DIAGNOSIS OF PNEUMONIA FROM X-RAYS USING DEEP LEARNING

A PROJECT REPORT

Submitted by

PAVITHRA D [211417104181]

POOJA K [211417104185]

PRIYADHARSCINI S.N [211417104202]

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2021

BONAFIDE CERTIFICATE

Certified that this project report **"DIAGNOSIS OF PNEUMONIA FROM X-RAYS USING DEEP LEARNING"** is the bonafide work of "**PAVITHRA D(211417104181), POOJA K(211417104185), PRIYADHARSCINI S.N (211417104202)"**who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,

HEAD OF THE DEPARTMENT
PROFESSOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
COLLEGE,NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

MRS.V.SATHYA PREIYA, M.C.A.,M.PHIL.,M.E.,
SUPERVISOR
ASSOCIATE PROFESSOR
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University

ProjectViva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like express our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** forthe support extended throughout the project.

We would like to thank my **Project Guide V.SATHYA PREIYA, M.C.A.,M.PHIL.,M.E.,**and all the faculty members of the Department of CSE for their advice and suggestions for the successfulcompletion of the project.

PAVITHRA D

POOJA K

PRIYADHARSCINI S.N

ABSTRACT

Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (purulent material), causing cough with phlegm or pus, fever, chills, and difficulty breathing. A variety of organisms, including bacteria, viruses and fungi, can cause pneumonia. It is an infection of the lungs with a range of possible causes. It can be a serious and life-threatening disease. It normally starts with a bacterial, viral, or fungal infection. The lungs become inflamed, and the tiny air sacs, or alveoli, inside the lungs fill up with fluid. Over 150 million people get infected with pneumonia on an annual basis especially children under 5 years old. Some who catch the new coronavirus get severe pneumonia in both lungs. COVID-19 pneumonia is a serious illness that can be deadly. Early detection of Pneumonia and COVID-19 is crucial in reducing mortality. The rich collection of annotated datasets piloted the robustness of deep learning techniques to effectuate the implementation of diverse medical imaging tasks. This proposed system involves detection of Pneumonia and COVID-19 based on deep learning which is proposed for thoracic X-Ray images. The convolution neural network (CNN) is designed to locate the affected Pneumonia and COVID-19 region and hence it can guarantee timely access to treatment and save much needed time and money for those already experiencing poverty.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
5.	**SYSTEM ARCHITECTURE**	17
	5.1 Architecture Overview	17
	5.2 Module Design Specification	17
	5.3 Program Design Language	23
6.	**SYSTEM IMPLEMENTATION**	26
	6.1 Sample Coding	26
7.	**SYSTEM TESTING**	41
	7.1 Test Cases.	41
8.	**CONCLUSION**	44
	8.1 Conclusion and Future Enhancements	44
	APPENDICES	45
	A.1 Sample Screens	45
	A.2 Publications	55
	A.3 Reference	62

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM
COVID	Corona Virus Diesease
CNN	Convolutional Neural Network
FLOP	Floating Point Operation
YOLO	You Only Look Once
SSD	Single Shot Multibox Detector
CT	Computed Tomography
IAVP	Influnzer-A Viral Pneumonia
GGO	Ground Glass Opacity
FC	Fully Connected
GUI	Graphical User Interface
RAM	Random Access Memory
AI	Artificial Intelligence
ReLU	Rectified Linear Unit
DWT	Discrete Wavelet Transform

CHAPTER 1

INTRODUCTION

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus pneumoniae. Some of its symptoms appear suddenly and may include chest pain and difficulty breathing, a high fever, shaking chills, excessive sweating, fatigue, and a cough with phlegm that persists or gets worse. Another infectious illness COVID-19, also known as Severe Acute Respiratory Syndrome Corona virus-2 is a contagious disease that is released from tiny droplets containing saliva or mucus from respiratory system of a diseased person who talks, sneeze, or cough. It spreads rapidly through close contact with somebody who is infected or tapping or holding a virus contaminated objects and surfaces. Older adults and people who have severe underlying medical conditions or prior cases of pneumonia seem to be at higher risk for developing more serious complications from the virus. With rising deaths and limited medical resources, doctors and medical professionals around the world are working around the clock to treat patients and prevent the spread of the virus. Severe forms of the virus can cause pneumonia leading to greater risk of death. It is crucial to have quick and accurate detection of pneumonia so patients can receive treatment in a timely manner especially in impoverished regions. With the growing technological advancements, it is possible to use tools based on deep learning frameworks to detect pneumonia and COVID-19 based on chest x-ray images.

1.1 OVERVIEW

The success of deep learning algorithms in analyzing medical images have lead Convolutional Neural Networks (CNNs) to gain much attention for disease classification. The progression of Deep Learning contributes to aid in the decision-making process of experts to diagnose patients with pneumonia and COVID-19. The

study employs a flexible and efficient approach of deep learning by applying the model of CNN in predicting and detecting a patient's unaffected and affected lungs with the disease , employing a chest X-ray image. The trained-model produced an accuracy rate of 95% during the training. The proposed system can detect and predict pneumonia and COVID-19 diseases based on chest X-ray images.

1.2 PROBLEM DEFINITION

The risk of pneumonia and it's severity in the form of COVID-19 are 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 and in recent times, the deadly coronavirus has claimed hundreds of thousands of lives in countries around the world. 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.

CHAPTER 2

LITERATURE SURVEY

2.1 Pneumonia Detection Using Convolutional Neural Networks (CNNs) by V. Sirish Kaushik, Anand Nayyar, Gaurav Kataria ,Rachna Jain at April 2020.

Pneumonia, an interstitial lung disease, is the leading cause of death in children under the age of five. Timely detection of pneumonia in children can help to fast-track the process of recovery. The paper presents convolutional neural network models to accurately detect pneumonic lungs from chest X-rays, which can be utilized in the real world by medical practitioners to treat pneumonia. CNN models have been created from scratch and trained on Chest X-Ray Images dataset on Kaggle. Keras neural network library with TensorFlow backend has been used to implement the models. Adam optimizer function was finalized to be used for all classifiers. The models presented at best could achieve 92.31% accuracy.

2.2 Multi-Objective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification by Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb,Erik Goodman, Wolfgang Banzhaf ,Vishnu Naresh Boddeti at September 2020.

Convolutional neural networks (CNNs) are the backbones of deep learning paradigms for numerous vision tasks. The paper proposes an evolutionary algorithm for searching neural architectures under multiple objectives, such as classification performance and floating point operations (FLOPs). By populating a set of architectures to approximate the entire Pareto frontier through genetic operations that recombine and modify architectural components progressively the proposed method overcomes the problem where obtained architectures are either solely optimized for classification performance, or only for one deployment scenario. Analysis towards validating the generalization and robustness aspects of the obtained architectures is also provided

along with an application to common thorax disease classification on human chest X-rays.

2.3 Covid-Cxnet: Detecting covid-19 in frontal chest x-ray images using deep learning by Arman haghanifar,Mahdiyar molahasani Majdabadi,Younhee choi,s. Deivalakshmi,Seokbum ko at July 2020.

One of the primary clinical observations for screening the novel coronavirus is capturing a chest x-ray image. In most patients, a chest x-ray contains abnormalities, such as consolidation, resulting from COVID-19 viral pneumonia. In this paper, numerous chest x-ray images from various sources are collected, and the largest publicly accessible dataset is prepared. Using the transfer learning paradigm, the well-known CheXNet model is utilized to develop COVID-CXNet.At first, a base convolutional model is designed and trained on different portions of the dataset. Then, pretrained models based on the ImageNet dataset are discussed. Finally, a pretrained model on a similar image type is explained. In order to train the base model, the optimizer is set to “adam” with the optimal learning rate obtained using exponentially learning rate increasing method.

2.4 Pneumonia Classification using Deep Learning in Healthcare by Garima Verma, Shiva Prakash at February 2020.

Pneumonia is a disease that is caused by various bacteria, virus etc. X-ray is one of the major diagnosis tools for diagnosing pneumonia. The research work mainly proposes a convolutional neural system (CNN) model prepared without any preparation to group and identify the occurrence of pneumonia disease from a given assortment of chest X-ray image tests.The Data Augmentation techniques have helped to perform various types of operations in the images. Keras which is an open-source neural network library in deep learning having tensorflow backend is used. Some of the research challenges include lack of homogeneity in the architectures, frameworks and devices is and lack of image datasets and information leads to inaccurate results.

2.5 Pneumonia Detection Using Convolutional Neural Networks by Sammy V. Militante, Brandon G. Sibbaluca at April 2020.

Pneumonia is an infectious and deadly illness in respiratory that is caused by bacteria, fungi, or a virus that infects the human lung air sacs with the load full of fluid or pus. Chest X-rays are the common method used to diagnose pneumonia and it needs a medical expert to evaluate the result of X-ray. Convolutional Neural Network is optimized to perform the complicated task of detecting diseases like pneumonia to assist medical experts in diagnosis and possible treatment of the disease. Multi-layered structure includes convolution layer, Max-pooling layer and fully connected layer. The proposed methodology presents the architectural design that is divided into three stages as pre-processing, handover learning and refinement, and classification. The ResNet model has the lowest accuracy rate of 74%.With this work, a distinguished process of diagnosing and detecting pneumonia can helpful in providing medical services.

2.6 Early Diagnosis of Pneumonia with Deep Learning by Deniz Yagmur Urey, Can Jozef Saul ,Doruk Taktakoglu at 2020.

One of the most conventional medical techniques used to diagnose the disease is chest x-ray. As the concentrated beam of electrons, called x-ray photons, go through the body tissues, an image is produced on the metal surface. During diagnosis, expert radiologists correspond white spots on the image to infiltrates identifying an infection, and white areas to the pneumonia fluid in the lungs. However, the limited color scheme of x-ray images consisting of shades of black and white, cause drawbacks when it comes to determining whether there is an infected area in the lungs or not. YOLO and SSD algorithms might be effective for the localization of the pneumonia region, while different preprocessing methods may be needed for training the respective algorithms which is the disadvantage.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System

The Computed Tomography (CT) imaging plays a critical role for detection of manifestations in the lung associated with COVID-19, where segmentation of the infection lesions from CT scans is important for quantitative measurement of the disease progression in accurate diagnosis and follow-up assessment. The manifestations of pneumonia as seen through computed tomography (CT) imaging show individual characteristics that differ from those of other types of viral pneumonia such as influenza-A viral pneumonia (IAVP). As manual segmentation of the lesions from 3D volumes is labor-intensive, time-consuming and suffers from inter- and intra-observer variabilities, automatic segmentation of the lesions is highly desirable in clinic practice. Despite its importance for diagnosis and treatment decisions, automatic segmentation of COVID-19 pneumonia lesions from CT scans is challenging due to several reasons. First, the infection lesions have a variety of complex appearances such as Ground-Glass Opacity (GGO), reticulation, consolidation and others. Second, the sizes and positions of the pneumonia lesions vary largely at different stages of the infection and among different patients. In addition, the lesions have irregular shapes and ambiguous boundaries, and some lesion patterns such as GGO have a low contrast with surrounding region. These challenges not only make it difficult to automatically segment the lesions, but also bring obstacles for obtaining accurate manual annotations for training. For medical image segmentation, its success relies on accurate annotation of a large set of training images implemented by experts, which is expensive to acquire and limited by the availability of experts. The experimental result of the benchmark dataset showed that the overall accuracy rate was 86.7% in terms of all the CT cases taken together.

3.2 Proposed System

Chest X-rays are one of the best methods for the detection of pneumonia. X-ray imaging is preferred over CT imaging because X-ray imaging typically takes considerably less time than CT imaging. X-rays are the most common and widely available diagnostic imaging technique, playing a crucial role in clinical care and epidemiological studies. The proposed methodology uses a deep transfer learning algorithm using CNN that extracts the features from the X-ray image that describes the presence of disease automatically and reports whether it is a case of pneumonia or COVID-19 or normal.

3.3 Requirement Analysis and Specification

3.3.1 Input Requirements

A dataset (or data set) is a collection of data, usually presented in tabular form. Each column represents a particular variable. Each row corresponds to a given member of the dataset in question. It lists values for each of the variables, such as height and weight of an object. Chest X-Ray Dataset is part of the COVID-19 and pneumonia Image Data Collection of chest X-ray of patients which are Normal or suspected of COVID-19 or pneumonia, in which 144 images are chest X-rays.

3.3.2 Output Requirements

Trained CNN model to train and test each x-ray image in which those images will pass through series of convolutional layers with filters (Kernals), Pooling, Fully Connected layers(FC),and apply Softmax function to classify an object with probabilistic values between 0 and 1.

3.3.3 Software Requirements

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows to easily manage conda packages,

environments, and channels without using command-line commands. Python Idle version 3.6 and above is required. A new environment (Pneu Packages) is created with the necessary packages of python.

3.3.4 Hardware Requirements

A minimum of 8GB RAM can do the job but 16GB RAM and above is recommended for deep learning tasks. When it comes to CPU, a minimum of 7th generation (Intel Core i7 processor) is recommended. However, getting Intel Core i5 with Turbo Boosts can do the trick. Windows Operating System Version 10 is used.

3.4 Technology Stack

Domain -Deep Learning

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

Front end- Python 3.9

- **OpenCV**
- **NumPy**
- **scikit-learn**
- **pandas**
- **matplotlib**
- **keras**

OpenCV

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. It makes use of Numpy, which is a highly optimized library for numerical

operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. It is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays. This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib. It is an open source computer vision and machine learning software library . It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

NumPy

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages which cannot be missed while learning data science , mainly because this library provides an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient. To make a numpy array, np.array() function is used.

Scikit-learn

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

The functionality that scikit-learn provides include:

- **Regression**, including Linear and Logistic Regression
- **Classification**, including K-Nearest Neighbors
- **Clustering**, including K-Means and K-Means++
- **Model selection**
- **Preprocessing**, including Min-Max Normalization

Pandas

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

We can analyze data in pandas with:

- ✓ Series
- ✓ DataFrames

Series

Series is one dimensional(1-D) array defined in pandas that can be used to store any data type.

DataFrames

DataFrames is two-dimensional(2-D) data structure defined in pandas which consists of rows and columns.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It consists of several plots like line, bar, scatter, histogram etc.

Keras

Keras is an open-source neural-network **library** written in **Python**. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It is developed using four guiding principles:

1.Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

2.Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

3.Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

4.Python: No separate model files with custom file formats. Everything is native Python.

Back end - TENSORFLOW 2.0

TensorFlow is an end-to-end open source platform for deep learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in deep learning and developers easily build and deploy deep learning powered applications. It is a general-purpose high-performance computing library open-sourced by Google in 2015. Since the beginning, its main focus was to provide high-performance APIs for building Neural Networks (NNs). However, with the advance of time and interest by the Deep learning community, the lib has grown to a full deep learning ecosystem.

CHAPTER 4

SYSTEM DESIGN

4.1 Data Dictionary

The data dictionary contains all the metadata (data about the data) for the files, tables and columns in a dataset. For all files it contains: The names of all the files in the dataset.

Train.csv

X-RAY IMAGE	LABEL	CLASS ID	CLASS
N1	TRAIN	1	NORMAL
N2	TRAIN	1	NORMAL
N3	TRAIN	1	NORMAL
N4	TRAIN	1	NORMAL
N5	TRAIN	1	NORMAL
N6	TRAIN	1	NORMAL
N7	TRAIN	1	NORMAL
N8	TRAIN	1	NORMAL
N9	TRAIN	1	NORMAL
N10	TRAIN	1	NORMAL
N11	TRAIN	1	NORMAL
N12	TRAIN	1	NORMAL
N13	TRAIN	1	NORMAL
N14	TRAIN	1	NORMAL
N15	TRAIN	1	NORMAL
N16	TRAIN	1	NORMAL
N17	TRAIN	1	NORMAL
N18	TRAIN	1	NORMAL
N19	TRAIN	1	NORMAL
N20	TRAIN	1	NORMAL
P1	TRAIN	0	PNEUMONIA
P2	TRAIN	0	PNEUMONIA
P3	TRAIN	0	PNEUMONIA
P4	TRAIN	0	PNEUMONIA
P5	TRAIN	0	PNEUMONIA
P6	TRAIN	0	PNEUMONIA
P7	TRAIN	0	PNEUMONIA
P8	TRAIN	0	PNEUMONIA
P9	TRAIN	0	PNEUMONIA

P10	TRAIN	0	PNEUMONIA
P11	TRAIN	0	PNEUMONIA
P12	TRAIN	0	PNEUMONIA
P13	TRAIN	0	PNEUMONIA
P14	TRAIN	0	PNEUMONIA
P15	TRAIN	0	PNEUMONIA
P16	TRAIN	0	PNEUMONIA
P17	TRAIN	0	PNEUMONIA
P18	TRAIN	0	PNEUMONIA
P19	TRAIN	0	PNEUMONIA
C1	TRAIN	2	COVID
C2	TRAIN	2	COVID
C3	TRAIN	2	COVID
C4	TRAIN	2	COVID
C5	TRAIN	2	COVID
C6	TRAIN	2	COVID
C7	TRAIN	2	COVID
C8	TRAIN	2	COVID
C9	TRAIN	2	COVID
C10	TRAIN	2	COVID
C11	TRAIN	2	COVID
C12	TRAIN	2	COVID
C13	TRAIN	2	COVID
C14	TRAIN	2	COVID
C15	TRAIN	2	COVID
C16	TRAIN	2	COVID
C17	TRAIN	2	COVID
C18	TRAIN	2	COVID
C19	TRAIN	2	COVID
C20	TRAIN	2	COVID

Fig:4.1(a) Train.csv

Test.csv

X-RAY IMAGE	LABEL	CLASS ID	CLASS
N1	TEST	1	NORMAL
N2	TEST	1	NORMAL
N3	TEST	1	NORMAL
P18	TEST	0	PNEUMONIA
P19	TEST	0	PNEUMONIA
C4	TEST	2	COVID
C16	TEST	2	COVID
P6	TEST	0	PNEUMONIA
C13	TEST	2	COVID

Fig:4.1(b) Test.csv

4.2 UML DIAGRAMS

4.2.1 Use Case Diagram

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow, however, use case diagram does not describe how those events are implemented.

