of 5 years. Pneumonia is the world's deadliest child killer claiming one young life every 39 seconds. The disease is preventable but still has a major effect than any other infection. This disease become more deadly as the germs that cause it is contagious.

The risk of pneumonia is immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. Due to the current rise of the COVID-19 pandemic, it is observed that people who are pneumonic are more prone to the virus than others.

This project aims at collecting and pre-processing the X-ray images from a dataset and creating a model for training and validating the dataset. The goal is to separate the positive and negative pneumonia reports and be extremely accurate with the results. This project is implemented on the anaconda platform.

## ACKNOWLEDGEMENT

Any project is a task of great enormity and it cannot be accomplished by an individual without support and guidance. I am grateful to a number of individuals whose professional guidance and encouragement has made this project completion a reality.

I have a great pleasure in expressing my deep sense of gratitude to the beloved Chairman **Dr. Mohan Manghnani** for having provided me with a great infrastructure and well-furnished labs.

I take this opportunity to express my profound gratitude to the Principal **Dr. Manjunatha** for his constant support and management.

I am grateful to **Dr.R J Anandhi**, Professor and Head of Department of ISE, New Horizon College of Engineering, Bengaluru for his strong enforcement on perfection and quality during the course of my project work.

I would like to express my thanks to the guide **Dr K. Saravanan**, Professor, Department of ISE, New Horizon College of Engineering, Bengaluru who has always guided me in detailed technical aspects throughout my project.

I would like to mention special thanks to all the Teaching and Non-Teaching staff members of Information Science and Engineering Department, New Horizon College of Engineering, Bengaluru for their invaluable support and guidance.

**DHANUSH BILIGIRI N H**

**1NH18IS030**

# TABLE OF CONTENTS

<b>CHAPTER 1.....</b>	<b>1</b>
Introduction.....	1
Motivation of project.....	2
 <b>CHAPTER 2.....</b>	 <b>3</b>
Python.....	3
Numpy.....	4
OpenCV.....	5
Keras.....	6
Matplotlib.....	7
 <b>CHAPTER 3.....</b>	 <b>9</b>
System Requirements .....	9
About the language .....	9
 <b>CHAPTER 4.....</b>	 <b>10</b>
Design Modules .....	10
 <b>CHAPTER 5.....</b>	 <b>10</b>
Code and Implementations .....	10
 <b>CHAPTER 6.....</b>	 <b>15</b>
Results and Discussions.....	15
Conclusion.....	20
 <b>BIBLIOGRAPHY .....</b>	 <b>21</b>

# LIST OF FIGURES

Figure 2.1: Python.....	3
Figure 2.2: Python Example .....	4
Figure 2.3: Numpy .....	5
Figure 2.4: OpenCV .....	5
Figure 2.5: Keras.....	6
Figure 2.6: Matplotlib .....	7
Figure 2.7: Matplotlib .....	7
Figure 4.1: Architecture .....	10
Chapter 6: Output Screenshots .....	15-20

## CHAPTER 1

# INTRODUCTION

Deep learning is an artificial intelligence(AI) function that imitates the working of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in AI that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network

Pneumonia is a lung illness caused by an acute respiratory infection. When a healthy person breathes, tiny sacs called alveoli in the lungs fill up with air. When a person develops pneumonia, the alveoli get clogged with pus and fluid, making breathing difficult and restricting oxygen intake. The risk of pneumonia is immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. Due to the current rise of the COVID-19 pandemic, it is observed that people who are pneumonic are more prone to the virus than others.

With the recent epidemic of COVID-19, commonly known as the coronavirus, it appears like history is repeating itself and we are returning to the Spanish influenza pandemic of the 1900s. The coronavirus is a lethal virus that has claimed the lives of hundreds of thousands of people all over the world. Adults over the age of 65, and for those with major underlying medical issues or a history of pneumonia, appear to be more susceptible to the virus's more catastrophic symptoms. Doctors and medical professionals all around the world are working around the clock to cure patients and prevent the virus from spreading due to mounting mortality and limited medical resources. Severe strains of the virus can cause pneumonia, which can lead to death. It is critical to have rapid and accurate pneumonia detection so that patients can obtain treatment as soon as possible, especially in impoverished areas.

With today's technology breakthroughs, technologies based on deep learning frameworks can be used to identify pneumonia based on chest x-ray pictures. The task at hand would be to assist in the diagnosing process, allowing for faster treatment and improved clinical outcomes.

## 1.1 The Motivation for the Project

Pneumonia is a frequent disease caused by bacteria, viruses, and fungus, among other microorganisms. The word "pneumonia" is derived from the Greek word "pneumon," which means "lung." As a result, the word pneumonia is linked to lung disease. Pneumonia is a condition that causes inflammation of the lung parenchyma in one or both lungs.

Food aspiration and chemical exposure, on the other hand, are two other causes of pneumonia. Pneumonia is caused by microorganisms infecting the lungs, causing the alveoli in the lungs to fill up with fluid or pus, reducing carbon dioxide (CO<sub>2</sub>) and oxygen (O<sub>2</sub>) exchange between the blood and the lungs, making it difficult for afflicted people to breathe. Shortness of breath, fever, cough, chest pain, and other symptoms of pneumonia can occur. Furthermore, patients with HIV/AIDS, diabetes, chronic respiratory diseases, cardiovascular diseases, cancer, hepatic disease, and other comorbidities, such as HIV/AIDS, diabetes, chronic respiratory diseases, cardiovascular diseases, cancer, and hepatic disease, are at risk of pneumonia.

COVID-19 is transferred through infected people's inhaled or produced respiratory droplets. Coronaviruses infect the alveoli (an organ responsible for the exchange of oxygen and carbon dioxide in the lungs), resulting in pneumonia. Dry cough, exhaustion, fever, septic shock, organ failure, anorexia, dyspnea, myalgias, sputum production, severe pneumonia, acute respiratory distress syndrome (ARDS), and other symptoms are all signs of COVID-19. The pandemic caused by SAR-CoV-2 is concerning because no approved treatment or vaccination exists.

Over 150 million people get infected by pneumonia on an annual basis especially children under the age of 5 years. Pneumonia is the world's deadliest child killer claiming one young life every 39 seconds. So it is mandatory to come up with an algorithm that will help in detecting Pneumonia.

Ensuring that the algorithm works accurately will be our utmost concern



## CHAPTER 2

## Python, OpenCV, Deep Learning

This project uses python as the programming language and uses Tkinter to provide a graphical user interface. Machine learning techniques are used to predict the price in the stock market.

### 2.1 PYTHON

Python is a scripting programming language that considered to be the best for beginners because of its simplicity and wide range of applications. Python is also referred as “general purpose” language by the community of programmers. It provides a wide range of data types, compound data structures such as lists,



**Figure 2.1: Python**

tuples and dictionaries. External libraries can also be used to achieve the output.

#### **Feature of Python is as follows:**

- It is simple to use.
- It is a free and open source programming language.
- It has a large library.
- Through python we can achieve object-oriented programming.
- Python has readability and elegant syntax.
- Python is a cross-platform.
- An interpreter is used to interpret the code line by line.
- Other programming languages such as C, C++, Java can be integrated with Python.
- It has a rich collection of Python data types.
- Python provides better run-time flexibility and a platform for GUI.

- Python provides conditionals, loops, functions.

A basic program in Python:

```
# This program prints Hello, world!  
  
print('Hello, world!')
```

Here, the in-built print() function is used to print the data on the output screen.

