

Covid-19 Pneumonia Identification

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

Kommana Bhanu Siva Kumar (Reg. No. 37110352)

Konchada Hema Abhishikth (Reg. No. 37110355)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

**(DEEMED TO BE UNIVERSITY) Accredited with Grade "A" by
NAAC JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI –
600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KOMMANA BHANU SIVA KUMAR (Reg. No. 37110352)** and **KONCHADA HEMA ABHISHIKTH (Reg. No.37110355)** who carried out the project entitled "**Covid-19 Pneumonia Identification**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Dr.Meera Gandhi,M.E., Ph.D.,

Head of the Department

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Kommana Bhanu Siva Kumar** and **Konchada Hema Abhishikth** hereby declare that the Project Report entitled “**Covid-19 Pneumonia Identification**” is done by us under the guidance of **Dr.Meera Gandhi** Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Meera Gandhi, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The presence of symptoms of COVID-19 pneumonia may be similar to other types of viral pneumonia. Because of this, it can be difficult to tell what's causing your condition without being tested for COVID-19 or other respiratory infections. to determine how COVID-19 pneumonia differs from other types of pneumonia. Information from these studies can potentially help in diagnosis and in furthering our understanding of how SARS-CoV-2 affects the lungs We present a Convolutional Neural Network in TensorFlow and Keras based Covid-19 pneumonia classification. the proposed system based on CNN using Pneumonia images to classifying the Covid-19, normal, pneumonia this system using CNN model. It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully covid-19&pneumonia. We have demonstrated the efficacy and potential of using deep CNN to image.

Table Content

Chapter 1 Introduction

1.1	Domain Overview	1-3
1.2	Deep Learning	4-7

Chapter 2 Literature Survey

2.1	Literature	8-11
2.2	Existing System	11-12
2.3	Proposed System	12-13

Chapter 3 Aim and Scope of Project

3.1	Aim of Project	14
3.2	Scope of Project	14
3.3	Advantages of proposed system	15

Chapter 4 Materials,Methods and Algorithm

4.1	Algorithm	16
4.2	Module 1	18
4.3	Module 2	19
4.4	Module 3	19-22
4.5	Module 4	23
4.6	Architecture	23
4.7	Requirements	
4.7.1	Hardware Required	24
4.7.2	Software Required	24
4.7.3	Libraries Required	24-25
4.7.4	Packages Required	26

Chapter 5 Results and Discussion

5.1 Output Screenshots	28-31
------------------------	-------

Chapter 6 Conclusion and Future Work

6.1 Conclusion	32
----------------	----

6.2 Future Work	32
-----------------	----

References	33-34
-------------------	-------

Appendices

Sample Code	33-37
-------------	-------

LIST OF FIGURES

Fig 1.1	Neural Network Working	5
Fig 4.1	Deep Learning Network	16
Fig 4.2	Training Normal X-ray Set	18
Fig 4.3	Training Pneumonia X-ray Set	19
Fig 4.4	Training Covid-19 X-ray Set	19
Fig 4.5	Sequential Method	20
Fig 4.6	Accuracy of trained model	21
Fig 4.7	Trained Model Loss Graph	22
Fig 4.8	Architecture Diagram	23
Fig 4.9	Anaconda Navigator UI	24
Fig 5.1	Epoch of Batches	27
Fig 5.2	Sequential Model Output	28
Fig 5.3	Specificity and Accuracy	28
Fig 5.4	Final Model Accuracy & Loss	29
Fig 5.5	Input X-ray Image	30
Fig 5.6	Output of Given X-ray	30