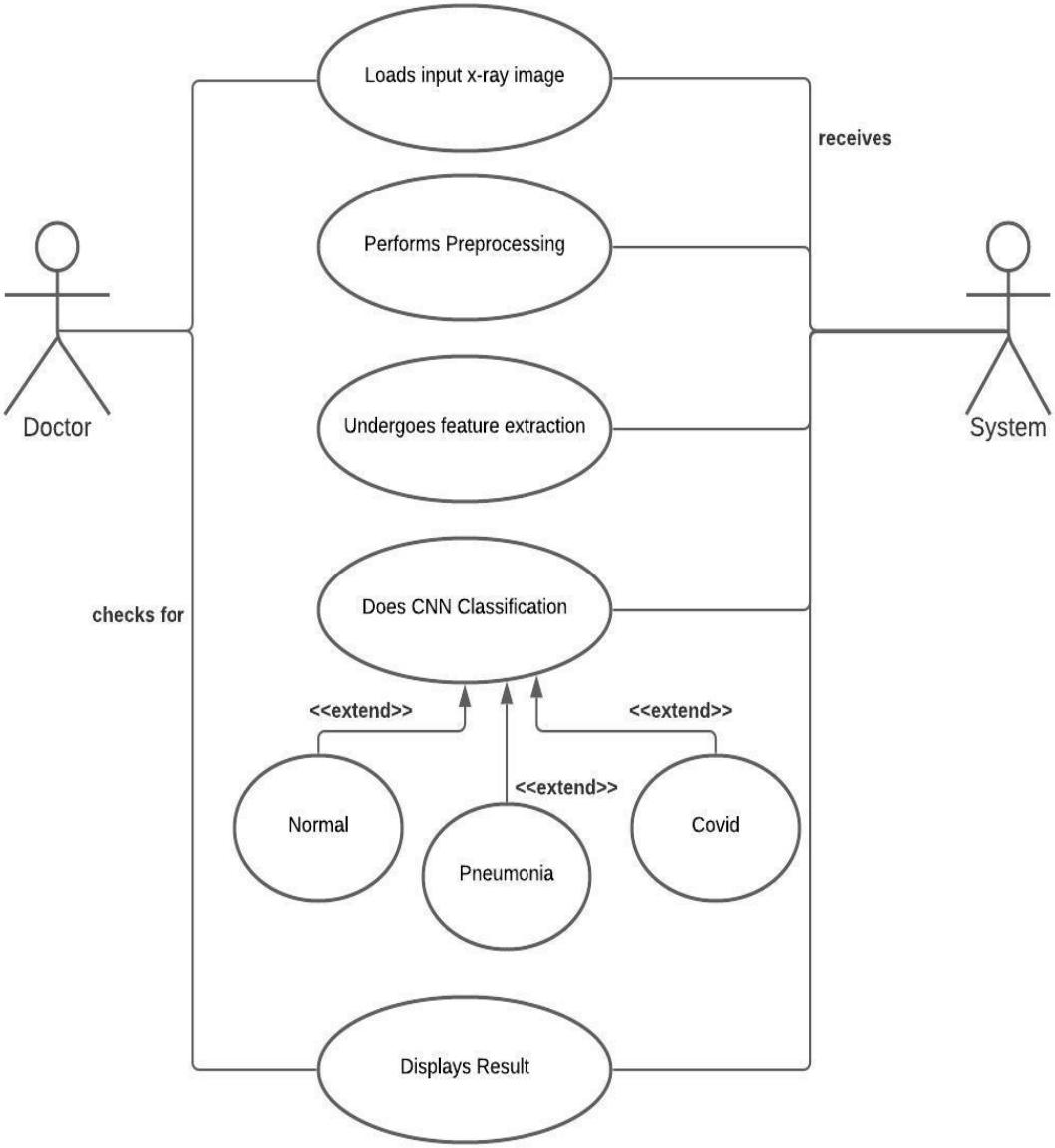


Fig:4.2.1 Use case Diagram for pneumonia detection

4.2.2 Activity Diagram

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).

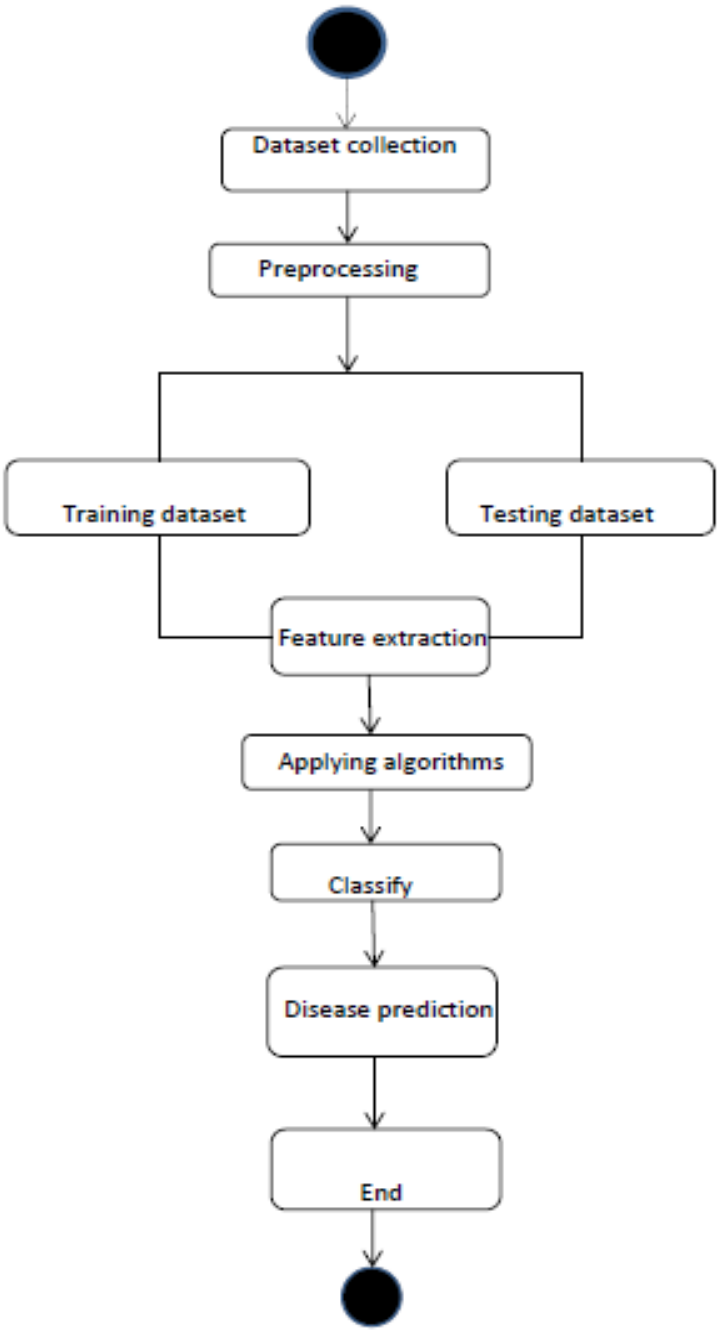


Fig:4.2.2 Activity Diagram for pneumonia detection

4.2.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

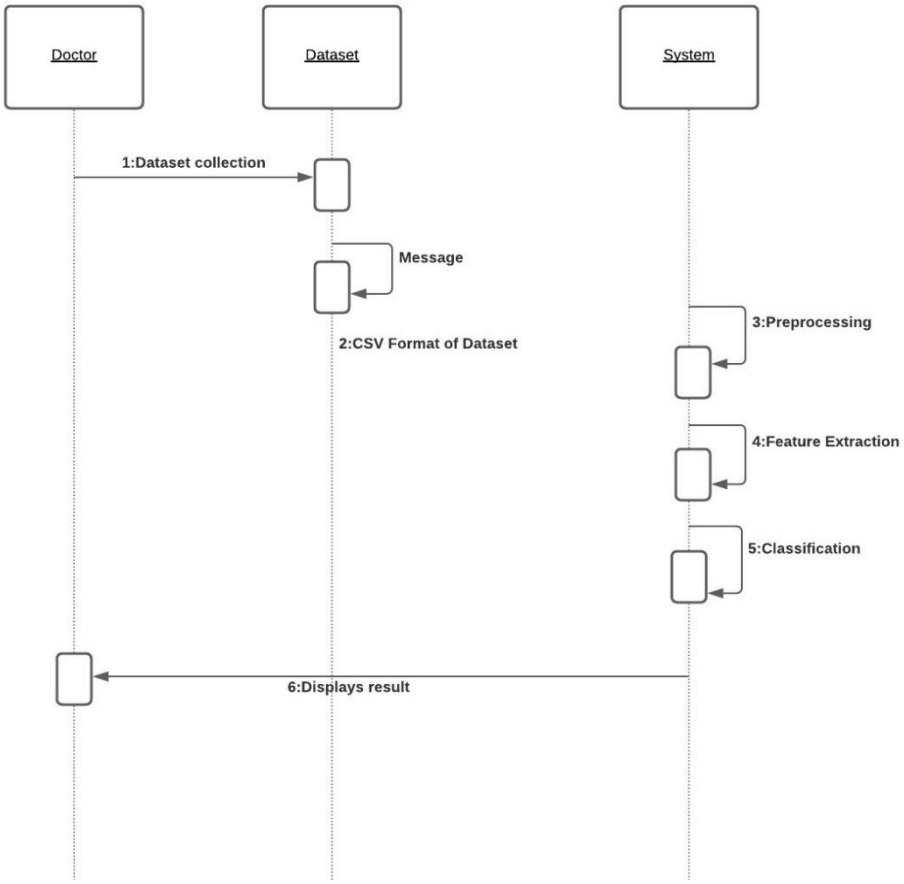


Fig:4.2.3 Sequence Diagram for pneumonia detection

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 Architecture Overview

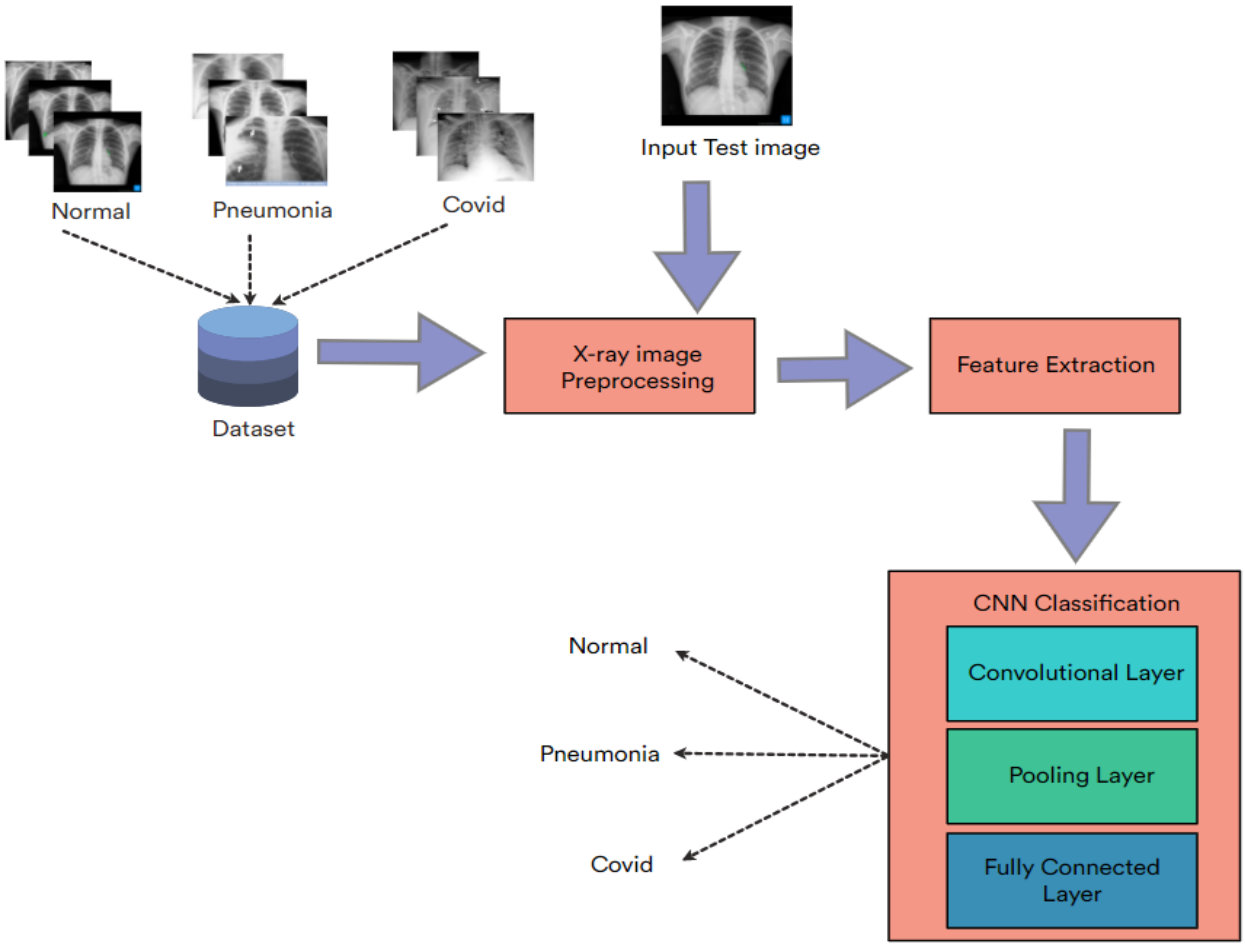


Fig:5.1 Architecture Diagram for pneumonia detection

5.2 Module Design Specification

5.2.1 Exploring the Dataset

The dataset that will be used for this project will be the Chest X-Ray Images (Pneumonia) from Kaggle. The dataset consists of training data, validation data, and testing data. The training data consists of 5,216 chest x-ray images with 1,738 images

shown to have pneumonia , 1,738 images shown to be COVID and 1,740 images shown to be normal. The validation data is relatively small with only 32 images with 8 cases of pneumonia ,8 cases of COVID and 8 normal cases. The testing data consists of 624 images split between 390 pneumonia cases ,34 COVID cases and 200 normal cases.

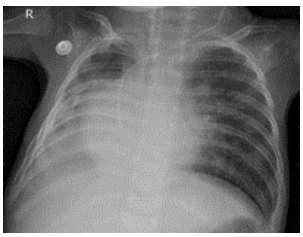
	NORMAL	PNEUMONIA	COVID
TRAIN			
TEST			
VALIDATE			

Fig:5.2.1 Sample Dataset

5.2.2 Pre-processing

The X-Ray image of a person is fed as the input to the pre-processor. If the images are of low resolution or poor contrast, the pre-processor will enhance the contrast to get the accurate classification type. It enhances the better visualization of the image.

Steps involved:1) Read the image

2) Resize the image

3) Remove noise

4) Segmentation.

DWT is the lossless image compression technique. High quality images that require large storage are to be compressed. So DWT is required to compress the image without any appreciable loss of information. Digitize the source image into signal. Decompose signal to wavelet (sub bands) LL,HL,HH .DWT retains images from LL to produce next level of decomposition, because the low frequency images has finer frequency and time resolution than high frequency images. For each level of decomposition DWT produces 4 images and size is reduced to 1/4 of original image.

5.2.3 CNN (Convolutional Neural Network) Classification

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analysing visual images. Their applications can be seen widely in the medical images analysis. The term "Convolution" in CNN denotes that two images can be represented as matrices which are multiplied to give an output that is used to extract features from the image.