Output:

```
Hello, world!
```

**Figure 2.2: Python Example**

### **Numpy:**

NumPy is a Python module that allows you to work with arrays. It also provides functions for working with matrices, fourier transforms, and linear algebra. Travis Oliphant invented NumPy in 2005. It is an open source project that is free to use. Numerical Python is the abbreviation for NumPy. We have lists in Python that act as arrays, however they are slow to process. NumPy intends to deliver a 50-x quicker array object than ordinary Python lists. The array object in NumPy is named ndarray, and it comes with several helper functions to make working with it a simple. In data research, when speed and resources are critical, arrays are widely employed. NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

Numpys are faster than arrays and the reason for that is because in the memory location, the numpy arrays are stored in a contiguous place unlike the lists. Therefore, it can be accessed and manipulated very efficiently by the processes.

Installation of mumpy becomes easy if there is Python and PIP already present in the system. Once installed, then numpy can be imported into the program.

An example is as follows:

```
import numpy
arr = numpy.array([1, 2, 3, 4, 5])
print(arr)
```

[1 2 3 4 5]

**Figure 2.3: Numpy**

An alias name is used while importing numpy as it can be used later on in the code instead of using the entire word numpy. i.e. import numpy as np.

### OpenCV:

The Open Source Computer Vision Library is a collection of programming functions geared mostly toward real-time computer vision. It was created by Intel and then sponsored by Willow Garage and Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations. The important features of tensorflow are:

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

```
import cv2,time
first_frame = None
video = cv2.VideoCapture(0)
while True:
    check, frame = video.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray,(21,21),0)
    if first_frame is None:
        first_frame = gray
        continue
```

Create a VideoCapture object to record video using web cam

Convert the frame color to gray scale

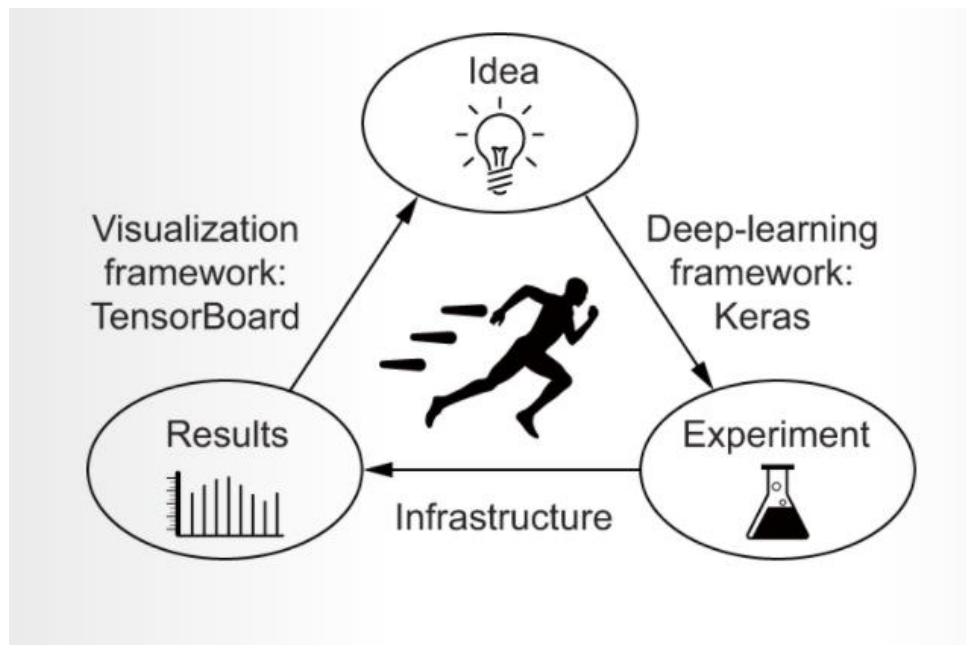
Convert the gray scale frame to GaussianBlur

This is used to store the first image/frame of the video

**Figure 2.4: OpenCV**

**Keras:**

Keras is another open source Python library that is free and easy to use which is majourly used for developing and for evaluating the deep learning models. The efficient numerical computational libraries such as the Theano and tensorflow is wrapped in the keras and allows to define and train the models of the neural networks in just a few lines of code



**Figure 2.5: Keras**

It is built to be modular, quick, and simple to use. It was created by Google developer François Chollet. Low-level computation is not handled by Keras. Instead, it makes use of a library known as the "Backend" to do it.

Some of the advantages of using keras is:

- Easy to understand and fast deployment: if you want to make a network model keras is very quick. It has a friendly API with the help of which we can easily understand the process. Code can be written with simple function and no need to have multiple parameters.
- Large community help: a lot of AI communities make use of keras for their deep learning and publish their codes.

- Multiple backend can be used: one can use tensorflow, Theano, CNTK as the backend for keras. Different backends can be used for different projects depending on the needs. Every backend has its own unique advantage.
- Easy model deployment and cross-platform: keras can be deployed on any device.

## MATPLOTLIB

Matplotlib library helps create visualization with Python.

Matplotlib is a library in Python that allows you to create static, animated, and interactive visualisations. It helps in making the thing easy even easy and hard things possible to do.

- Create: With just a few lines of code, you can create publication-quality graphs. It makes use of interactive figures that can zoom, update, pan etc.
- Customize: the user can have full control of the line styles, axes properties, font properties etc. Export to and embed in a variety of file types and interactive contexts.
- Extend: user can explore the tailored functionality which will be provided by the third party packages. Many external learning resources also help in learning more about the matplotlib library.

A lot of the matplotlib utilities fall under the pyplot submodule, and is usually imported under the alias name plt.

```
import matplotlib.pyplot as plt
```

**Figure 2.6: Matplotlib**

Plotting of graphs using matplotlib:

Plot()- this function is used to draw a points in the graph. By default, the line is drawn from point to point. Parameters are taken by the function to specify the points in the diagram.

The xlabel() and ylabel() functions can be used to set labels for the x-axis and the y-axis.

Parameter 1: contains the points on the x-axis.

Parameter 2: contains the points on the y-axis.

An example for plotting a graph using matplotlib is as follows:

```
#Three lines to make our compiler able to
draw:
import sys
import matplotlib
matplotlib.use('Agg')

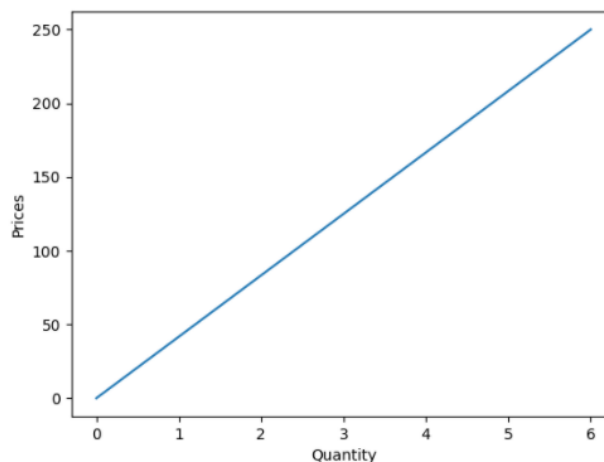
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.xlabel("Quantity")
plt.ylabel("Prices")
plt.plot(xpoints, ypoints)
plt.show()

#Two lines to make our compiler able to
draw:
plt.savefig(sys.stdout.buffer)
sys.stdout.flush()
```

**Figure 2.7: Matplotlib**

The output will be as follows:



**Figure 2.8: Output for Matplotlib**

## CHAPTER 3

### SYSTEM REQUIREMENTS

The project has been developed utilizing Python, Tkinter, Machine learning. The requirements in order to develop this project and system design and flow is described in this chapter.