CHAPTER 1

INTRODUCTION

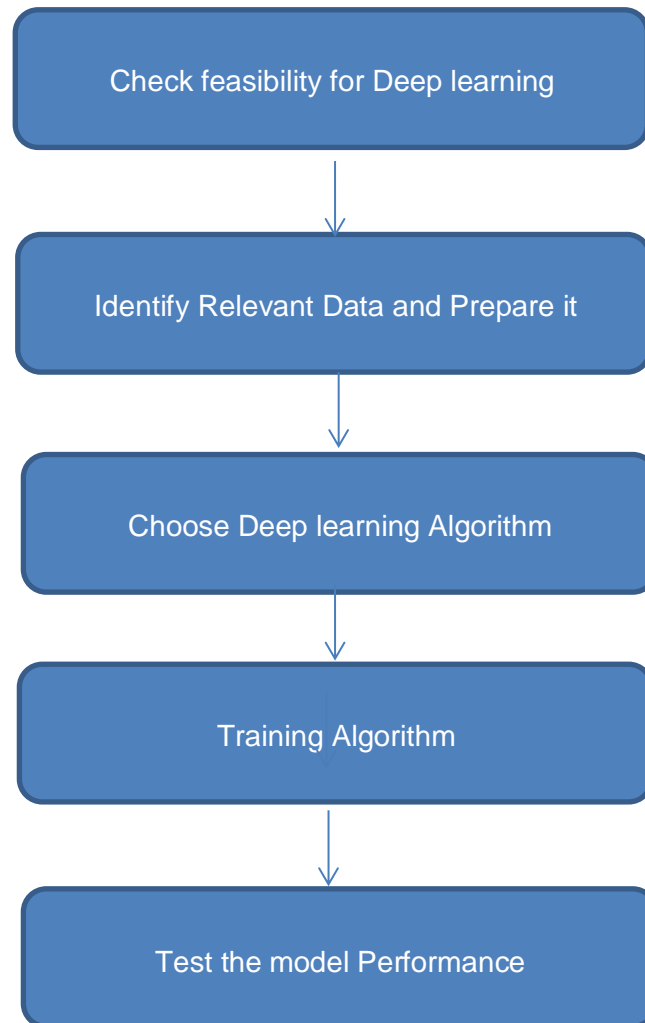
1.1 Domain Overview: -

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.



NUMPY: -

Numpy is the core library for scientific computing in Python. It provides a highperformance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A Numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension. NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with 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.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays. an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

1.2 Deep Learning: -

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.
2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).

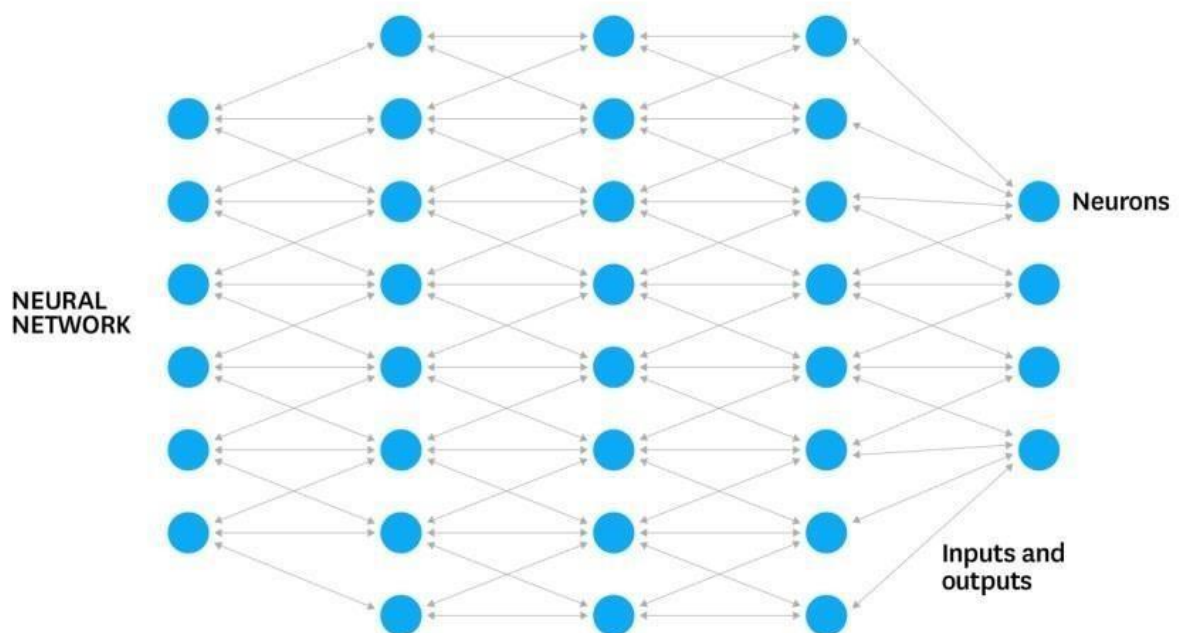


Fig: -1.1 Neural Network working

Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters.

Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output.

Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs

Inputs are passed into the first layer. Individual neurons receive the inputs, with each of them receiving a specific value. After this, an output is produced based on these values.

Outputs

The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined.

Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)
5. Deep Belief Network (DBN)

Artificial neural networks are one of the main tools used in machine learning. As the

“neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

As a result, we can say that ANNs are composed of multiple nodes. That imitate biological neurons of the human brain. Although, we connect these neurons by links. Also, they interact with each other. Although, nodes are used to take input data. Further, perform simple operations on the data. As a result, these operations are passed to other neurons. Also, output at each node is called its activation or node value.