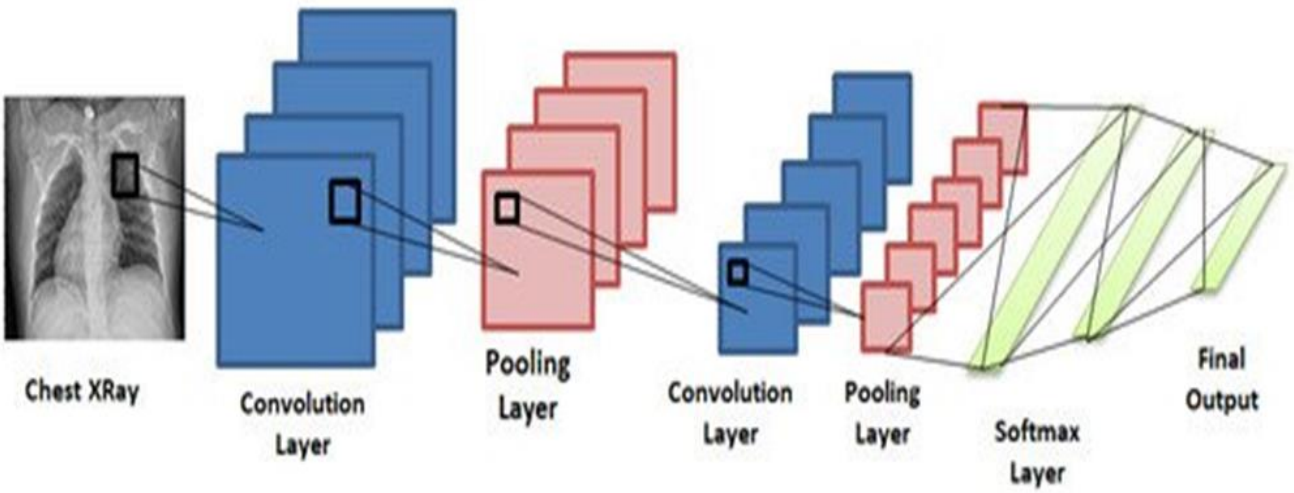


Fig:5.2.3 CNN model Diagram

5.2.4 Layers of CNN

1.Input Layer

Modified GLCM image is fed as a input in the form of 3-D array of pixel values.

2. Convolutional Layer

In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM). The dot products so obtained are called **feature maps**. The sum of those dot products is used to produce the output image which is fed as input to next layer.

3. ReLU Layer (rectified linear activation function or ReLU)

ReLU (Rectifier Linear Unit) is one of the activation function. Its formula is max(x,0) means if the resultant value coming from node is positive then output would be the same positive value and if it is negative value then output would be zero. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer

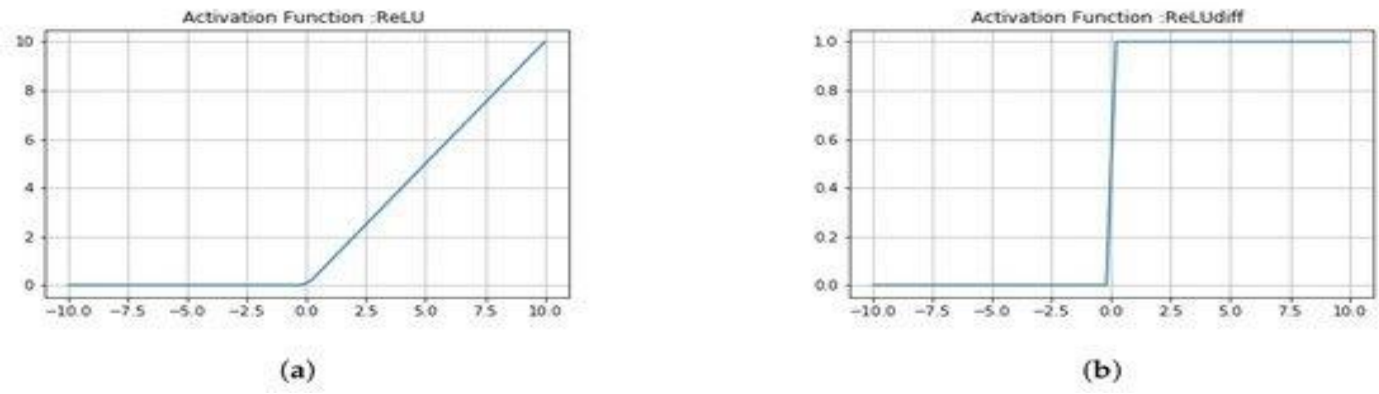


Fig:5.2.4(a) ReLU Activation Function Diagram

4. Pooling Layer

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently. The most common approach used in pooling is **max pooling.**

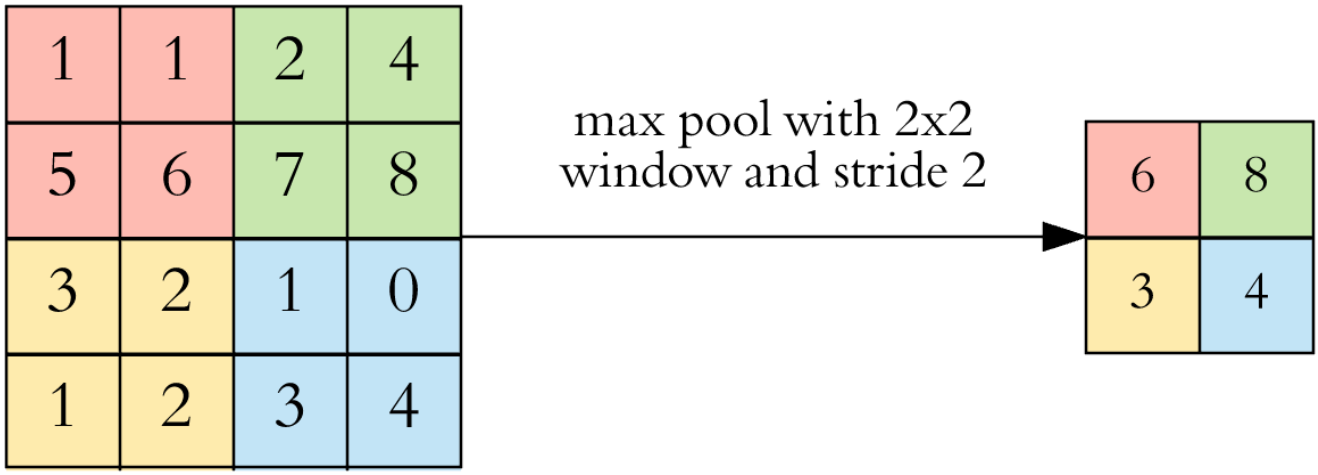


Fig:5.2.4(b) Max Pooling Diagram

5.Fully-Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the class score for each of the classification category. In this, the input image from the previous

layers are flattened and fed to the FC layer. In this stage, the classification process begins to take place.

6.Drop out Layer

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model.

5.2.5 Softmax Function

The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. That is, softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels. The final output is calculated using softmax which gives the probability of each class for given features.

5.2.6 Adam Optimizer

The **Adam optimization algorithm** is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights **iterative based in training data**. The attractive benefits of using Adam are,

- ✓ Straightforward to implement.
- ✓ Computationally efficient.
- ✓ Little memory requirements.
- ✓ Invariant to diagonal rescale of the gradients.

- ✓ Well suited for problems that are large in terms of data and/or parameters.
- ✓ Appropriate for non-stationary objectives.
- ✓ Appropriate for problems with very noisy/or sparse gradients.
- ✓ Hyper-parameters have intuitive interpretation and typically require little tuning.

5.2.7 Training

The sample of data used to fit the model. The actual dataset that used to train the model (weights and biases in the case of Neural Network). The model sees and learns from this data. A number of recommended approaches in the toolkit that could be experimented with,

- ✓ Weight Initialization.
- ✓ Learning Rate.
- ✓ Activation Functions.
- ✓ Network Topology.
- ✓ Batches and Epochs.
- ✓ Regularization.
- ✓ Optimization and Loss.
- ✓ Early Stopping.

5.2.8 Testing

The dataset contains a test folder and in a test.csv file, the details related to the image path and their respective class labels are specified. The image path and labels are extracted using pandas. Then to predict the model, the images are resized to 30×30 pixels and numpy array containing all image data are made. From the sklearn.metrics, the confusion_matrix is imported and observed how the model predicted the actual labels. As a result 95% accuracy of the model is achieved.

5.2.9 Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

5.3 ALGORITHM

5.3.1 Initial Setup

1.Install the anaconda navigator and python idle.

2.Create the environment (Pneu Packages) in the anaconda navigator with the required packages installed.

5.3.2 Dataset

1.Import the necessary packages.

2.Load the train images.

3.Load the test images.

4.Read the train images.

5.Read the test images.

5.3.3 Cnn_train_test

1.Import the necessary packages.

2.Initialize the filter_size and num_filters in the convolutional layers.

3.Specify the required classes(Pneumonia,Normal,Covid) for the categorization.

4.Specify the train_path, test_path, checkpoint_dir and val_path.

5.Read the dataset details.

6.Get some random images and their labels from the train set.

7.Plot the images and labels using the helper function.

8.Create new weights with the given shape of the image.

9.Create new biases, one for each filter.

10.Add the biases to the result of convolution.

11.Perform max pooling and padding in the pooling layer.

12.In the flatten layer, get the shape of the input image.

13.Create new weights and biases.

14.Classification and optimization are performed in the fully connected layer.

15.In the Tensorflow session, the epochs with validation loss, validation accuracy and training accuracy are predicted.

16.To measure the accuracy,the confusion matrix is constructed.

5.3.4 Output Procedure

1.Open the terminal with the environment created in the anaconda navigator.

2.Move to the particular Directory.

3.Run the cnn_train_test.py python file.

4.Accuracy of training, testing is obtained with the confusion matrix.

5.Actual output after classification is displayed with the input image as frame.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Sample Code

cnn_test_train.py

```
from tkinter import *
import time
import math
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import dataset
import cv2
import os
import tkinter as tk
from sklearn.metrics import confusion_matrix
from datetime import timedelta
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy


filter_size1 = 3
num_filters1 = 32
filter_size2 = 3
```

```
num_filters2 = 32
filter_size3 = 3
num_filters3 = 64
fc_size = 128
num_channels = 3
img_size = 32
img_size_flat = img_size * img_size * num_channels
img_shape = (img_size, img_size)
classes = ['Pneumonia','Normal','Covid']
num_classes = len(classes)
batch_size = 1
validation_size = .16
early_stopping = None
train_path = 'data/'
test_path = 'test/'
checkpoint_dir = "models/"
data = dataset.read_train_sets(train_path, img_size, classes, validation_size=validation_size)
test_images, test_ids = dataset.read_test_set(test_path, img_size)
print("Size of:")
print("- Training-set: {}".format(len(data.train.labels)))
print("- Test-set: {}".format(len(test_images)))
print("- Validation-set: {}".format(len(data.valid.labels)))
images, cls_true  = data.train.images, data.train.cls

def new_weights(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.05))
def new_biases(length):
    return tf.Variable(tf.constant(0.05, shape=[length]))
```

```
def new_conv_layer(input,              # The previous layer.
                   num_input_channels, # Num. channels in prev. layer.
                   filter_size,        # Width and height of each filter.
                   num_filters,        # Number of filters.
                   use_pooling=True):  # Use 2x2 max-pooling.

    shape = [filter_size, filter_size, num_input_channels, num_filters]
    weights = new_weights(shape=shape)
    biases = new_biases(length=num_filters)
    layer = tf.nn.conv2d(input=input,
                         filter=weights,
                         strides=[1, 1, 1, 1],
                         padding='SAME')
    layer += biases

    if use_pooling:
        layer = tf.nn.max_pool(value=layer,
                               ksize=[1, 2, 2, 1],
                               strides=[1, 2, 2, 1],
                               padding='SAME')

    layer = tf.nn.relu(layer)
    return layer, weights

def flatten_layer(layer):

    layer_shape = layer.get_shape()
    num_features = layer_shape[1:4].num_elements()
    layer_flat = tf.reshape(layer, [-1, num_features])
```

```
    return layer_flat, num_features

def new_fc_layer(input,
                 num_inputs,
                 num_outputs,
                 use_relu=True):
    weights = new_weights(shape=[num_inputs, num_outputs])
    biases = new_biases(length=num_outputs)
    layer = tf.matmul(input, weights) + biases



    if use_relu:
        layer = tf.nn.relu(layer)

    return layer

x = tf.compat.v1.placeholder(tf.float32, shape=[None, img_size_flat], name='x')
x_image = tf.reshape(x, [-1, img_size, img_size, num_channels])
y_true = tf.placeholder(tf.float32, shape=[None, num_classes], name='y_true')
y_true_cls = tf.argmax(y_true, dimension=1)
layer_conv1, weights_conv1 = \
    new_conv_layer(input=x_image,
                   num_input_channels=num_channels,
                   filter_size=filter_size1,
                   num_filters=num_filters1,
                   use_pooling=True)



layer_conv1
```

```
layer_conv2, weights_conv2 = \
    new_conv_layer(input=layer_conv1,
                   num_input_channels=num_filters1,
                   filter_size=filter_size2,
                   num_filters=num_filters2,
                   use_pooling=True)

layer_conv3, weights_conv3 = \
    new_conv_layer(input=layer_conv2,
                   num_input_channels=num_filters2,
                   filter_size=filter_size3,
                   num_filters=num_filters3,
                   use_pooling=True)
layer_conv2
layer_conv3

layer_flat, num_features = flatten_layer(layer_conv3)

layer_fc1 = new_fc_layer(input=layer_flat,
                         num_inputs=num_features,
                         num_outputs=fc_size,
                         use_relu=True)
layer_fc2 = new_fc_layer(input=layer_fc1,
                         num_inputs=fc_size,
                         num_outputs=num_classes,
                         use_relu=False)



y_pred = tf.nn.softmax(layer_fc2)
```

```
y_pred_cls = tf.argmax(y_pred, dimension=1)



cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=layer_fc2,
                                                        labels=y_true)

cost = tf.reduce_mean(cross_entropy)
optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=1e-4).minimize(cost)

correct_prediction = tf.equal(y_pred_cls, y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
session =  tf.compat.v1.Session()
session.run(tf.initialize_all_variables())
train_batch_size = batch_size

def print_progress(epoch, feed_dict_train, feed_dict_validate, val_loss):

    acc = session.run(accuracy, feed_dict=feed_dict_train)
    val_acc = session.run(accuracy, feed_dict=feed_dict_validate)
    msg = "Epoch {0} --- Training Accuracy: {1:>6.1%}, Validation Accuracy: {2:>6.1%}, Validation Loss: {3:.3f}"
    print(msg.format(epoch + 1, acc, val_acc, val_loss))
total_iterations = 0

def optimize(num_iterations):
    global total_iterations
    start_time = time.time()
    best_val_loss = float("inf")
```

```
    patience = 0

    for i in range(total_iterations,
                   total_iterations + num_iterations):
        x_batch, y_true_batch, _, cls_batch = data.train.next_batch(train_batch_size)
        x_valid_batch, y_valid_batch, _, valid_cls_batch =
data.valid.next_batch(train_batch_size)
        x_batch = x_batch.reshape(train_batch_size, img_size_flat)
        x_valid_batch = x_valid_batch.reshape(train_batch_size, img_size_flat)



        feed_dict_train = {x: x_batch,
                           y_true: y_true_batch}

        feed_dict_validate = {x: x_valid_batch,
                              y_true: y_valid_batch}
        session.run(optimizer, feed_dict=feed_dict_train)
        if i % int(data.train.num_examples/batch_size) == 0:
            val_loss = session.run(cost, feed_dict=feed_dict_validate)
            epoch = int(i / int(data.train.num_examples/batch_size))

            print_progress(epoch, feed_dict_train, feed_dict_validate, val_loss)

            if early_stopping:
                if val_loss < best_val_loss:
                    best_val_loss = val_loss
                    patience = 0
                else:
                    patience += 1
```

```
            if patience == early_stopping:
                break
    total_iterations += num_iterations
    end_time = time.time()
    time_dif = end_time - start_time
    print("Time elapsed: " + str(timedelta(seconds=int(round(time_dif)))))
print(total_iterations)
##Helper-function to plot example errors
def plot_example_errors(cls_pred, correct):
    incorrect = (correct == False)
    # Get the images from the test-set that have been
    # incorrectly classified.
    images = data.valid.images[incorrect]

    # Get the predicted classes for those images.
    cls_pred = cls_pred[incorrect]

    # Get the true classes for those images.
    cls_true = data.valid.cls[incorrect]

    # Plot the first 9 images.
##    plot_images(images=images[0:9],
##                cls_true=cls_true[0:9],
##                cls_pred=cls_pred[0:9])
def plot_confusion_matrix(cls_pred):
    cls_true = data.valid.cls

    # Get the confusion matrix using sklearn.
```

```
    cm = confusion_matrix(y_true=cls_true,
                          y_pred=cls_pred)

    # Print the confusion matrix as text.
    print(cm)
    # Plot the confusion matrix as an image.
    plt.matshow(cm)

    # Make various adjustments to the plot.
    plt.colorbar()
    tick_marks = np.arange(num_classes)
    plt.xticks(tick_marks, range(num_classes))
    plt.yticks(tick_marks, range(num_classes))
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

def print_validation_accuracy(show_example_errors=False,
                              show_confusion_matrix=False):
    num_test = len(data.valid.images)
    cls_pred = np.zeros(shape=num_test, dtype=np.int)
    i = 0

    while i < num_test:
        j = min(i + batch_size, num_test)
       images = data.valid.images[i:j, :].reshape(batch_size, img_size_flat)
       labels = data.valid.labels[i:j, :]
      feed_dict = {x: images,
                   y_true: labels}
```

```
    cls_pred[i:j] = session.run(y_pred_cls, feed_dict=feed_dict)

    # Set the start-index for the next batch to the
    # end-index of the current batch.
    i = j

cls_true = np.array(data.valid.cls)
cls_pred = np.array([classes[x] for x in cls_pred])

# Create a boolean array whether each image is correctly classified.
correct = (cls_true == cls_pred)
correct_sum = correct.sum()
acc = float(correct_sum) / num_test

# Print the accuracy.
msg = "Accuracy on Validation-Set: {0:.1%}"
print(msg.format(acc, correct_sum, num_test))

# Plot some examples of mis-classifications, if desired.
#  if show_example_errors:
# print("Example errors:")
#plot_example_errors(cls_pred=cls_pred, correct=correct)

# Plot the confusion matrix, if desired.
if show_confusion_matrix:
    print("Confusion Matrix on Validation")
    plot_confusion_matrix(cls_pred=cls_pred)
```

```
optimize(num_iterations=400)  # We performed 100 iterations above.
#print_validation_accuracy(show_example_errors=True)
print_validation_accuracy(show_example_errors=True, show_confusion_matrix=True)

def plot_conv_weights(weights, input_channel=0):

    w = session.run(weights)
    w_min = np.min(w)
    w_max = np.max(w)
    num_filters = w.shape[3]
    num_grids = math.ceil(math.sqrt(num_filters))
    fig, axes = plt.subplots(num_grids, num_grids)
    for i, ax in enumerate(axes.flat):

        if i<num_filters:

            img = w[:, :, input_channel, i]
            ax.imshow(img, vmin=w_min, vmax=w_max,
                      interpolation='nearest', cmap='seismic')



        ax.set_xticks([])
        ax.set_yticks([])
  plt.show()

def plot_conv_layer(layer, image):
   image = image.reshape(img_size_flat)
   feed_dict = {x: [image]}
```

```
    values = session.run(layer, feed_dict=feed_dict)
    num_filters = values.shape[3]
    num_grids = math.ceil(math.sqrt(num_filters))
    fig, axes = plt.subplots(num_grids, num_grids)
    for i, ax in enumerate(axes.flat):

        if i<num_filters:

            img = values[0, :, :, i]
            ax.imshow(img, interpolation='nearest', cmap='binary')
        ax.set_xticks([])
        ax.set_yticks([])
    plt.show()


def plot_image(image):
    plt.imshow(image.reshape(img_size, img_size, num_channels),
               interpolation='nearest')
    plt.show()
    '''plot_conv_weights(weights=weights_conv1)

    plot_conv_layer(layer=layer_conv1, image=image)
    plot_conv_weights(weights=weights_conv2, input_channel=0)
    plot_conv_layer(layer=layer_conv2, image=image)
    #session.close()
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()'''
```

```
classes = { 0:'Pneumonia',
            1:'Normal',
            2:'Covid'}

top=tk.Tk()
top.geometry('800x600')
top.title('PNEUMONIA AND COVID DETECTION')
top.configure(background='light blue')

label=Label(top,background='light blue', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image =cv2.imread(file_path)
    cv2.imshow("frame",image)
    image= cv2.resize(image, (img_size, img_size), cv2.INTER_LINEAR) / 255
    feed_dict_test = {
        x: image.reshape(1, img_size_flat),
        y_true: np.array([[2,1,0]])
    }

    test_pred = session.run(y_pred_cls, feed_dict=feed_dict_test)
    sign=classes[test_pred[0]]
    print(sign)
    label.configure(foreground='#011638', text=sign)
    image1 = test_images[0]
    plot_image(image1)
```

```
def show_classify_button(file_path):
    classify_b=Button(top,text="Classify X-ray Image",command=lambda:
classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156',
foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)

def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)

        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass

upload=Button(top,text="Upload an X-ray
image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))

upload.pack(side=BOTTOM,pady=50)
```

```
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="PNEUMONIA AND COVID DETECTION",pady=20, font=('arial',20,'bold'))
heading.configure(background='light blue',foreground='black')
heading.pack()
top.mainloop()
```