#### 3.1. Project System Requirements

##### ➤ **Software Requirements:**

- Front end: Python 3.7 or higher, PyCharm, spyder or any other IDE.

##### ➤ **Hardware Requirements:**

- Operating system: Windows 7 or above.
- Speed: 1 GHz
- Processor i3, i5 7th Gen or above
- Hard disk: 1TB
- Ram: 8GB

#### 3.2 ABOUT THE LANGUAGE

Spam classifier using Naïve Bayes algorithm is implemented using python on anaconda spyder terminal. Most of the machine learning algorithms are implemented using python.

Python has various features such as:

- 1) it is considered to be an expressive language which means it is more understandable and readable.
- 2) it is free and is an open source – python can be downloaded for free from the website and is available with the source code which makes it an open source.
- 3) multiple libraries – Python comes with many built in packages which makes it easier for the users to build applications rapidly.
- 4) Cross platform language – python can run on different platforms such as windows, linux, unix etc.

## CHAPTER 4

## DESIGN MODULES

The project has been implemented with some functions. The discussion of the flow of the project has been described in this chapter.

## 4.1 ARCHITECTURE

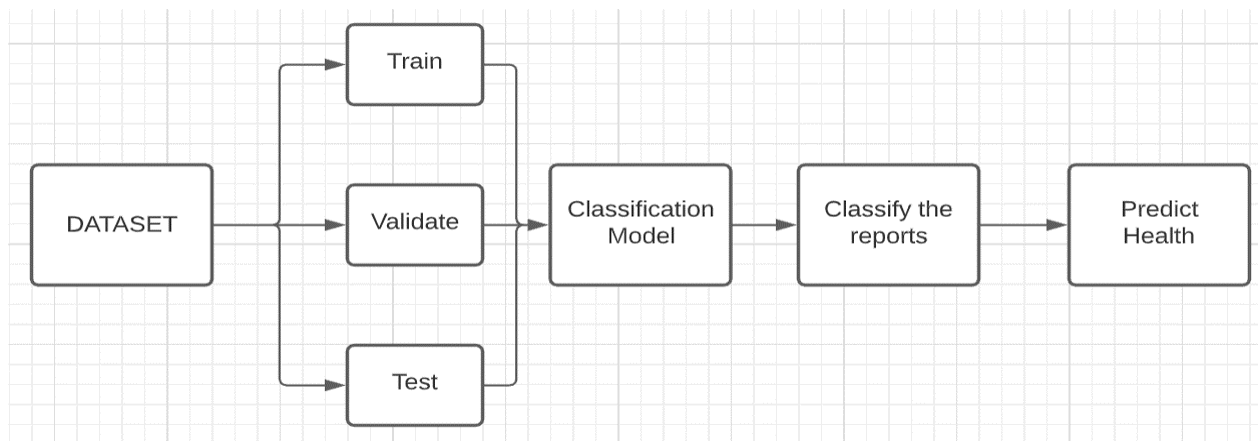


Figure 4.1: Architecture

## CHAPTER 5

## CODE AND IMPLEMENTATION

The code for the project has been discussed in this chapter.

```

# General libraries
import os
import numpy as np
import pandas as pd
import random
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

# Deep learning libraries
import keras.backend as K
from keras.models import Model, Sequential
from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

```



```

import tensorflow as tf

# Setting seeds for reproducibility
seed = 232
np.random.seed(seed)
tf.random.set_seed(seed)

# input_path = '../input/chest_xray/chest_xray/'
input_path = "D:\\Dataset\\"

fig, ax = plt.subplots(2, 3, figsize=(15, 7))
ax = ax.ravel()
plt.tight_layout()

for i, _set in enumerate(['train', 'val', 'test']):
    set_path = input_path+_set
    ax[i].imshow(plt.imread(set_path+"\\NORMAL\\"+os.listdir(set_path+"\\NORMAL")[0]),
    cmap='gray')
    ax[i].set_title('Set: {}, Condition: Normal'.format(_set))
    ax[i+3].imshow(plt.imread(set_path+'/PNEUMONIA/'+os.listdir(set_path+'/PNEUMONIA')[0]),
    cmap='gray')
    ax[i+3].set_title('Set: {}, Condition: Pneumonia'.format(_set))
# Distribution of our datasets
for _set in ['train', 'val', 'test']:
    n_normal = len(os.listdir(input_path + _set + '/NORMAL'))
    n_infect = len(os.listdir(input_path + _set + '/PNEUMONIA'))
    print('Set: {}, normal images: {}, pneumonia images: {}'.format(_set, n_normal, n_infect))

# input_path = '../input/chest_xray/chest_xray/'
input_path = 'D:\\Dataset\\'

def process_data(img_dims, batch_size):
    # Data generation objects
    train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.3, vertical_flip=True)
    test_val_datagen = ImageDataGenerator(rescale=1./255)

    # This is fed to the network in the specified batch sizes and image dimensions
    train_gen = train_datagen.flow_from_directory(
        directory=input_path+'train',
        target_size=(img_dims, img_dims),
        batch_size=batch_size,

```

```
class_mode='binary',
shuffle=True)

test_gen = test_val_datagen.flow_from_directory(
    directory=input_path+'test',
    target_size=(img_dims, img_dims),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True)

# I will be making predictions off of the test set in one batch size
# This is useful to be able to get the confusion matrix
test_data = []
test_labels = []

for cond in ['/NORMAL/', '/PNEUMONIA/']:
    for img in (os.listdir(input_path + 'test' + cond)):
        img = plt.imread(input_path+'test'+cond+img)
        img = cv2.resize(img, (img_dims, img_dims))
        img = np.dstack([img, img, img])
        img = img.astype('float32') / 255
        if cond=='/NORMAL/':
            label = 0
        elif cond=='/PNEUMONIA/':
            label = 1
        test_data.append(img)
        test_labels.append(label)

test_data = np.array(test_data)
test_labels = np.array(test_labels)

return train_gen, test_gen, test_data, test_labels

# Hyperparameters
img_dims = 150
epochs = 10
batch_size = 32
|
# Getting the data
train_gen, test_gen, test_data, test_labels = process_data(img_dims, batch_size)
```

```
# Input layer
inputs = Input(shape=(img_dims, img_dims, 3))

# First conv block
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Second conv block
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Third conv block
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Fourth conv block
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

# Fifth conv block
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

# FC layer
x = Flatten()(x)
x = Dense(units=512, activation='relu')(x)
x = Dropout(rate=0.7)(x)
x = Dense(units=128, activation='relu')(x)
x = Dropout(rate=0.5)(x)
x = Dense(units=64, activation='relu')(x)
```

```
x = Dropout(rate=0.3)(x)

# Output layer
output = Dense(units=1, activation='sigmoid')(x)

# Creating model and compiling
model = Model(inputs=inputs, outputs=output)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Callbacks
checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True,
save_weights_only=True)
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=2,
mode='max')
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')

# Fitting the model
hist = model.fit(
    train_gen, steps_per_epoch=train_gen.samples // batch_size,
    epochs=epochs, validation_data=test_gen,
    validation_steps=test_gen.samples // batch_size, callbacks=[checkpoint, lr_reduce])

fig, ax = plt.subplots(1, 2, figsize=(10, 3))
ax = ax.ravel()

for i, met in enumerate(['accuracy', 'loss']):
    ax[i].plot(hist.history[met])
    ax[i].plot(hist.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])

from sklearn.metrics import accuracy_score, confusion_matrix

preds = model.predict(test_data)

acc = accuracy_score(test_labels, np.round(preds))*100
cm = confusion_matrix(test_labels, np.round(preds))
tn, fp, fn, tp = cm.ravel()
```

```

print('CONFUSION MATRIX -----')
print(cm)

print('\nTEST METRICS -----')
precision = tp/(tp+fp)*100
recall = tp/(tp+fn)*100
print('Accuracy: {}'.format(acc))
print('Precision: {}'.format(precision))
print('Recall: {}'.format(recall))
print('F1-score: {}'.format(2*precision*recall/(precision+recall)))

print('\nTRAIN METRIC -----')
print('Train acc: {}'.format(np.round((hist.history['accuracy'][-1])*100, 2)))