CHAPTER 2

LITERATURE SURVEY

1.Title: Automatic Detection and Diagnosis of Severe Viral Pneumonia CT Images
Based on LDASVM

Author: Gengfei Ling, Congcong Cao

The identification of pneumonia types mainly depends on the experience of doctors, but some CT images of pneumonia are very similar, even experienced doctors are prone to misdiagnosis. In order to solve the problems of inefficiency, coarse granularity and poor accuracy under the background of large data, LDA-SVM (Linear Discriminate Analysis-support vector machine) classification algorithm in machine learning field is introduced. LDA is used to extract features from images, and SVM classifier is used to classify the sub-datasets with strong fusion features. On this basis, fusion index and intermediary centrality index are selected to measure the fusion degree of patent sub-centralization technology and identify the key technologies in the fusion process, Because of the fusion of several algorithms, the algorithm needs many iteration training, and the computation time is too long. And simulation results show that our proposed method has significant improvement on identification accuracy rate. The algorithm needs many iteration training, and the computation time is too long. LDA is used to extract features from images, and SVM classifier is used to classify the sub-datasets with strong fusion features

2.Title: Dual-Sampling Attention Network for Diagnosis of COVID-19 from
Community Acquired Pneumonia

Author: Xi Ouyang , Jiayu Huo , Liming Xia , Fei Shan

The coronavirus disease (COVID-19) is rapidly spreading all over the world, and has infected more than 1,436,000 people in more than 200 countries and territories as of April 9, 2020. Detecting COVID-19 at early stage is essential to deliver proper healthcare to the patients and also to protect the uninfected population. To this end, They develop a dual-sampling attention network to automatically diagnose COVID19 from the community acquired pneumonia (CAP) in chest computed tomography (CT). In particular, we propose a novel online attention module with a 3D convolutional network (CNN) to focus on the infection regions in lungs when making decisions of diagnoses. Note that there exists imbalanced distribution of the sizes of the infection regions between COVID-19 and CAP, partially due to fast progress of

COVID-19 after symptom onset. Therefore, we develop a dual-sampling strategy to mitigate the imbalanced learning s. In the training-validation stage, we collect 2186 CT scans from 1588 patients for a 5-fold cross-validation. In the testing stage, we employ another independent large-scale testing dataset including 2796 CT scans from 2057 patients. COVID-19, it is important to get the diagnosis result at soon as possible. Although RT-PCR is the current ground truth to diagnose COVID19, it will take up to days to get the final results and the capacity of the tests is also limited in many places especially in the early outbreak [8]. In this study, we explore a machine learning method to perform automatic COVID-19 diagnosis from CAP in chest CT images. We evaluate our method by the largest multi-center CT data in the world, to the best of our knowledge.

3.Title: Field Trial of Aspiration Pneumonia Predicion based on Electronic Medical Records.

Author: Masahiro Hayashitani, Eiji Yumoto, Toshinori Hosoi, Masahiro Kubo. A prediction method of aspiration pneumonia in department of neurosurgery and demonstrate the method in a hospital in order to reduce workload about care for aspiration pneumonia. In a field trial, medical staff provide preventive cares based on the output of the method. They show that the trial reduces workload of the medical staff, and the number of patients with aspiration pneumonia was reduced. They proposed the prediction method of aspiration pneumonia in department of neurosurgery in order to reduce workload about care for aspiration pneumonia. The prediction method is based on electronic medical records including age, sex, and vital signs. They demonstrated the method in KIH. In the field trial, the medical staff provided preventive cares based on the output of method. They showed that the trial reduced care time of the medical staff by 10%, and the number of patients with aspiration pneumonia was reduced. A long-time trial will be conducted in order to confirm further effects.

4.Title: Trend Prediction of Influenza and the Associated Pneumonia in Taiwan Using Machine Learning

Author: Sing-Ling Jhuo, Mi-Tren Hsieh, Ting-Chien Weng, Mei-Juan Chen and Chieh-Ming Yang.

Trend prediction of influenza and the associated pneumonia can provide the information for taking preventive actions for public health. This paper uses meteorological and pollution parameters, and acute upper respiratory infection (AURI) outpatient number as input to multilayer perceptron (MLP) to predict the patient number of influenza and the associated pneumonia in the following week. The meteorological parameters in use are temperature and relative humidity, air pollution parameters are Particulate Matter 2.5 (PM 2.5) and Carbon Monoxide (CO), and the patient prediction includes both outpatients and inpatients. Patients are classified by textiles into three categories: high, moderate, and low volumes. The regional data analyses with various age groups are also provided in this paper. The meteorological parameters (temperature and relative humidity), air pollution parameters (PM 2.5 and CO) of 30 consecutive days, and the number of AURI patient of previous week are utilized to forecast the trend of the patients of flu and viral pneumonia of subsequent week by MLP machine learning. This work can provide prevention actions for diffusion of influenza.