CHAPTER 7

SYSTEM TESTING

When considering the strategy of Deep Learning testing, think accuracy and efficiency as a primary goal in the quality assurance. Benefits such as detecting redundant unsuccessful tests, and keeping untested code out of production, prediction, and prevention ultimately reduce much of the risk in the deployment phase. Some of the critical contributions quality assurance include, Defect alerts Enhanced analytics Faster predictions Improved optimization Cleaner traceability Real-time feedback and the sooner can implement an in-house AI platform to assist in application testing; will discover a more accurate and efficient deployment with reduced effort.

The 3 classes with which the model is tested are,

1.NORMAL

2.PNEUMONIA

3.COVID

7.1 Test cases

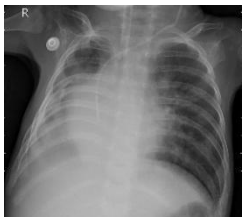
Test case:1 Normal

S.NO	Input x-ray image	Classified result	Expected output	Actual output	Test case Pass/Fail
1.		Normal	Normal	Normal	Pass
2.		Normal	Normal	Normal	Pass

3.		Normal	Normal	Normal	Pass

7.1(a) Test cases of Normal class

Test case:2 Pneumonia

S.NO	Input x-ray image	Classified result	Expected output	Actual output	Test case Pass/Fail
1.		Pneumonia	Pneumonia	Pneumonia	Pass
2.		Pneumonia	Pneumonia	Pneumonia	Pass
3.		Pneumonia	Pneumonia	Pneumonia	Pass

7.1(b) Test cases of Pneumonia class

Test case:3 Covid

S.NO	Input x-ray image	Classified result	Expected output	Actual output	Test case Pass/Fail
1.		Covid	Covid	Covid	Pass
2.		Covid	Covid	Covid	Pass
3.		Covid	Covid	Covid	Pass

7.1(c) Test cases of COVID class

CHAPTER 8

8.1 Conclusion

The risk of pneumonia and it's severity in the form of COVID-19 are immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. The proposed system can detect and predict pneumonia and COVID-19 diseases at an early stage based on chest X-ray images. The study employs a flexible and efficient approach of deep learning by applying the model of CNN in predicting and detecting a patient's unaffected and affected lungs with the disease , employing a chest X-ray image. The models presented could achieve 90% accuracy and 100% of validation accuracy. High validation accuracy will ensure that the number of false-negative instances is lower, hence lowers the risk to the patient's life. Thus, it is concluded that CNN classifier therefore, be effectively used for early detection of pneumonia and COVID-19 in children as well as adults. A large number of X-ray images can be processed very quickly to provide highly precise diagnostic results, thus helping health care systems provide efficient patient care services and reduce mortality rates.

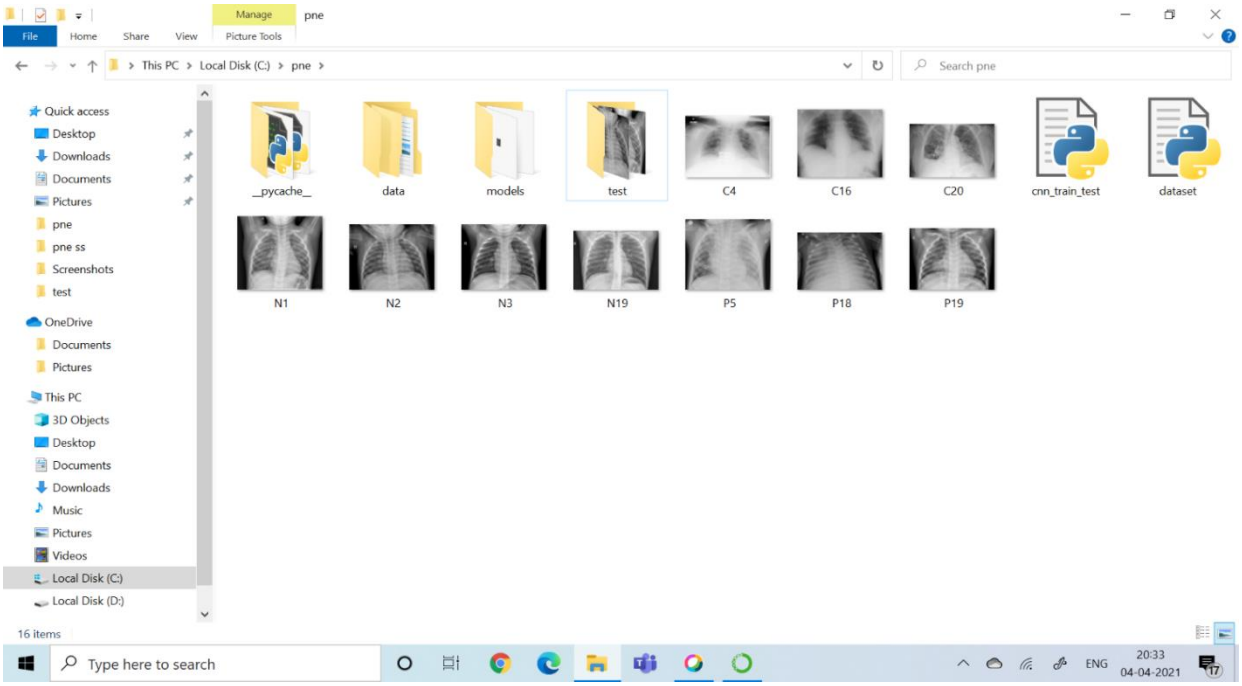
8.2 Future Enhancement

In future, the further classifications of pneumonia can be detected along with pneumonia and COVID-19 with much greater accuracy. This work could also be extended to detect and classify X-ray images consisting of lung cancer and pneumonia. Distinguishing X-ray images that contain lung cancer and pneumonia has been a big issue in recent times, and our next approach should be to tackle this problem. And also the severity of the COVID-19 can also be predicted.

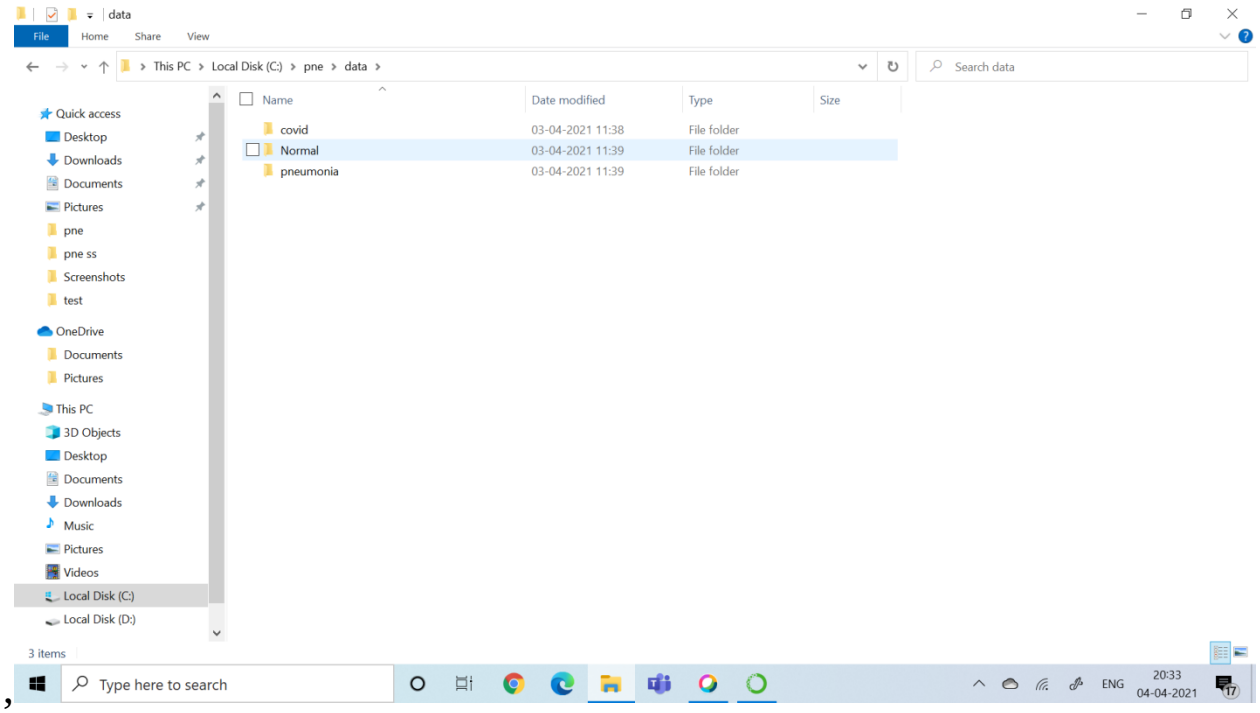
APPENDICES

A.1 Sample Screenshots

The below picture contain all the folders to run the complete project.