```

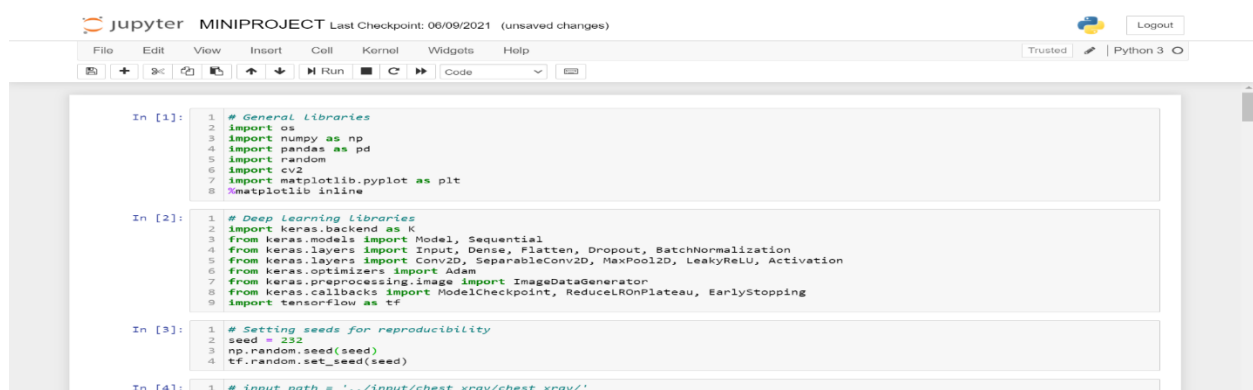
## CHAPTER 6

### RESULTS AND DISCUSSIONS

An important step in documenting is the results of a project to display. It gives the reader an idea of how the program works from the user's point of view and how to use the program. In the following screenshots, the operation of the Pneumonia Detection using Deep Learning is demonstrated.

#### 6.1 OUTPUT SCREENSHOTS

1)



```

jupyter MINIPROJECT Last Checkpoint: 06/09/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: 1 # General Libraries
        2 import os
        3 import numpy as np
        4 import pandas as pd
        5 import random
        6 import cv2
        7 import matplotlib.pyplot as plt
        8 %matplotlib inline

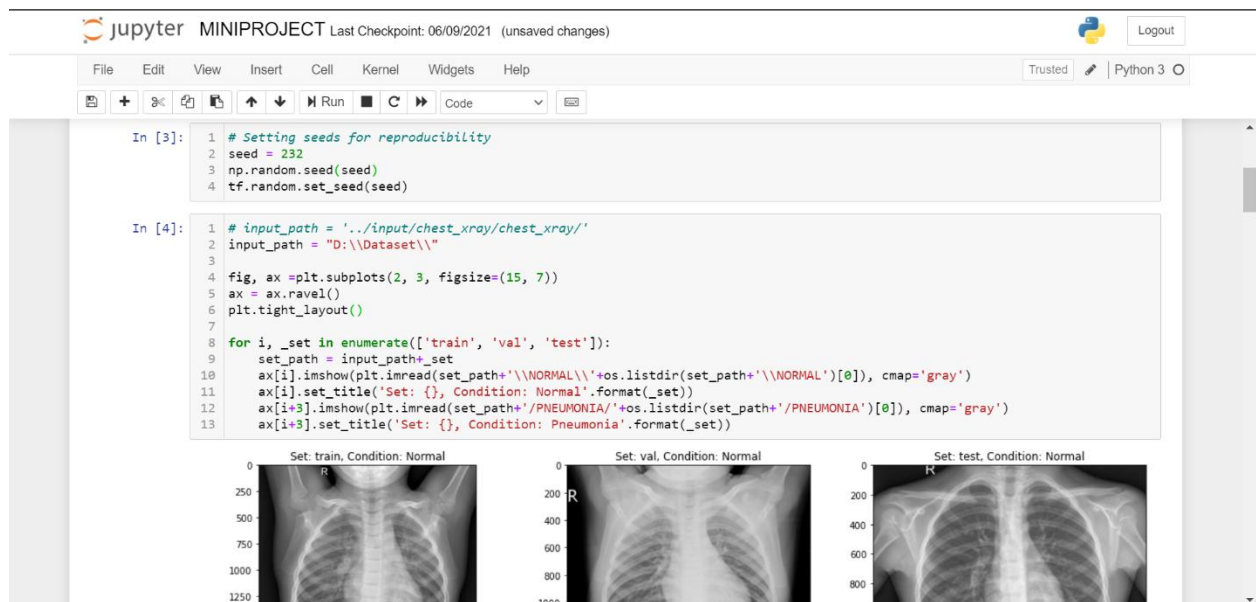
In [2]: 1 # Deep Learning Libraries
        2 import keras.backend as K
        3 from keras.models import Model, Sequential
        4 from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
        5 from keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation
        6 from keras.optimizers import Adam
        7 from keras.preprocessing.image import ImageDataGenerator
        8 from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
        9 import tensorflow as tf

In [3]: 1 # Setting seeds for reproducibility
        2 seed = 232
        3 np.random.seed(seed)
        4 tf.random.set_seed(seed)

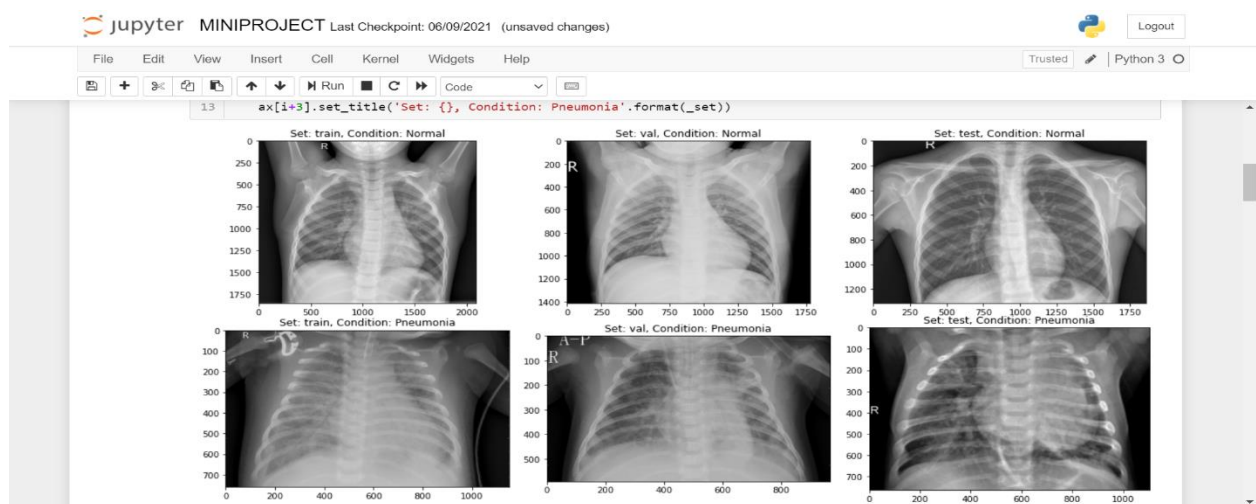
In [4]: 1 # input_path = './input/chest_xray/chest_xray/'