5.Title: Cloud-Based Smart Dog Music Therapy and Pneumonia Detection System for Reducing the Difficulty of Caring for Patients with Dementia

Author: Mei-Jung Lyu , Shyan-Ming Yuan

There is currently no cure for Alzheimer's disease, leaving patients to rely solely on good quality care services to prolong their life. Besides, it has been found that the onset of pneumonia can accelerate the progression of dementia and even lead to death. This has led to an increased caregiving burden and a level of emotional stress among caregivers that can be unbearable. The aim of this study was to build a Smart Dog music therapy and pneumonia detection system, which combines a robotic dog, cloud technology, a multi-agent system, an adaptive network-based fuzzy inference system (ANFIS), a web application, and sensor technology to deliver care for patients with Alzheimer's disease and help mitigate the difficulties faced by caregivers. The use of the system to determine its usefulness in caregiving work and whether it improved their overall caregiving experience. A majority of the interviewed caregivers agreed that the system brought about improvements.

5. Title: Deep Regression via Multi-Channel Multi-Modal Learning for Pneumonia Screening

Author: Qiuli Wang, Dan Yang, Zhihuan Lii, Xiahong Zhang

Pneumonia screening is one of the most crucial steps in the pneumonia diagnosing system, which can improve the work efficiency of the radiologists and prevent delayed treatments. In this paper, we propose a deep regression framework for automatic pneumonia screening, which jointly learns the multi-channel images and multi-modal information (i.e., clinical chief complaints, age, and gender) to simulate the clinical pneumonia screening process. We demonstrate the advantages of the framework in three ways. First, visual features from multi-channel images (Lung Window Images, High Attenuation Images, Low Attenuation Images) can provide more visual features than single image channel, and improve the ability of screening pneumonia with severe diseases. Second, the proposed framework treats chest CT scans as short video frames and analyzes them by using Recurrent Convolutional Neural Network, which can automatically extract multiple image features from multichannel image slices. Third, chief complaints and demographic information can provide valuable prior knowledge enhancing the features from images and further promote performance. The proposed framework has been extensively validated in 900 clinical cases. Compared to the baseline, the proposed framework improves the accuracy by 2.3% and significantly improves the sensitivity by 3.1%.

2.2 Existing System:

This reports Segmentation of pneumonia lesions from CT scans of COVID-19 patients is important for accurate diagnosis and follow-up. Deep learning has a potential to automate this task but requires a large set of high-quality annotations that are difficult to collect. Learning from noisy training labels that are easier to obtain has a potential to alleviate this problem. To this end, we propose a novel noise-robust framework to learn from noisy labels for the segmentation task. 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. For the COVID-19 pneumonia lesion segmentation task, the pixel-level annotations are often noisy and clean annotations are extremely difficult to collect due to several reasons. First, different annotators may have different annotation standards that lead to inter-

observer variability, and high intra-observer variability may also exist. These variabilities are very likely to cause noise in the annotations, that demonstrates disagreement between two annotators. Second, to reduce the annotation efforts, some researchers use a human-in-the-loop strategy, where the annotator only provides refinements to algorithm-generated labels for annotating. In such cases, the annotations can be largely biased towards the results of the algorithm and thus contain noisy pixel-level labels. In addition, collecting less accurate annotations from non-experts is another promising solution to overcome the limited availability of experts, but these annotations are also noisy at pixel level and may limit the performance of the deep learning model. This can be either due to challenges for accurate annotation, such as low contrast, ambiguous boundaries and complex appearances of the target, or caused by low-cost inaccurate annotations such as annotations provided by non-experts, human in the loop strategies and some algorithms generating pseudo labels. They One advantage of our noise robust Dice loss function is that it does not depend on a specific and can be combined with different training strategies, such as a standard training process and the self-assembling framework in our method.

Drawback:

- It has not focused on AlexNet CNN in keras and TensorFlow as classifier.
- They are not using OpenCV computer vision technique
- It has not focused on increasing the recognition rate and classification accuracy of severity image of chest X-ray.

2.3 Proposed System:

- We are proposing recognition framework based on the structured two dimensional convolutional neural networks (CNNs) type of AlexNet to classify the Covid-19&Pneumonia and improve the accuracy of workflow.
- The proposed method for this project is to train a Deep Learning algorithm capable of classifying Covid-19&Pneumonia images and data preprocessing and visualizing the image then feature extracting to build AlexNet CNN using Covid-19&Pneumonia image dataset we classify it such as Covid-19,normal,pneumonia this system using CNN model.