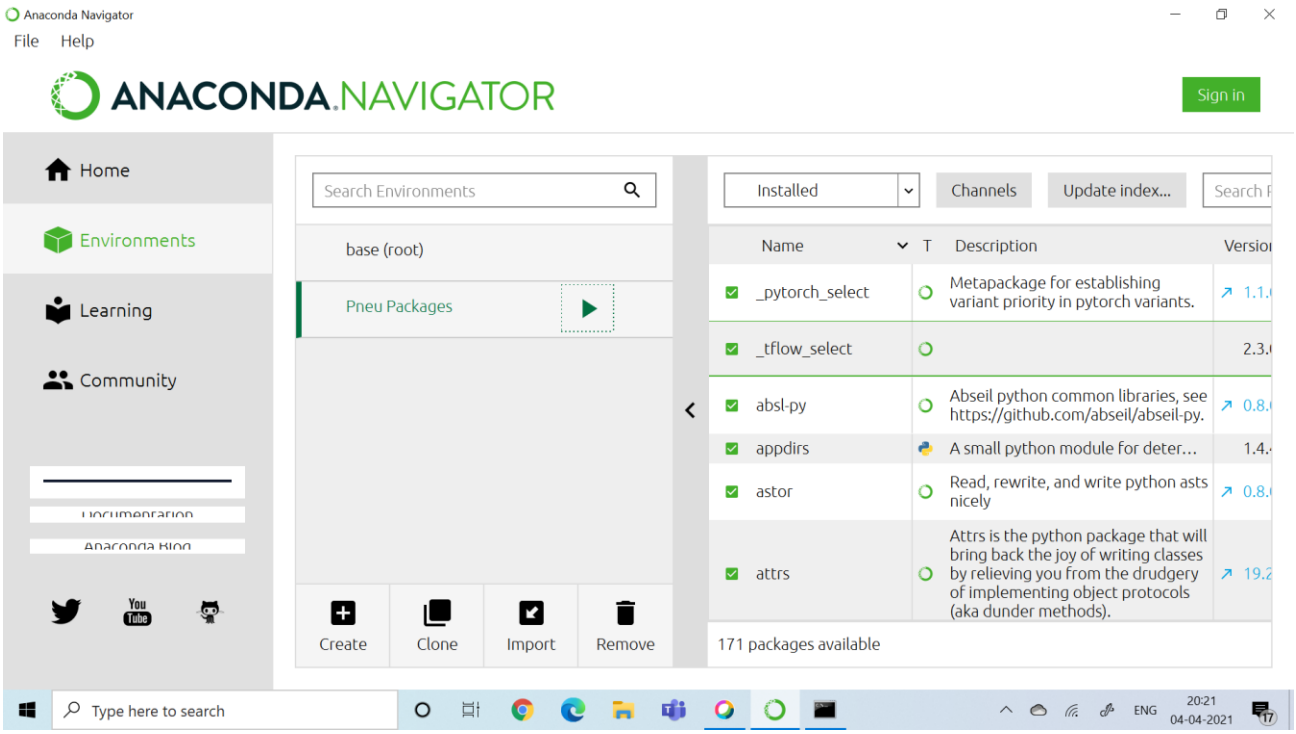


The three classes in the train folder are shown in the screenshot below

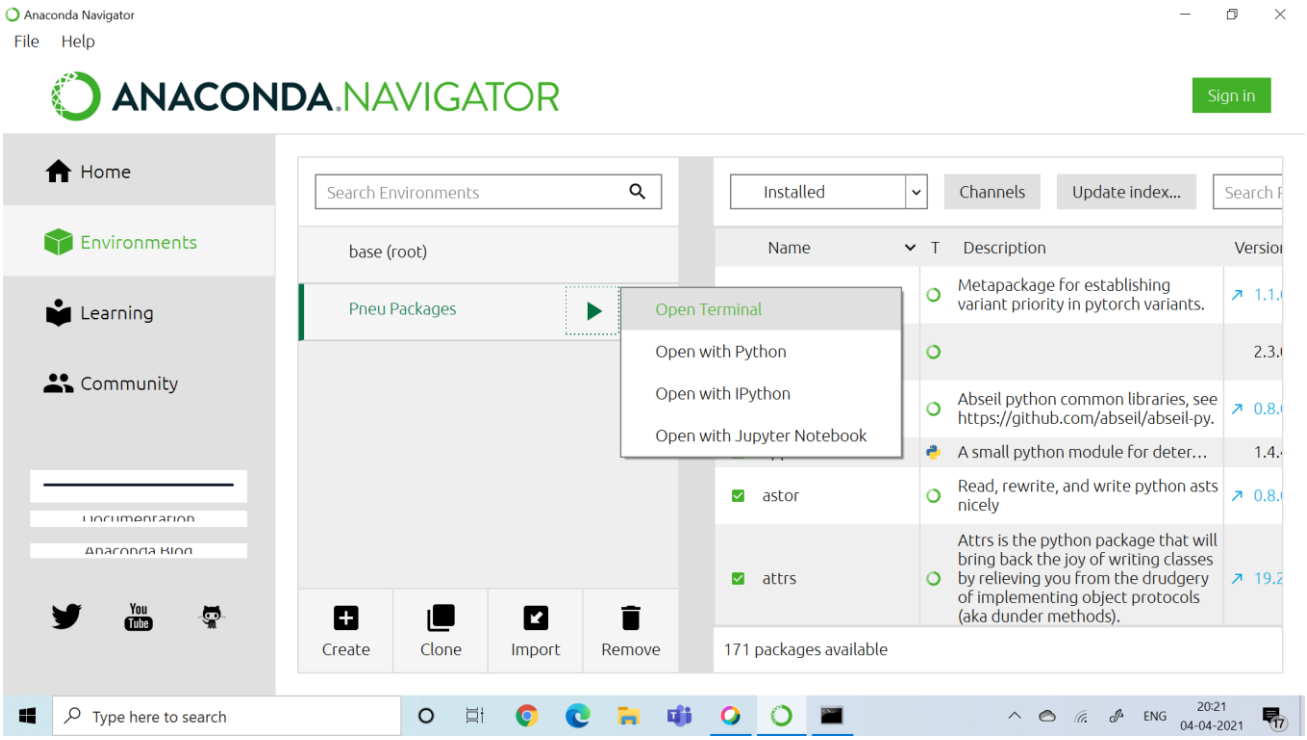


,

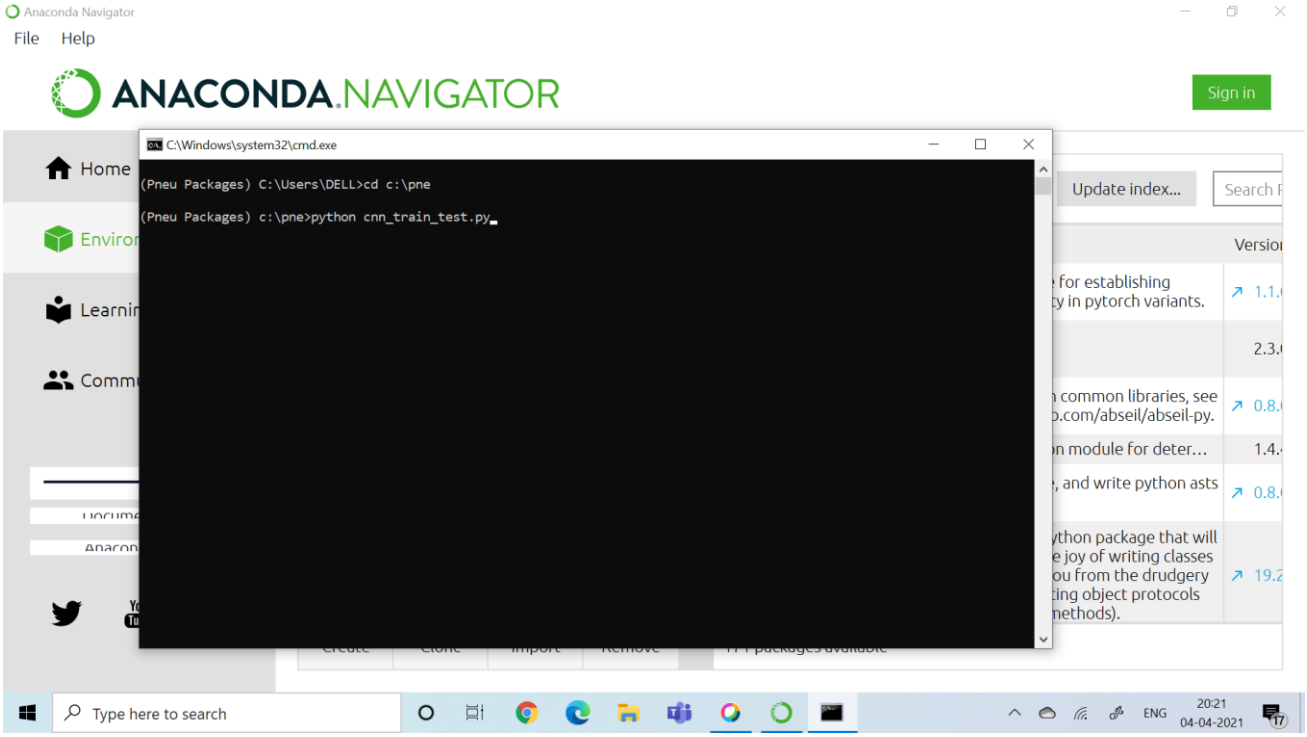
Open the Anaconda Navigator with the newly created environment.



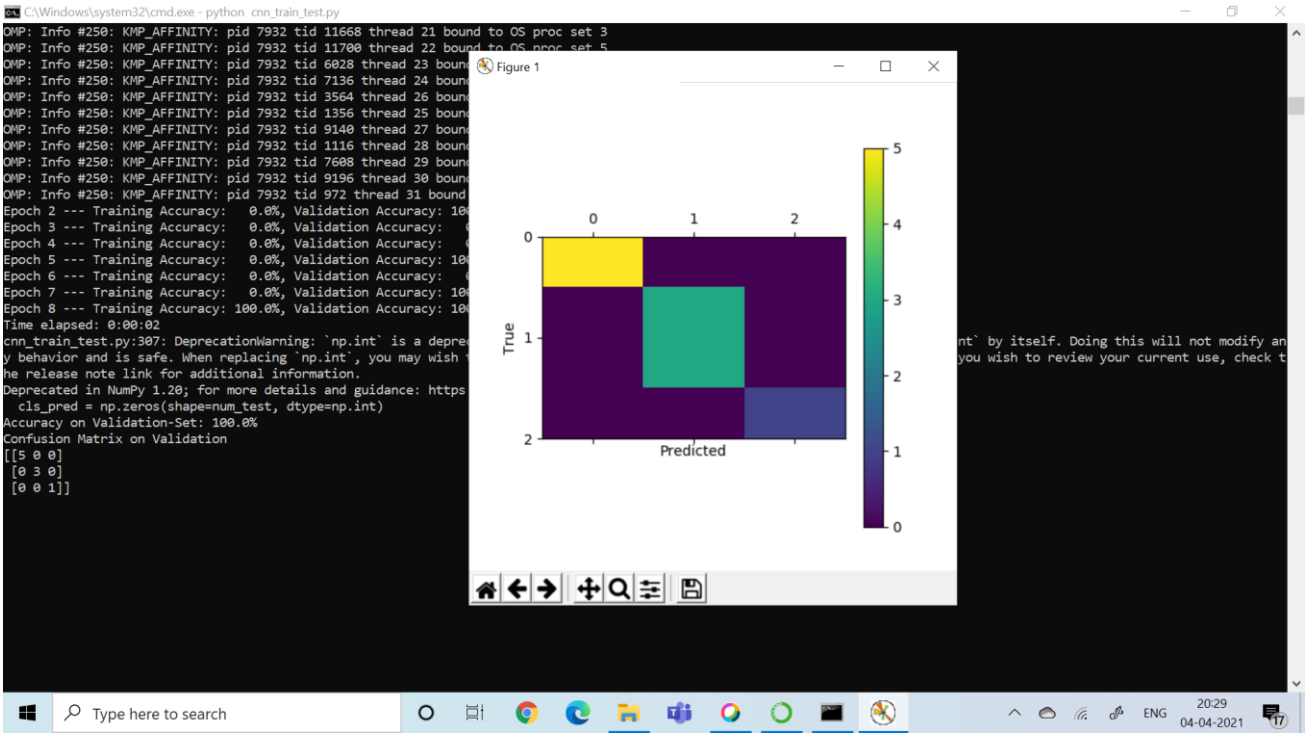
Open the terminal with the environment where there are necessary packages installed.



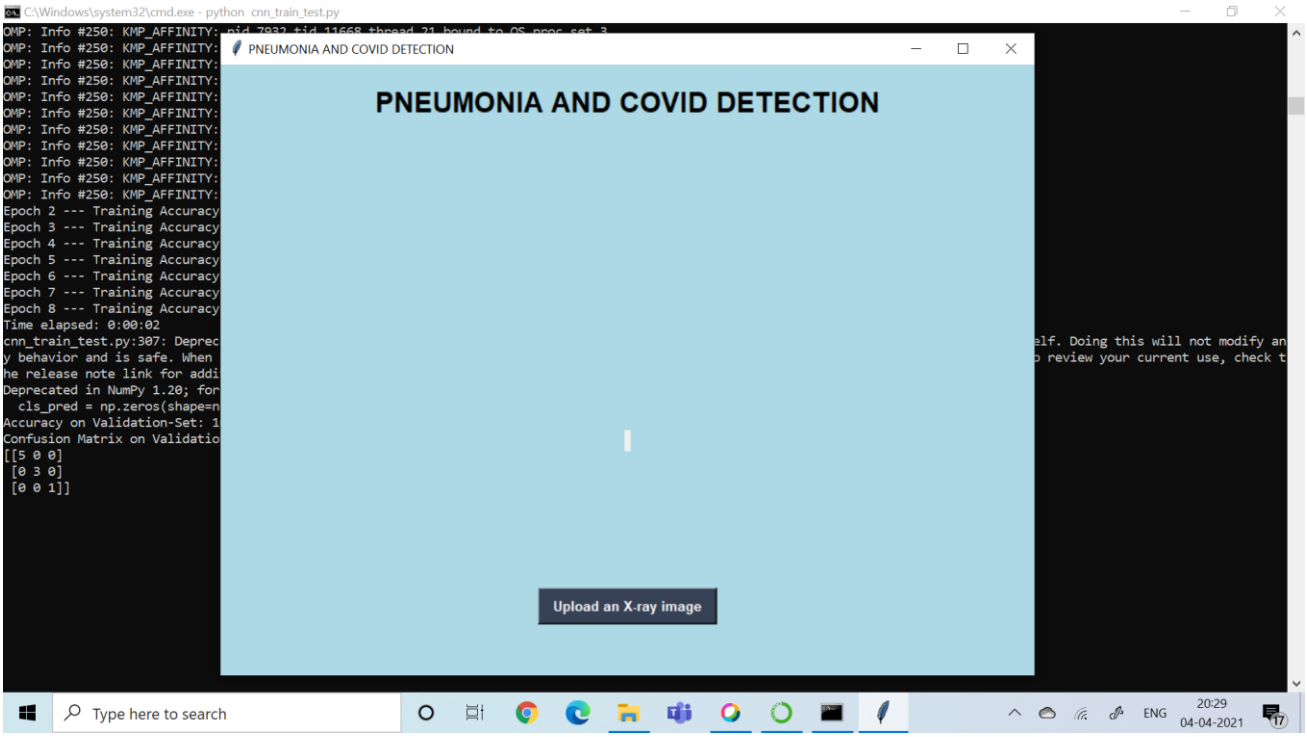
Move to the respective drive and run the code.



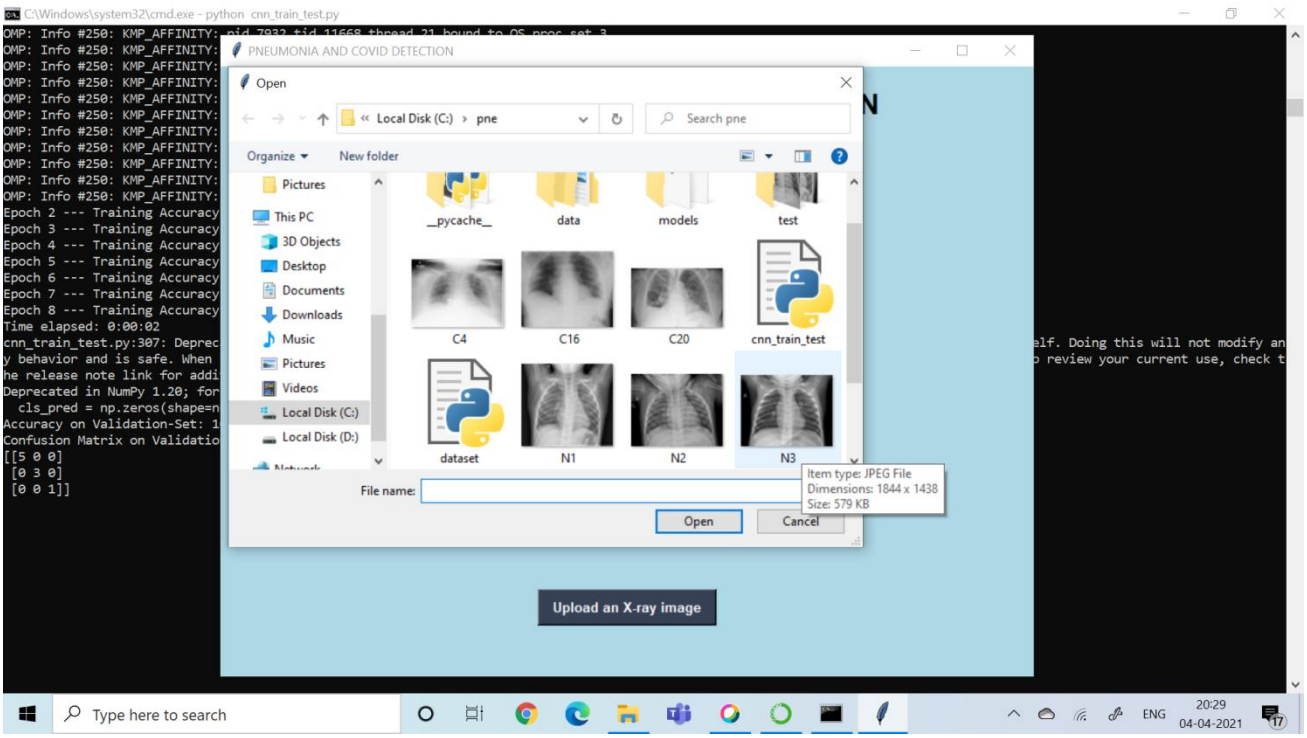
There appears the confusion matrix and accuracy on the validation set.