```

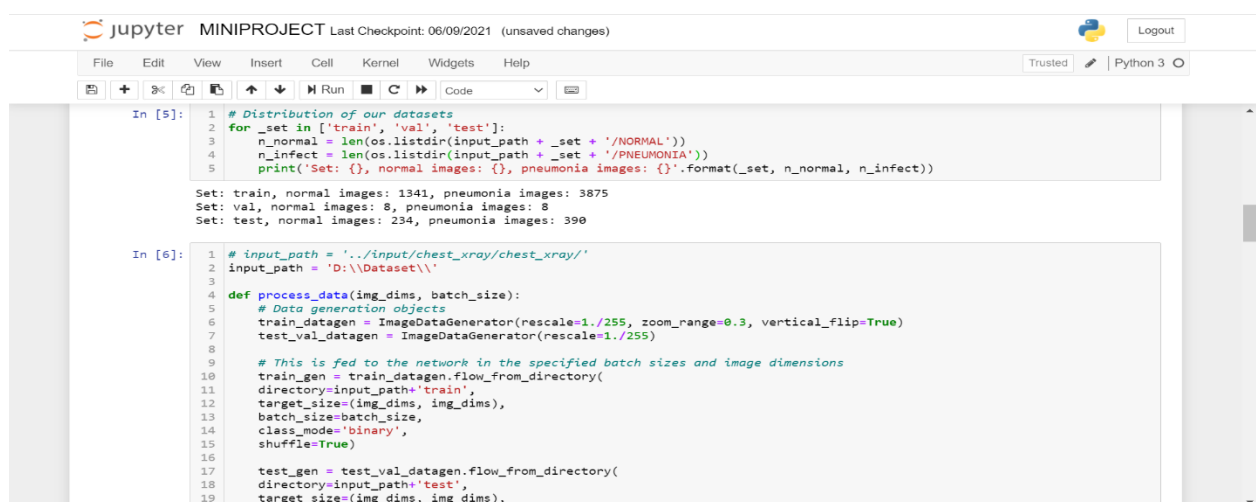
2)



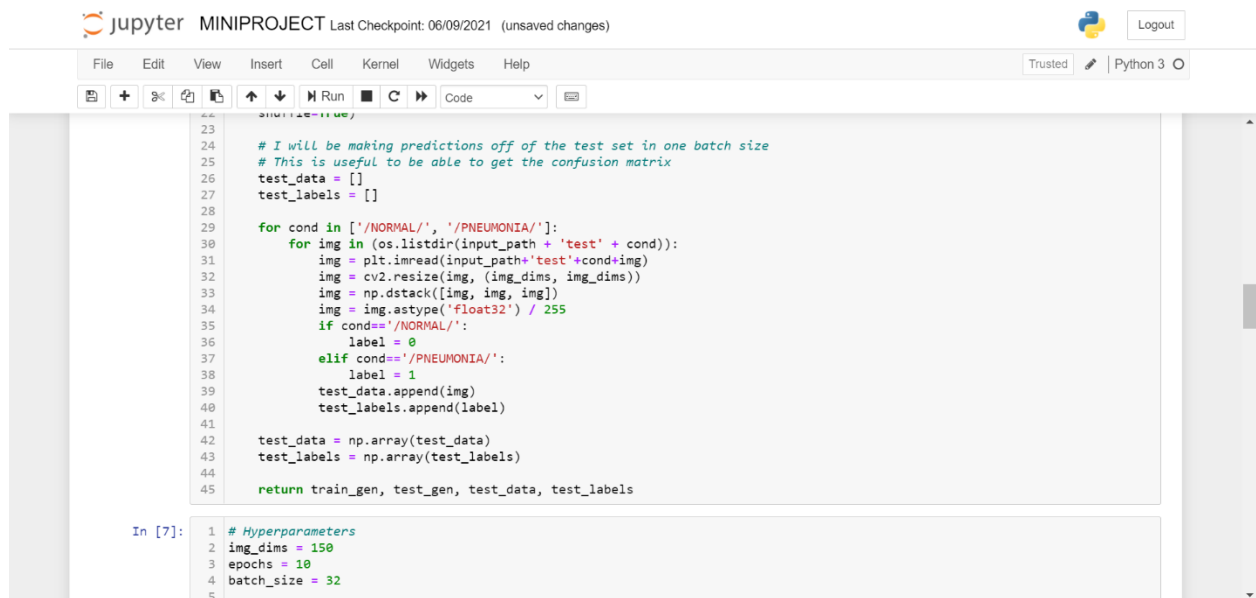
3)



4)



5)

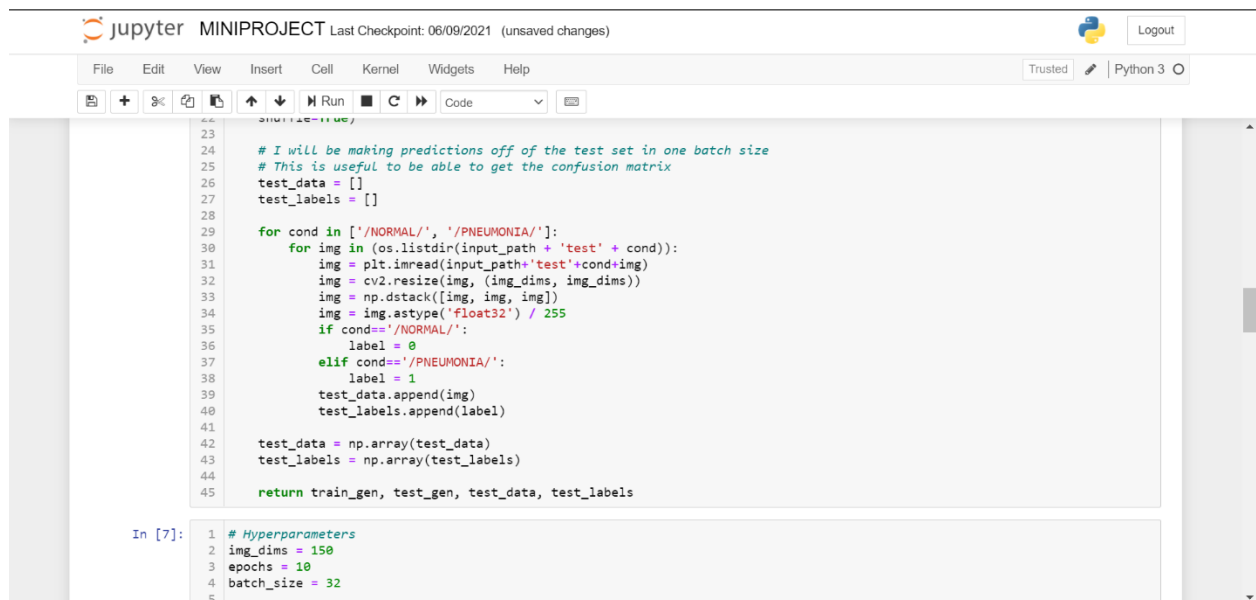


```

23
24
25 # I will be making predictions off of the test set in one batch size
26 # This is useful to be able to get the confusion matrix
27 test_data = []
28 test_labels = []
29
30 for cond in ['/NORMAL/', '/PNEUMONIA/']:
31     for img in os.listdir(input_path + 'test' + cond):
32         img = plt.imread(input_path + 'test' + cond + img)
33         img = cv2.resize(img, (img_dims, img_dims))
34         img = np.dstack([img, img, img])
35         img = img.astype('float32') / 255
36         if cond == '/NORMAL/':
37             label = 0
38         elif cond == '/PNEUMONIA/':
39             label = 1
40         test_data.append(img)
41         test_labels.append(label)
42
43 test_data = np.array(test_data)
44 test_labels = np.array(test_labels)
45
46 return train_gen, test_gen, test_data, test_labels
47
In [7]:
1 # Hyperparameters
2 img_dims = 150
3 epochs = 10
4 batch_size = 32
5

```

6)

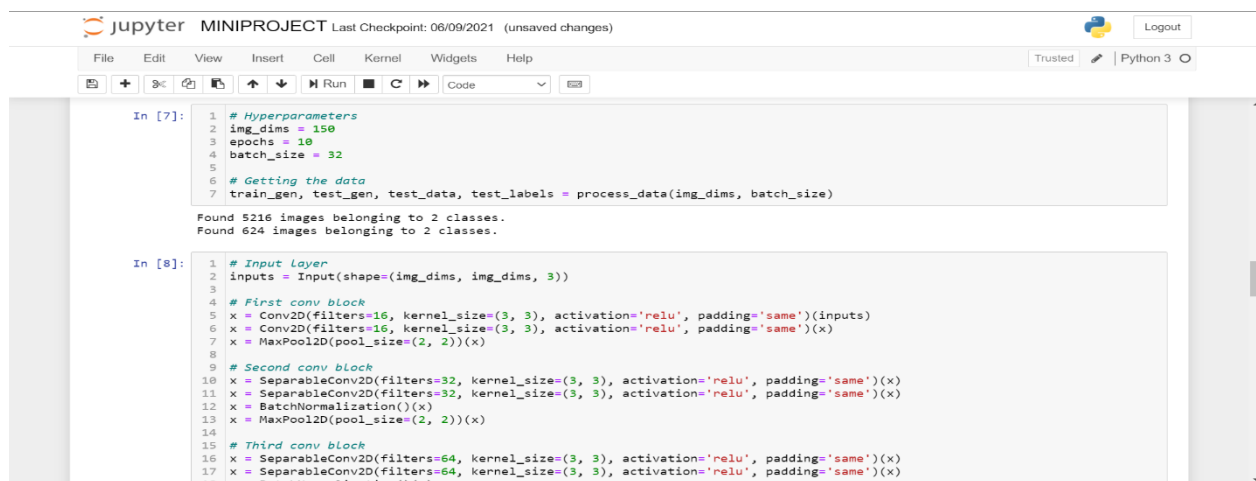


```

23
24
25 # I will be making predictions off of the test set in one batch size
26 # This is useful to be able to get the confusion matrix
27 test_data = []
28 test_labels = []
29
30 for cond in ['/NORMAL/', '/PNEUMONIA/']:
31     for img in os.listdir(input_path + 'test' + cond):
32         img = plt.imread(input_path + 'test' + cond + img)
33         img = cv2.resize(img, (img_dims, img_dims))
34         img = np.dstack([img, img, img])
35         img = img.astype('float32') / 255
36         if cond == '/NORMAL/':
37             label = 0
38         elif cond == '/PNEUMONIA/':
39             label = 1
40         test_data.append(img)
41         test_labels.append(label)
42
43 test_data = np.array(test_data)
44 test_labels = np.array(test_labels)
45
46 return train_gen, test_gen, test_data, test_labels
47
In [7]:
1 # Hyperparameters
2 img_dims = 150
3 epochs = 10
4 batch_size = 32
5

```

7)



```

In [7]:
1 # Hyperparameters
2 img_dims = 150
3 epochs = 10
4 batch_size = 32
5
6 # Getting the data
7 train_gen, test_gen, test_data, test_labels = process_data(img_dims, batch_size)
8
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
In [8]:
1 # Input Layer
2 inputs = Input(shape=(img_dims, img_dims, 3))
3
4 # First conv block
5 x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
6 x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(x)
7 x = MaxPool2D(pool_size=(2, 2))(x)
8
9 # Second conv block
10 x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
11 x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
12 x = BatchNormalization()(x)
13 x = MaxPool2D(pool_size=(2, 2))(x)
14
15 # Third conv block
16 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
17 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
18 x = BatchNormalization()(x)

```



8)



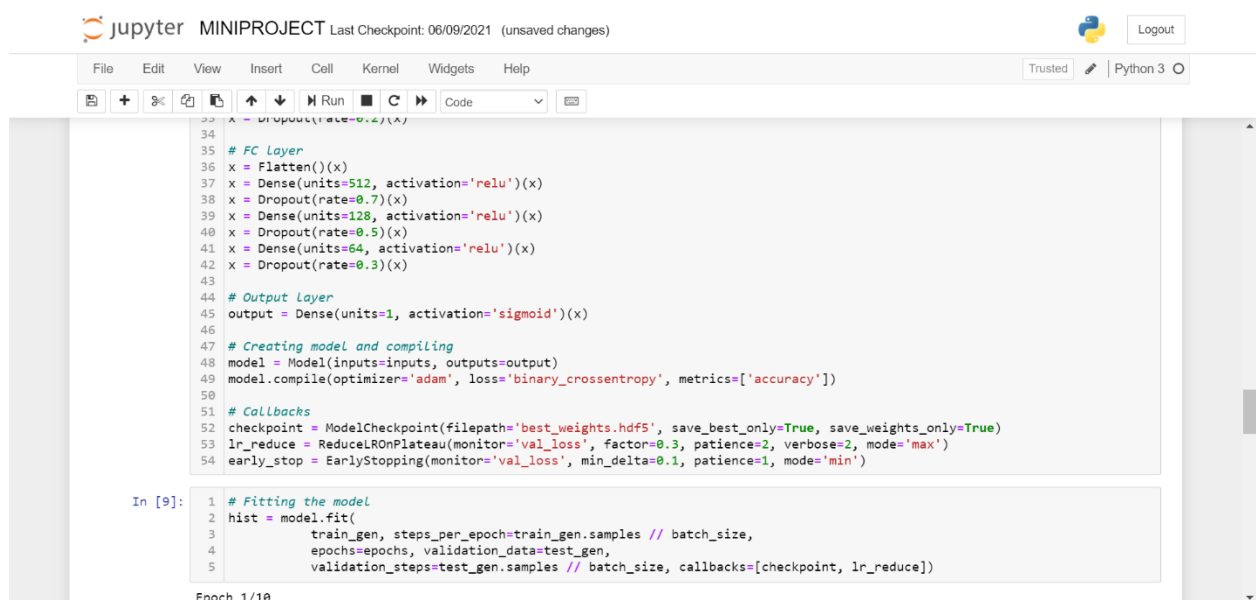
Jupyter MINIPROJECT Last Checkpoint: 06/09/2021 (unsaved changes)

```

11 x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
12 x = BatchNormalization()(x)
13 x = MaxPool2D(pool_size=(2, 2))(x)
14
15 # Third conv block
16 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
17 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
18 x = BatchNormalization()(x)
19 x = MaxPool2D(pool_size=(2, 2))(x)
20
21 # Fourth conv block
22 x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
23 x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
24 x = BatchNormalization()(x)
25 x = MaxPool2D(pool_size=(2, 2))(x)
26 x = Dropout(rate=0.2)(x)
27
28 # Fifth conv block
29 x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
30 x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
31 x = BatchNormalization()(x)
32 x = MaxPool2D(pool_size=(2, 2))(x)
33 x = Dropout(rate=0.2)(x)
34
35 # FC Layer
36 x = Flatten()(x)
37 x = Dense(units=512, activation='relu')(x)
38 x = Dropout(rate=0.7)(x)
39 x = Dense(units=128, activation='relu')(x)
40 x = Dropout(rate=0.5)(x)