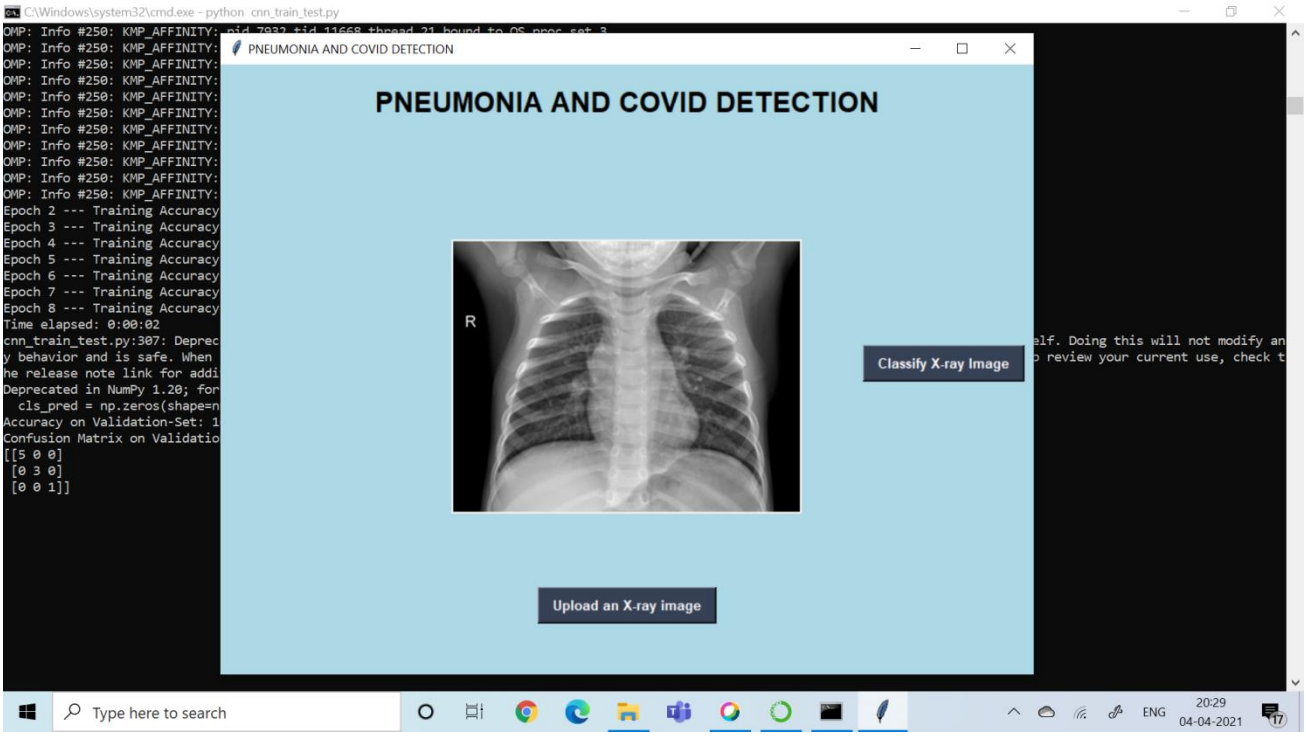
On closing the pictorial view of confusion matrix, there appears a gui where the x-ray images can be uploaded.



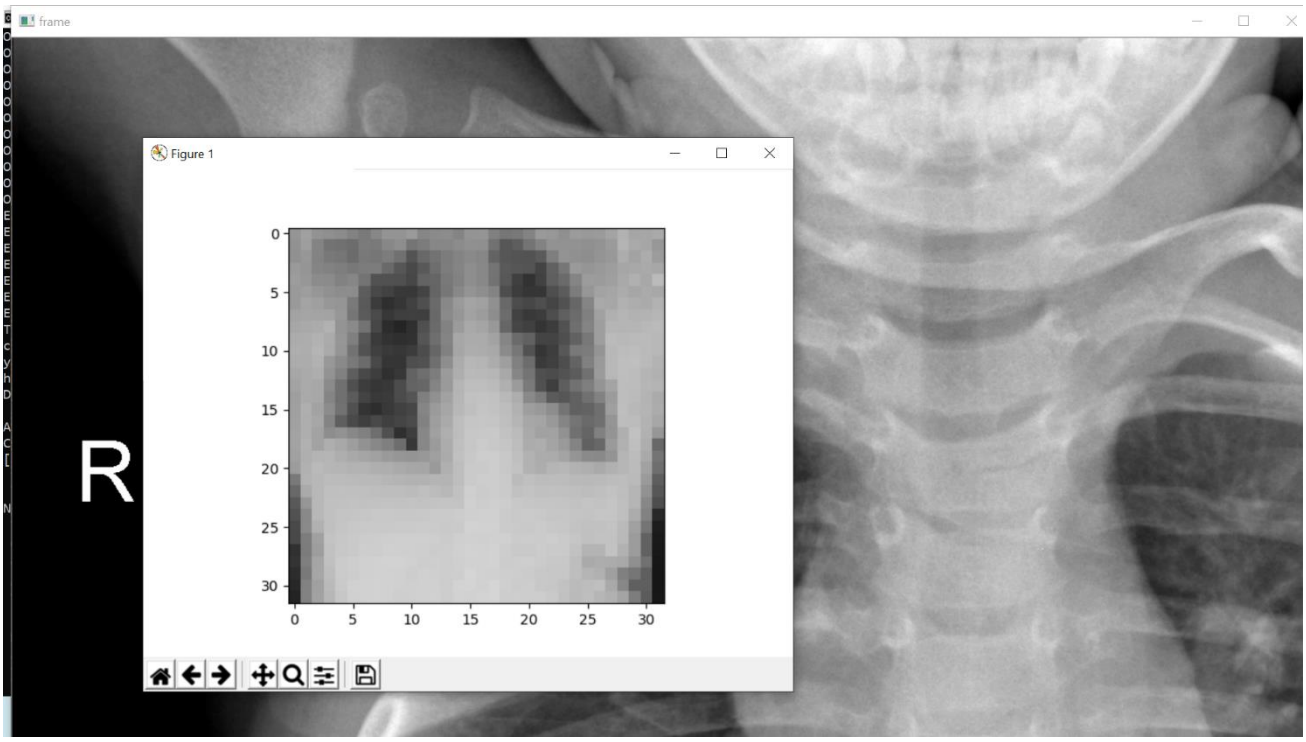
Initially upload the normal x-ray image,



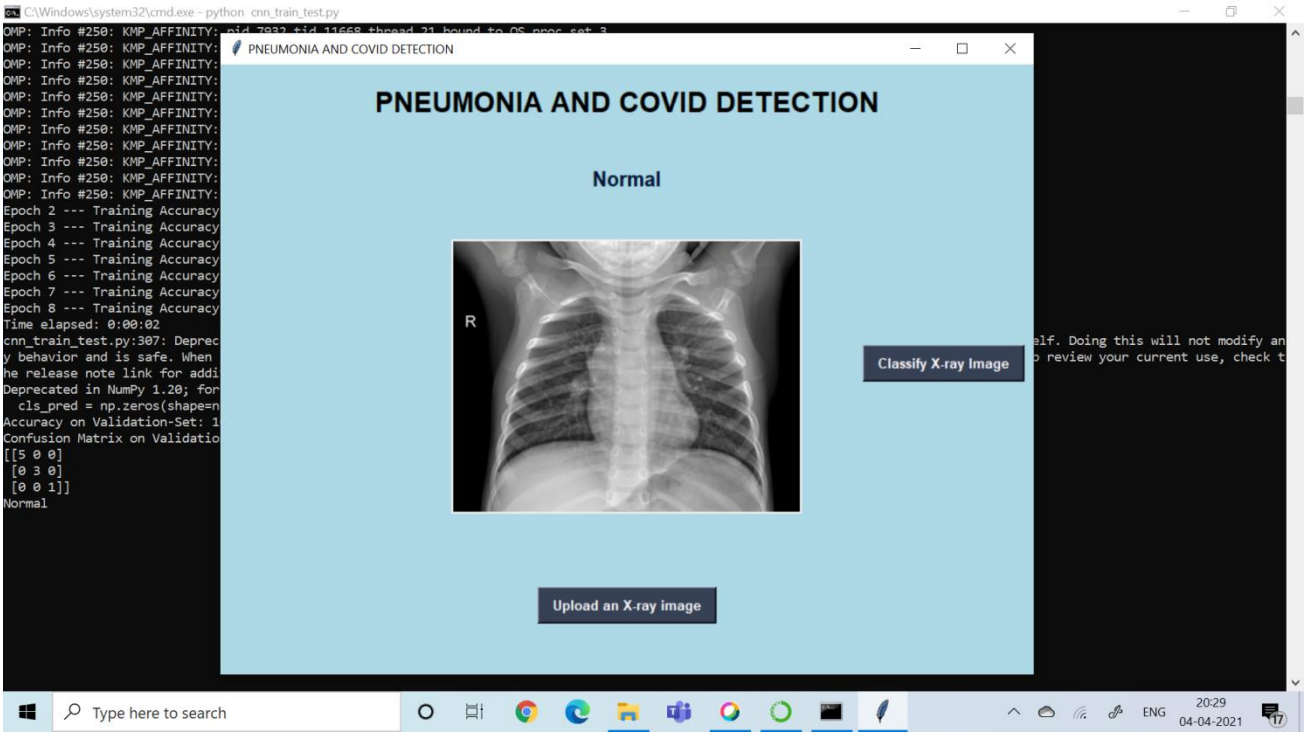
The selected image appears on the gui,



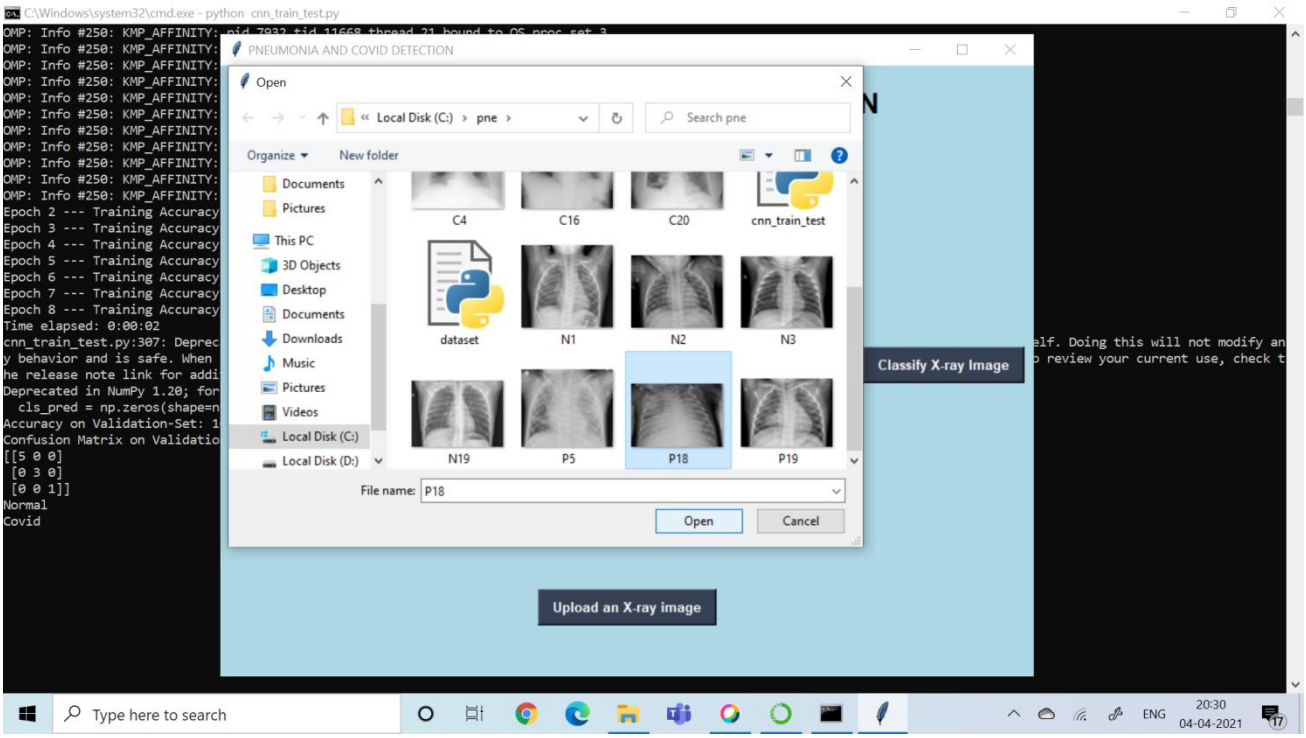
On clicking classify x-ray image, the frame with the input image and its segmentation opens up.

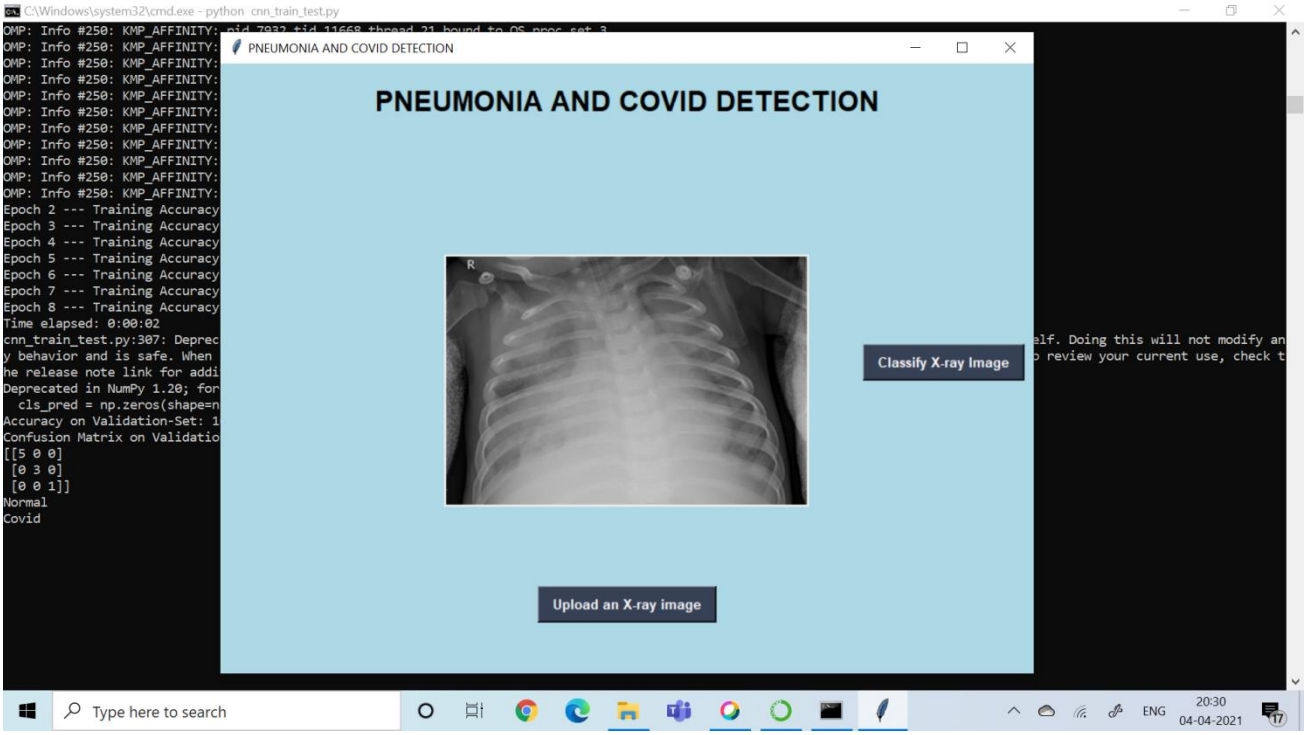


Further,the image gets classified to be **NORMAL**.