```

9)



Jupyter MINIPROJECT Last Checkpoint: 06/09/2021 (unsaved changes)

```

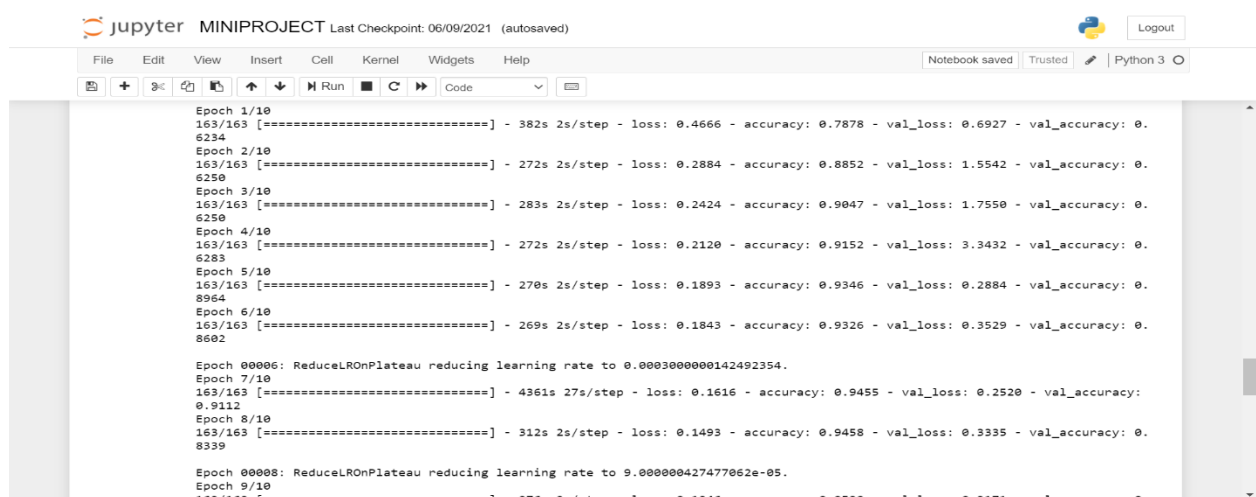
33 x = Dropout(rate=0.2)(x)
34
35 # FC Layer
36 x = Flatten()(x)
37 x = Dense(units=512, activation='relu')(x)
38 x = Dropout(rate=0.7)(x)
39 x = Dense(units=128, activation='relu')(x)
40 x = Dropout(rate=0.5)(x)
41 x = Dense(units=64, activation='relu')(x)
42 x = Dropout(rate=0.3)(x)
43
44 # Output Layer
45 output = Dense(units=1, activation='sigmoid')(x)
46
47 # Creating model and compiling
48 model = Model(inputs=inputs, outputs=output)
49 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
50
51 # Callbacks
52 checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True, save_weights_only=True)
53 lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=2, mode='max')
54 early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')

In [9]: 1 # Fitting the model
2 hist = model.fit(
3         train_gen, steps_per_epoch=train_gen.samples // batch_size,
4         epochs=epochs, validation_data=test_gen,
5         validation_steps=test_gen.samples // batch_size, callbacks=[checkpoint, lr_reduce])

```

Epoch 1/10

10)



Jupyter MINIPROJECT Last Checkpoint: 06/09/2021 (autosaved)

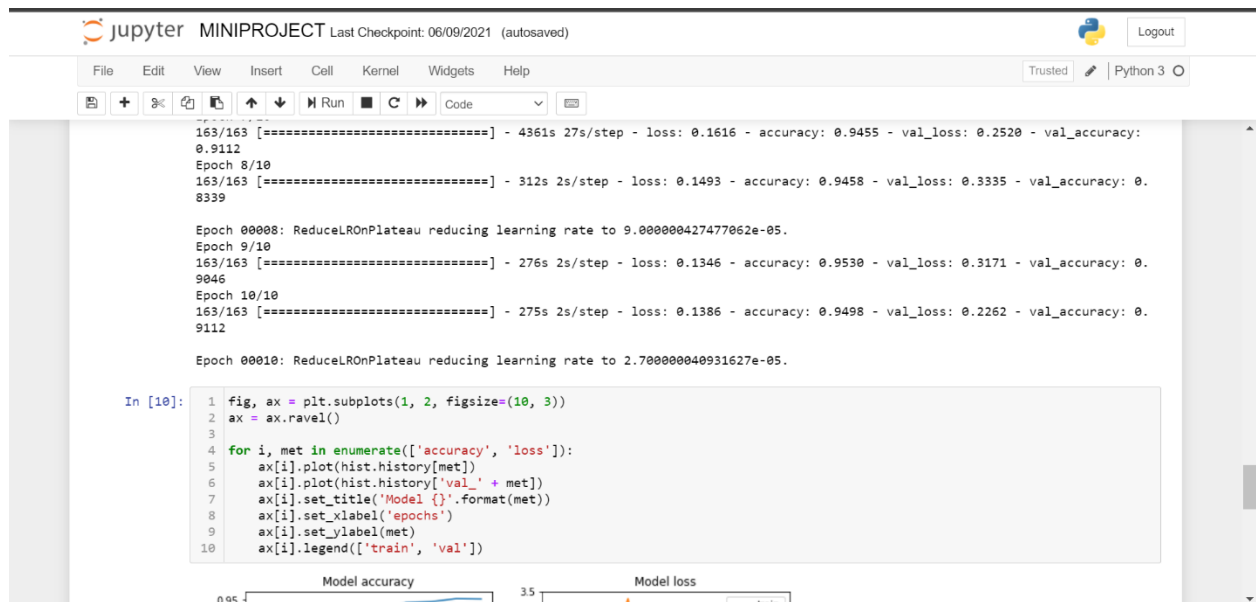
```

Epoch 1/10
163/163 [=====] - 382s 2s/step - loss: 0.4666 - accuracy: 0.7878 - val_loss: 0.6927 - val_accuracy: 0.6234
Epoch 2/10
163/163 [=====] - 272s 2s/step - loss: 0.2884 - accuracy: 0.8852 - val_loss: 1.5542 - val_accuracy: 0.6250
Epoch 3/10
163/163 [=====] - 283s 2s/step - loss: 0.2424 - accuracy: 0.9047 - val_loss: 1.7550 - val_accuracy: 0.6250
Epoch 4/10
163/163 [=====] - 272s 2s/step - loss: 0.2120 - accuracy: 0.9152 - val_loss: 3.3432 - val_accuracy: 0.6283
Epoch 5/10
163/163 [=====] - 270s 2s/step - loss: 0.1893 - accuracy: 0.9346 - val_loss: 0.2884 - val_accuracy: 0.8964
Epoch 6/10
163/163 [=====] - 269s 2s/step - loss: 0.1843 - accuracy: 0.9326 - val_loss: 0.3529 - val_accuracy: 0.8602
Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.003000000142492354.
Epoch 7/10
163/163 [=====] - 4361s 27s/step - loss: 0.1616 - accuracy: 0.9455 - val_loss: 0.2520 - val_accuracy: 0.9112
Epoch 8/10
163/163 [=====] - 312s 2s/step - loss: 0.1493 - accuracy: 0.9458 - val_loss: 0.3335 - val_accuracy: 0.8339
Epoch 00008: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 9/10
163/163 [=====] - 276s 2s/step - loss: 0.1346 - accuracy: 0.9530 - val_loss: 0.3171 - val_accuracy: 0.8339