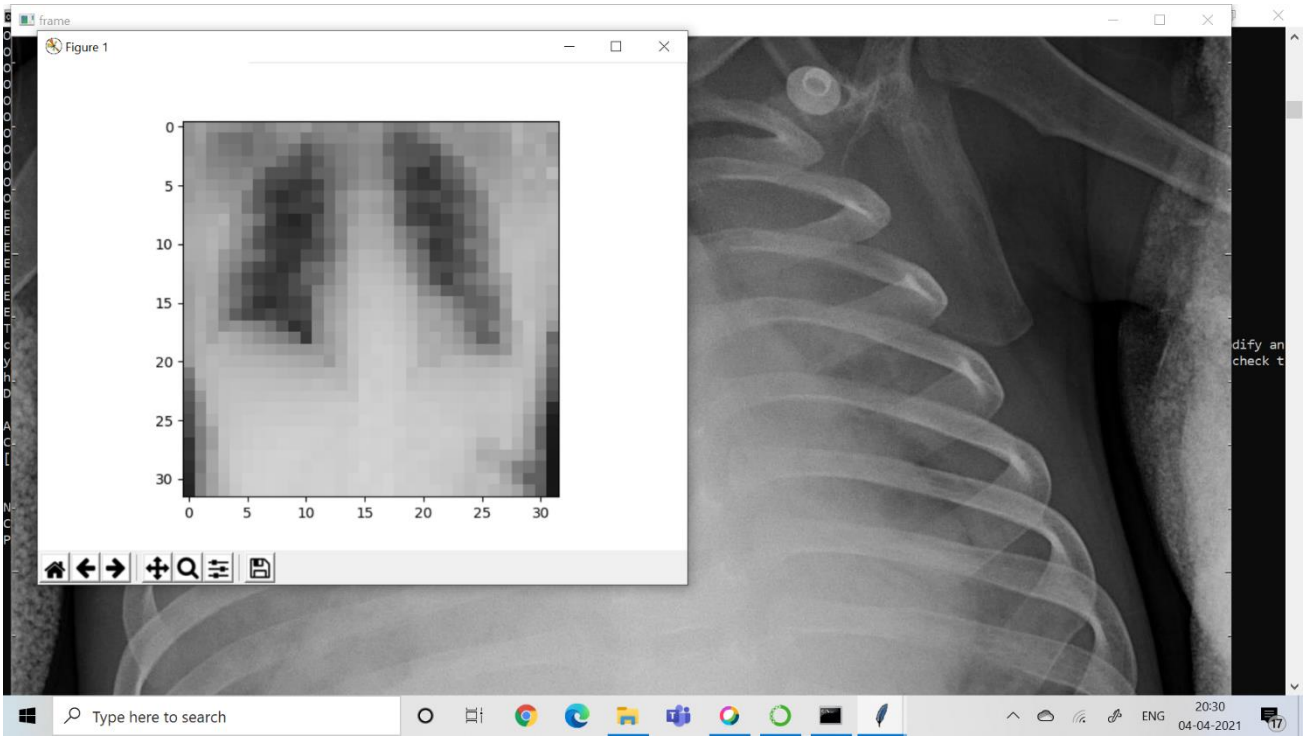


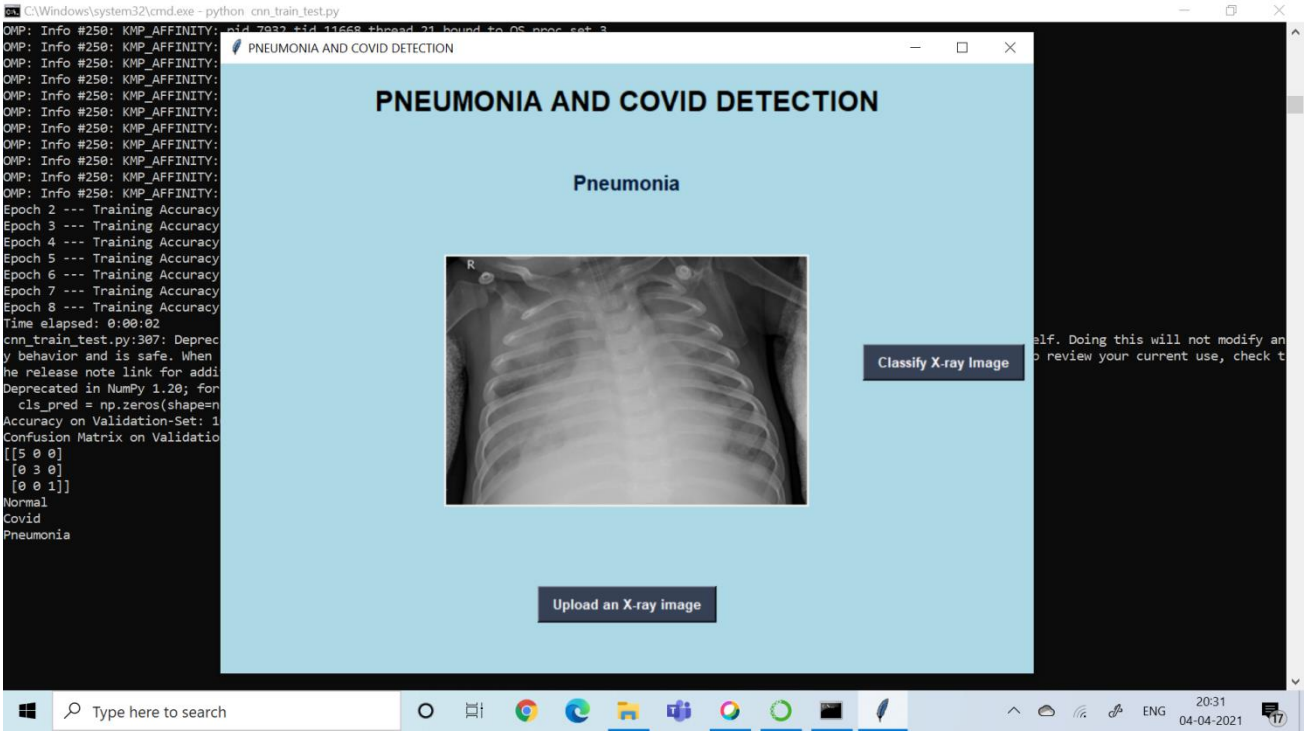
Similarly, pneumonia affected x-ray image is uploaded.



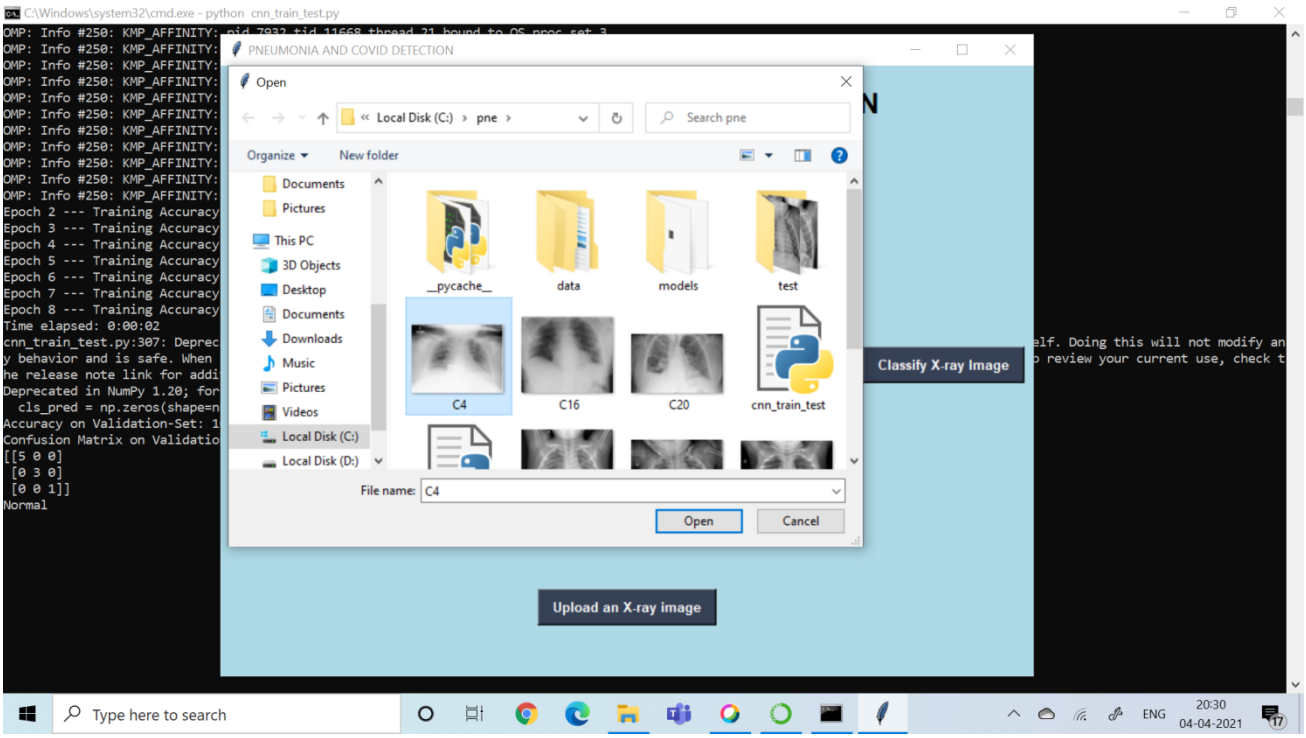


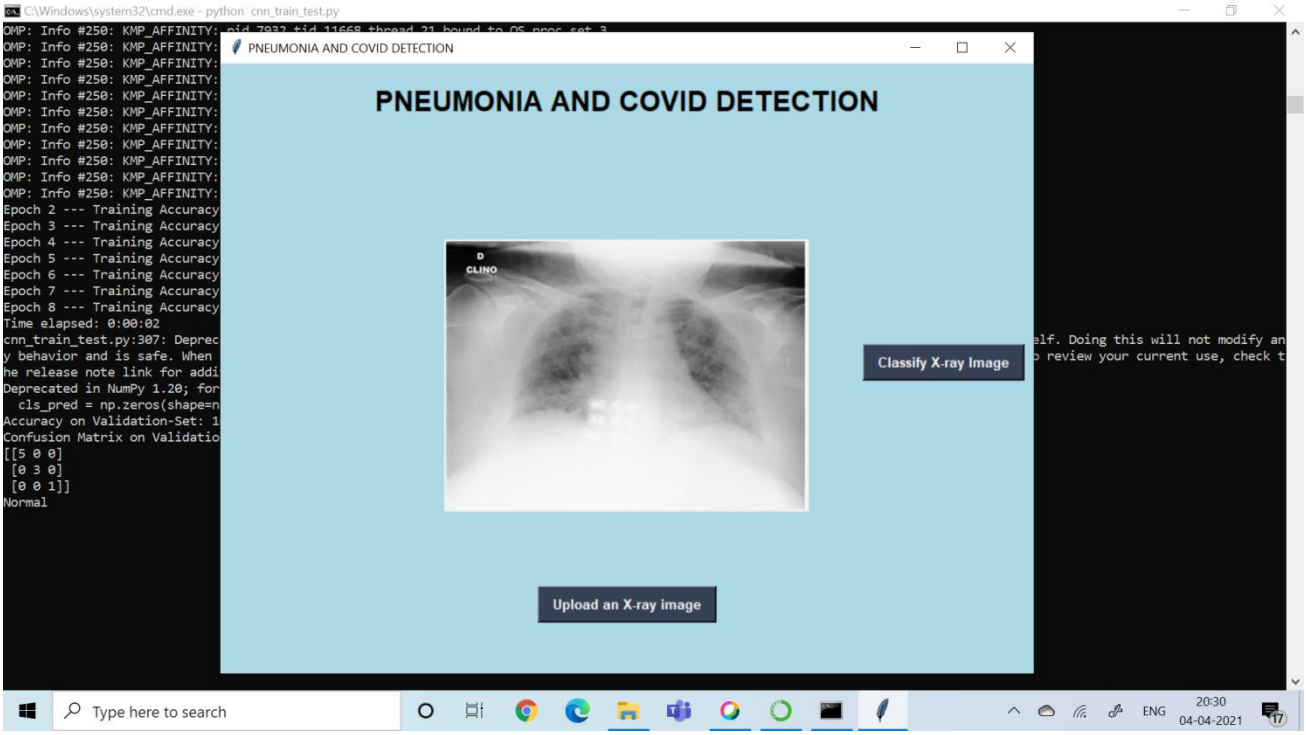
The image gets classified to be **PNEUMONIA**.



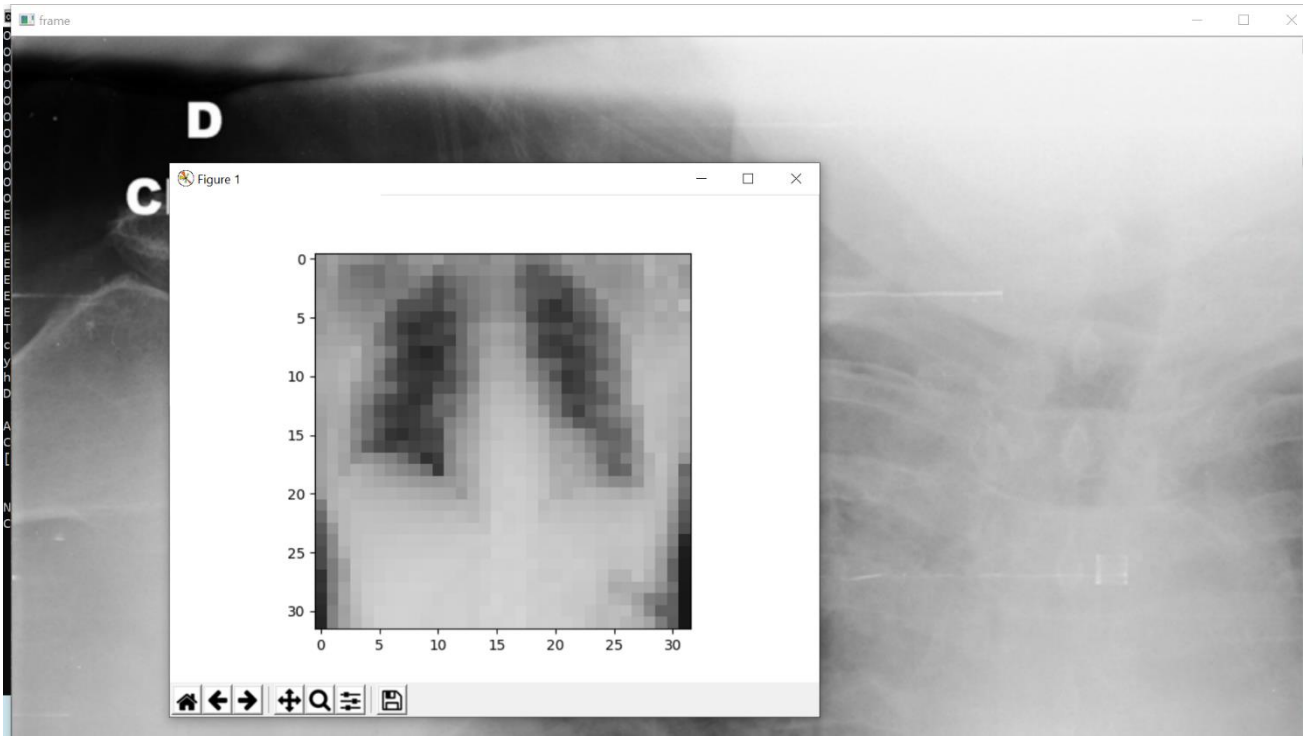


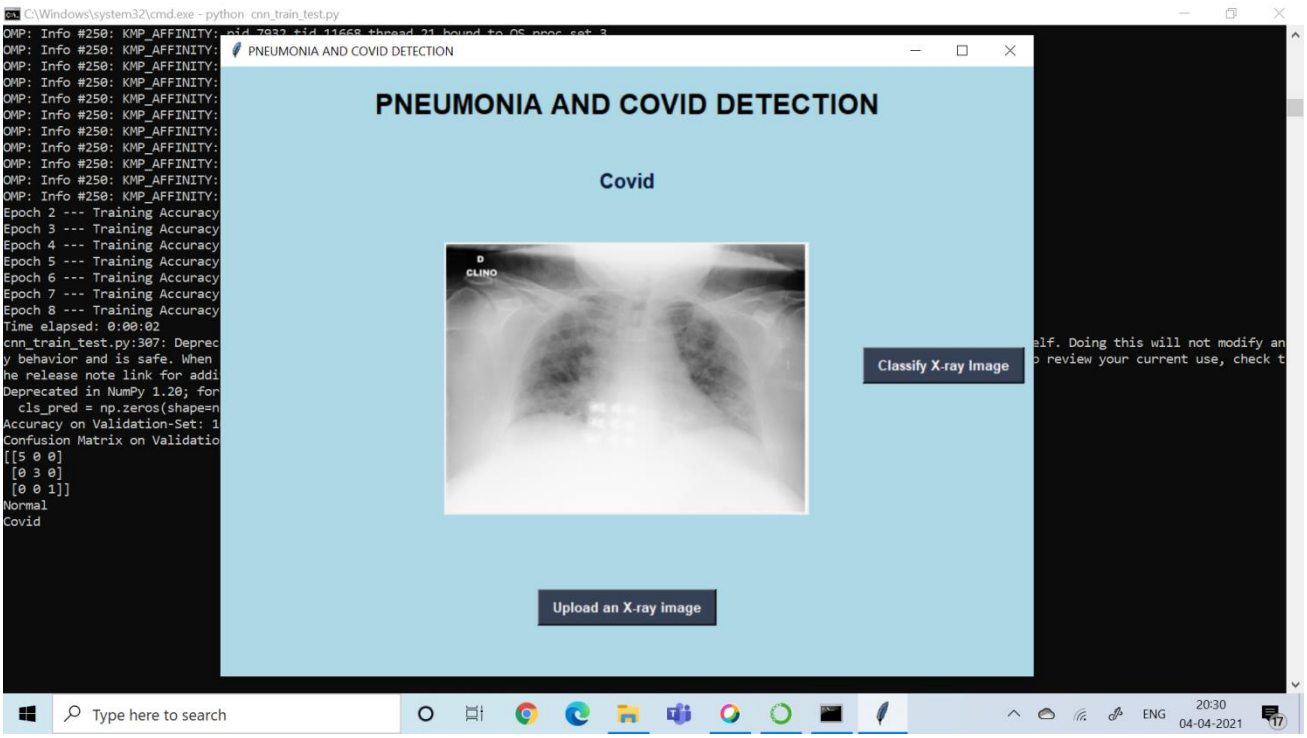
Now, the covid affected x-ray image is uploaded.