```



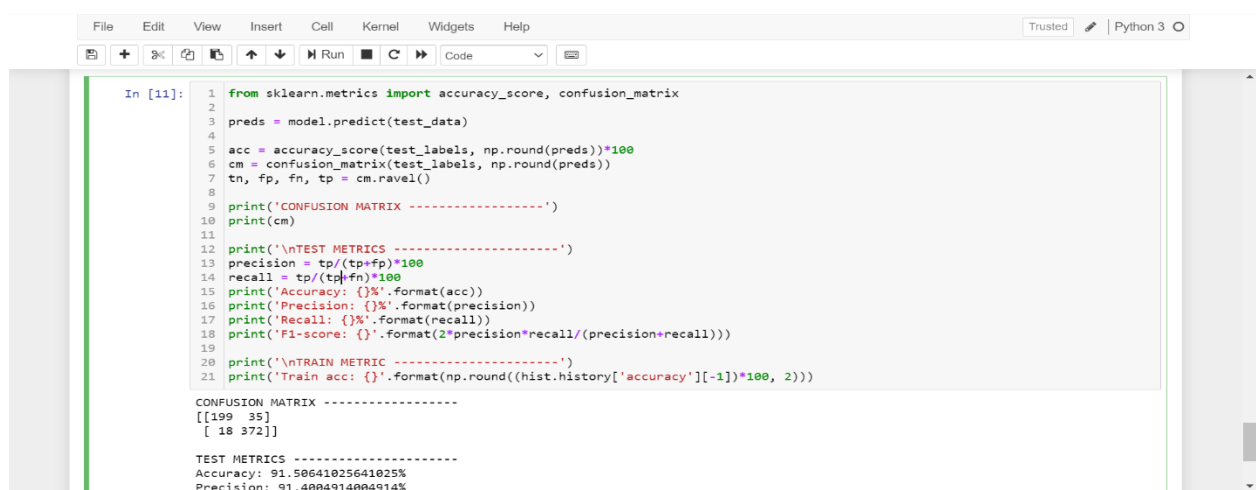
11)



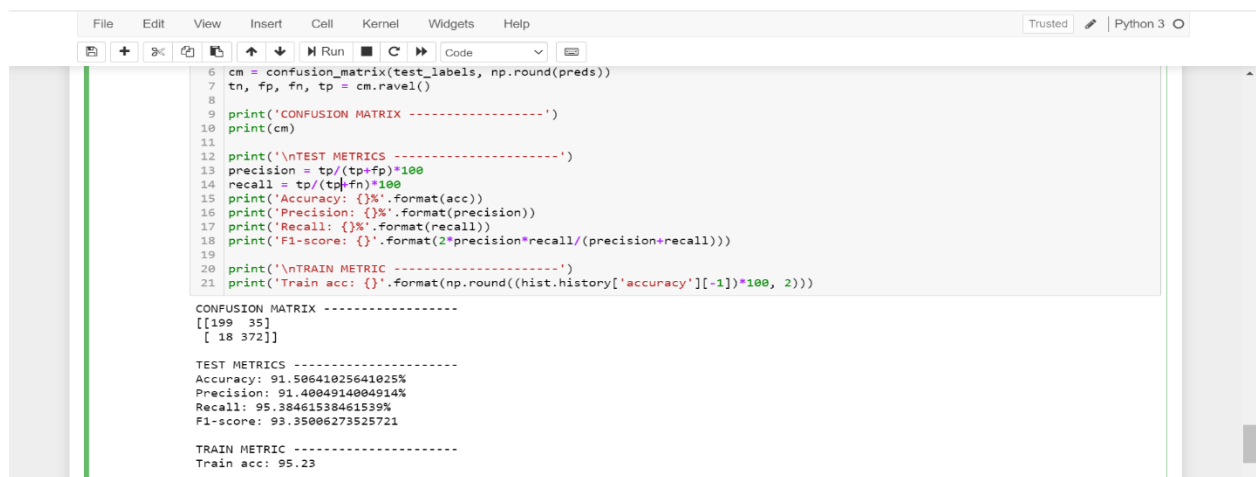
12)



13)



14)



```

6 cm = confusion_matrix(test_labels, np.round(preds))
7 tn, fp, fn, tp = cm.ravel()
8
9 print('CONFUSION MATRIX -----')
10 print(cm)
11
12 print('\nTEST METRICS -----')
13 precision = tp/(tp+fp)*100
14 recall = tp/(tp+fn)*100
15 print('Accuracy: {}'.format(acc))
16 print('Precision: {}'.format(precision))
17 print('Recall: {}'.format(recall))
18 print('F1-score: {}'.format(2*precision*recall/(precision+recall)))
19
20 print('\nTRAIN METRIC -----')
21 print('Train acc: {}'.format(np.round((hist.history['accuracy'][-1])*100, 2)))

```

CONFUSION MATRIX -----

```
[[199  35]
 [ 18 372]]
```

TEST METRICS -----

```
Accuracy: 91.50641025641025%
Precision: 91.4004914004914%
Recall: 95.38461538461539%
F1-score: 93.35006273525721
```

TRAIN METRIC -----

```
Train acc: 95.23
```

## CONCLUSION

With the current implementation of the project, the conclusions that have been drawn are discussed in this chapter.

### 7.1 Conclusion

The proposed problem statement has been solved and realized through python programming language. The program successfully trains the model on the given dataset. The trained model is further tested and then commences to compute the analysis on the input x-rays using the trained model and outputs the result. The above system forms the basis of x-ray image classification using Deep Learning in Statistics. The project was successfully completed giving an almost accurate prediction and analysis

- Over 150 million people get infected by pneumonia on an annual basis especially children under the age of 5 years. Pneumonia is the world's deadliest child killer claiming one young life every 39 seconds. So it is mandatory to come up with an algorithm that will help in detecting Pneumonia.
- Ensuring that the algorithm works accurately will be our utmost concern.

The Model has proven to be the best suited to train the data and to test with the accuracy level of about 91.5%.

## BIBLIOGRAPHY

- [\(193\) Predicting Lungs Disease using Deep Learning - YouTube](#)
- Stack Overflow
- Kaggle
- [Pneumonia Detection using Deep Learning | by Allen Kong | Towards Data Science](#)