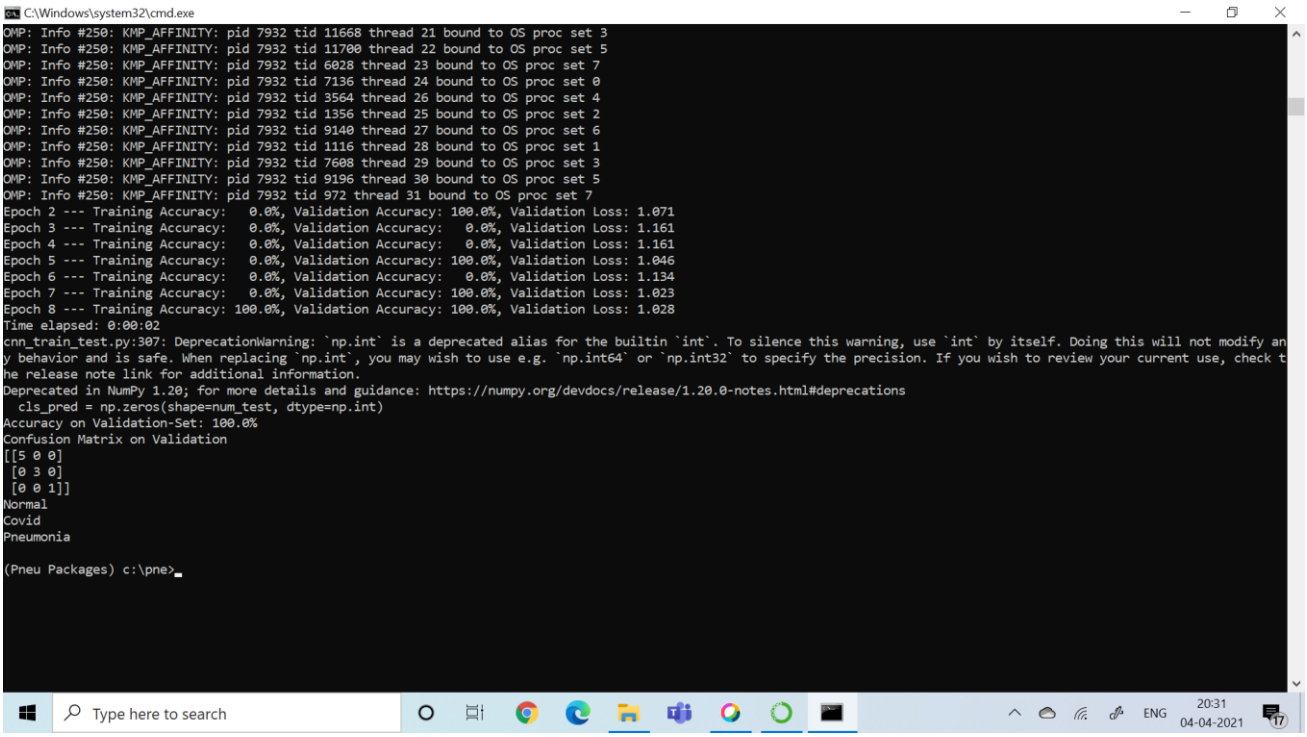


The result shows that the input image is classified to be **COVID**.





The complete classification is performed.



A2. PUBLICATIONS

Journal name- Journal of Emerging Technologies and Innovative Research.

Paper title- Diagnosis of Pneumonia from X-rays using Deep learning.

Publication issue - Volume 8, Issue 2, May – 2021

DIAGNOSIS OF PNEUMONIA FROM X-RAYS USING DEEP LEARNING

V. Sathya Preiya, Pavithra D, Pooja K,Priyadharscini S
NAssociateProfessor[a], Students[a]
[a]DepartmentofComputerScienceEngineering,
[a]PanimalarEngineeringCollege,Chennai,T
amilnadu,India

Abstract –Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (purulent material), causing cough with phlegm or pus, fever, chills, and difficulty breathing. A variety of organisms, including bacteria, viruses and fungi, can cause pneumonia. It is an infection of the lungs with a range of possible causes. It can be a serious and life-threatening disease. It normally starts with a bacterial, viral, or fungal infection. The lungs become inflamed, and the tiny air sacs, or alveoli, inside the lungs fill up with fluid. Over 150 million people get infected with pneumonia on an annual basis especially children under 5 years old. COVID-19 pneumonia is a serious illness that can be deadly. Early detection of Pneumonia and COVID-19 is crucial in reducing mortality. The rich collection of annotated datasets piloted the robustness of deep learning techniques to effectuate the implementation of diverse medical imaging tasks. This proposed system involves detection of Pneumonia and COVID-19 based on deep learning which is proposed for thoracic X-Ray images.

Keywords - Pneumonia, X-rays, Convolutional neural network, deep learning.

I. INTRODUCTION

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus pneumoniae. Some of its symptoms appear suddenly and may include chest pain and difficulty breathing, a high fever, shaking chills, excessive sweating, fatigue, and a cough with phlegm that persists or gets worse. Another infectious illness COVID-19, also known as Severe Acute Respiratory Syndrome Corona virus-2 is a contagious disease that is released from tiny droplets containing saliva or mucus from respiratory system of a diseased person who talks, sneeze, or cough. It spreads rapidly through close contact with somebody who is infected or tapping or holding a virus contaminated objects and also the surfaces.

Older adults and people who have severe underlying medical conditions or prior cases of pneumonia seem to be at higher risk for developing more serious complications from the virus. With rising deaths and limited medical resources, doctors and medical professionals around the world are working around the clock to treat patients and prevent the spread of the virus.

It is crucial to have quick and accurate detection of pneumonia so patients can receive treatment in a timely manner especially in impoverished regions. With the growing technological advancements, it is possible to use tools based on deep learning frameworks to detect pneumonia and COVID-19 based on chest x-ray images.

The successes of deep learning algorithms in analyzing medical images have lead Convolutional Neural Networks (CNNs) to gain much attention for disease classification. The progression of Deep Learning contributes to aid in the decision-making process of experts to diagnose patients with pneumonia and COVID-19.

The study employs a flexible and efficient approach of deep learning by applying the model of CNN in predicting and detecting a patient's unaffected and affected lungs with the disease, employing a chest X-ray image. The trained-model produced an accuracy rate of 95% during the training. The proposed system can detect and predict pneumonia and COVID-19 diseases based on chest X-ray images.

II. RELATEDWORKS

Timely detection of pneumonia in children can help to fast-track the process of recovery. The convolutional neural network models to accurately detect pneumonic lungs from chest X-rays, which can be utilized in the real world by medical practitioners to treat pneumonia. CNN models have been created from scratch and trained on Chest X-Ray Images dataset on Kaggle. Keras neural network library with Tensor Flow backend has been used to implement the models. Adam optimizer function was finalized to be used for all classifiers. The models presented at best could achieve 92.31% accuracy.

Evolutionary algorithm for searching neural architectures under multiple objectives, such as classification performance and floating point operations (FLOPs). By populating a set of architectures to approximate the entire Pareto frontier through genetic operations that recombine and modify architectural components progressively the proposed method overcomes the problem where obtained architectures are either solely optimized for classification performance, or only for one deployment scenario. Analysis towards validating the generalization and robustness aspects of the obtained

architectures is also provided along with an application to common thorax disease classification on human chest X-rays.

One of the primary clinical observations for screening the novel corona virus is capturing a chest x-ray image. In most patients, a chest x-ray contains abnormalities, such as consolidation, resulting from COVID-19 viral pneumonia. In this paper, numerous chest x-ray images from various sources are collected, and the largest publicly accessible dataset is prepared. Using the transfer learning paradigm, the well-known CheXNet model is utilized to develop COVID-CXNet. At first, a base Convolutional model is designed and trained on different portions of the dataset. Then, pretrained models based on the ImageNet dataset are discussed. Finally, a pretrained model on a similar image type is explained.

The research work mainly proposes a Convolutional neural system (CNN) model prepared without any preparation to group and identifies the occurrence of pneumonia disease from a given assortment of chest X-ray image tests. The Data Augmentation techniques have helped to perform various types of operations in the images. Keras which is an open-source neural network

library in deep learning having tensor flow backend is used. Some of the research challenges include lack of homogeneity in the architectures, frameworks and device is and lack of image datasets and information leads to inaccurate results.

During diagnosis, expert radiologists corresponds white spots on the image to infiltrates identifying an infection, and white areas to the pneumonia fluid in the lungs. However, the limited color scheme of x-ray images consisting of shades of black and white, cause drawbacks when it comes to determining whether there is an infected area in the lungs or not. YOLO and SSD algorithms might be effective for the localization of the pneumonia region, while different pre-processing methods may be needed for training the respective algorithms which is the disadvantage.

II.PROPOSEDSYSTEM

Chest X-rays are one of the best methods for the detection of pneumonia. X-ray imaging is preferred over CT imaging because X-ray imaging typically takes considerably less time than CT imaging. X-rays are the most common and widely available diagnostic imaging technique, playing a crucial role in clinical care and epidemiological studies. The proposed methodology uses a deep transfer learning algorithm using CNN that extracts the features from the X-ray image that describes the presence of disease automatically and reports whether it is a case of pneumonia or COVID-19 or normal.

III. Modules:

1. Exploring the Dataset

The dataset that will be used for this project will be the Chest X-Ray Images from Kaggle. The dataset consists of training data, validation data, and testing data. The training data consists of 144 chest x-ray images with 50 images shown to have pneumonia, 50 images shown to be COVID and 44 images shown to be normal. The testing data consists of 20% of images in training images.

2. Pre-processing

The X-Ray image of a person is fed as the input to the pre-processor. If the images are of low resolution or poor contrast, the pre-processor will

enhance the contrast to get the accurate classification type. It enhances the better visualization of the image.

Steps involved are as follows:

- Read the image
- Resize the image
- Remove noise
- Segmentation.

Read the image -Reading an image

Resize the image -Some images vary in size, so to establish a base size for all image compression or resizing is done

Remove the noise -In order to smooth our image,the unwanted noise has to be removed.

Segmentation - Separating the leisen from the image is segmentation.

DWT:

It is the lossless image compression technique use in preprocessing .High quality images that require large storage are to be compressed. So DWT is required to compress the image without any appreciable loss of information. Digitize the source image into signal. Decompose signal to wavelet (sub bands) LL, LH, HL, HH. DWT retains images from LL to produce next level of decomposition, because the low frequency images have finer frequency and time resolution than high frequency images.DWT produces 4 images and size is reduced to 1/4 of original image.

3. Classification

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analysing visual images. Their applications can be seen widely in the medical images analysis. The term "Convolution" in CNN denotes that two images can be represented as matrices which are multiplied to give an output that is used to extract features from the image.

Layers of CNN

1.Input Layer

Modified GLCM image is fed as a input in theform of 3-D array of pixel values.

2.Convolutional Layer

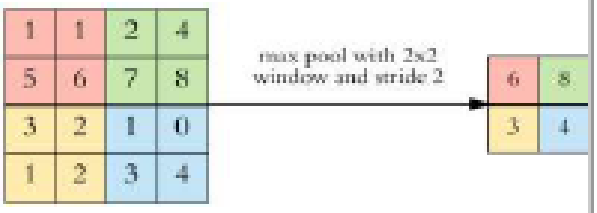
In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM). The dot products so obtained are called **feature maps**. The sum of those dot products is used to produce the output image which is fed as input to next layer.

3.ReLU Layer (rectified linear activation function or ReLU)

ReLU (Rectifier Linear Unit) is one of theactivation function. Its formula is max(x,0) means if the resultant value coming from node is positive then output would be the same positive value and if it is negative value then output would be zero. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer

4. Pooling Layer

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network.Pooling layer operates on each feature map independently. The most common approach used in pooling is max pooling.



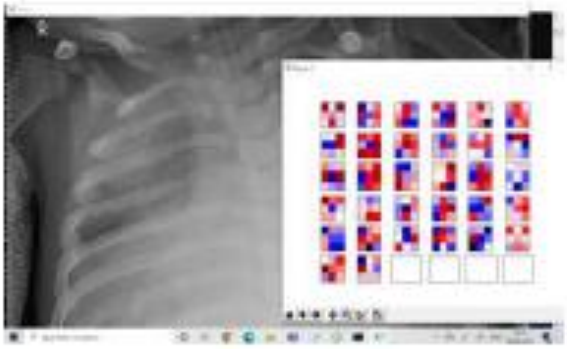
5.Fully-Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the class score for each of the classification category. In this, the input image from the previous layers are flattened and fed to the FC layer. In this stage, the classification process begins to take place.

6.Drop out Layer

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model.

Softmax Function



V. CONCLUSION

The risk of pneumonia and it's severity in the form of COVID-19 are immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. The proposed system can detect and predict pneumonia and COVID-19 diseases at an early stage based on chest X-ray images. The study employs a flexible and efficient approach of deep learning by applying the model of CNN in predicting and detecting a patient's unaffected and affected lungs with the disease, employing a chest X-ray image. The models presented could achieve 90% accuracy and 100% of validation accuracy. High validation accuracy will ensure that the number of false-negative instances is lower, hence lowers the risk to the patient's life. Thus, it is concluded that CNN classfier therefore, be effectively used for early detection of pneumonia and COVID-19 in children as well as adults. A large number of X-ray images can be processed very quickly to provide highly precise diagnostic results, thus helping health care systems provide efficient patient care services and reduce mortality rates.

REFERENCES

1. Tatiana Gabruseva, Dmytro Poplavskiy,Alexandr A. Kalinin, "Deep Learning for Automatic Pneumonia Detection" in 2020.
2. Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta ,Tony Duan, Daisy Ding, Aarti Bagul, Robyn L. Ball Curtis, Langlotz Katie Shpanskaya, "Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning" in 2017.
3. Vikash Chouhan , Sanjay Kumar Singh, Aditya Khamparia , Deepak Gupta , Prayag Tiwari , Carina Moreira , Robertas Damasevicius , and Victor Hugo C. de Albuquerque, "A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-ray Images" in 2020.
4. Manuel Vázquez Enríquez, Juan Carlos Burguillo Rial,"A Deep Learning approach for pneumonia detection on chest X-Ray", in 2019
5. Deniz Yagmur Urey, Can Jozef Saul, Doruk Taktakoglu", Early Diagnosis of Pneumonia with Deep Learning" in 2019.
6. Mohammad Rahimzadeh, Abolfazl Attar,"A Modified deep convolutional neural network for detecting COVID-19 and pneumonia fromchestX-ray images based on the concatenation and ResNet50V2" in 2020.
7. Okeke Stephen, Mangal Sain, Uchenna Joseph Maduh and Do-Un Jeong," An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare" in 2019.
8. V. Sirish Kaushik, Anand Nayyar, Gaurav Kataria ,Rachna Jain, "Pneumonia Detection Using Convolutional Neural Networks " in 2020.

9. Jaiswal, A.K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., Rodrigues, J.J.: Identifyingpneumonia in chest x-rays: a deep learning approach.

10. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, Radiologist-Level Pneumonia Detection on Chest X-rays with Deep Learning (2017).

11. Krizhevsky, A., Sutskever, I., Hinton, G.E:. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105(2012)

REFERENCE

1. Tatiana Gabruseva, Dmytro Poplavskiy,Alexandr A. Kalinin, “Deep Learning for Automatic Pneumonia Detection” in 2020.

2. Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta ,Tony Duan, Daisy Ding, Aarti Bagul, Robyn L. Ball Curtis, Langlotz Katie Shpanskaya, “Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning” in 2017.

3. Vikash Chouhan , Sanjay Kumar Singh, Aditya Khamparia , Deepak Gupta , Prayag Tiwari , Carina Moreira , Robertas Damasevicius , and Victor Hugo C. de Albuquerque, ”A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-ray Images” in 2020.

4. Manuel Vázquez Enríquez, Juan Carlos Burguillo Rial,”A Deep Learning approach for pneumonia detection on chest X-Ray”, in 2019.

5. Deniz Yagmur Urey, Can Jozef Saul, Doruk Taktakoglu”, Early Diagnosis of Pneumonia with Deep Learning” in 2019.

6. Mohammad Rahimzadeh, Abolfazl Attar,”A Modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation and ResNet50V2” in 2020.

7. Okeke Stephen, Mangal Sain, Uchenna Joseph Maduh and Do-Un Jeong,” An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare” in 2019.

8. V. Sirish Kaushik, Anand Nayyar, Gaurav Kataria ,Rachna Jain, “Pneumonia Detection Using Convolutional Neural Networks ” in 2020.

9. Jaiswal, A.K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., Rodrigues, J.J.: Identifyingpneumonia in chest x-rays: a deep**.**