- It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully covid-19&pneumonia. We have demonstrated the efficacy and potential of using deep CNN to images..

Advantages:

- The large amount of chest x-ray data can be train on artificial neural network.
- It is best model for deep learning technique to easily classifying Covid-19&Pneumonia .

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 Aim of project: -

To Train large amount of chest scan data on artificial neural network. The recognition framework based on the structured two dimensional convolutional neural networks (CNNs) type of Alex Net to classify the Covid-19 & Pneumonia and improve the accuracy of workflow. To make it best model for deep learning technique to easily classifying "Covid-19" & "Pneumonia".

The proposed procedure includes numerous collaborative steps, such as data preprocessing, feature extraction using various x-rays, feature selection using information value, classification, as well as model validation and result interpretation.

3.2 Scope of the Project

The scope of the project is to easily classify the disease whether it is covid-19, pneumonia or normal one based on chest x-ray scan.

We can easily identify what type of infection caused without being tested based on a chest scan.

We can quickly identify the person having covid-19.

3.3 Advantages of Proposed System

We are proposing recognition framework based on the structured two dimensional convolutional neural networks (CNNs) type of AlexNet to classify the Covid-19 & Pneumonia and improve the accuracy of workflow.

The proposed method for this project is to train a Deep Learning algorithm capable of classifying Covid-19 & Pneumonia images and data preprocessing and visualizing the image then feature extracting to build AlexNet CNN using Covid-19 & Pneumonia

image dataset we classify it such as Covid-19, normal, pneumonia this system using CNN model.

It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully covid-19 & pneumonia.

We have demonstrated the efficacy and potential of using deep CNN to images. The large amount of chest x-ray data can be train on artificial neural network. It is best model for deep learning technique to easily classifying Covid-19 & Pneumonia.

CHAPTER 4

MATERIALS,METHODS AND ALGORITHMS

4.1 Algorithm: -

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations. Deep learning requires large amounts of labeled data. Ex: - driverless car development requires millions of images and thousands of hours of video.

1. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

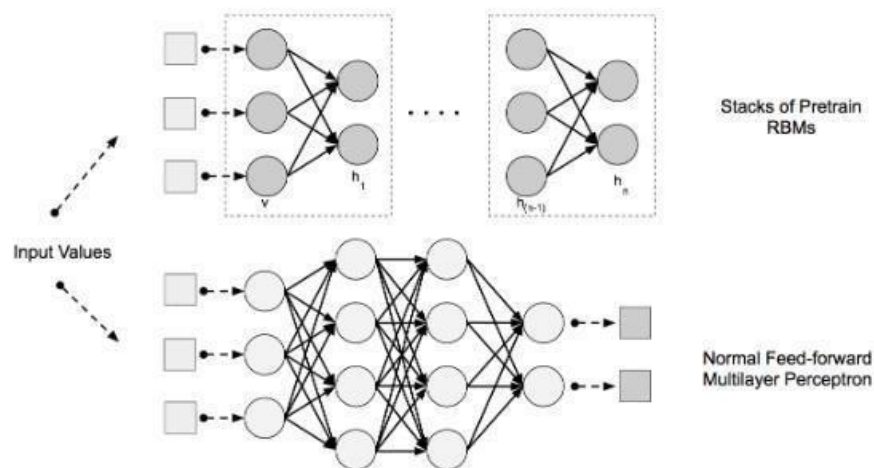


Fig 4.1: - Deep Learning Networks

One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or Conv Net). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not retrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

CNNs learn to detect different features of an image using tens or hundreds of hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognize.

How to Create and Train Deep Learning Models: -

a) Training from Scratch: -

To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train.

b) Transfer Learning: -

Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also

has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours

c) Feature Extraction: -

A slightly less common, more specialized approach to deep learning is to use the network as a feature extractor. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process. These features can then be used as input to a machine learning model such as support vector machines (SVM).

The main contributions of this project therefore are (Modules):

- Data Analysis - Import the given image from dataset and training the module with manual CNN. (module01)
- Dataset Preprocessing - To train the dataset by using AlexNet. (module02)
- Training the Model - To train the dataset using LeNet. (module03)
- Testing of Dataset - Deploying the model in Django Framework and predicting output. (module 04)

4.2 Module 1: Import the given image from dataset:

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.

NORMAL->

Trained data for dir_name_train_NORMAL:

```
===== Images in: Data/train/NORMAL
images_count: 471
min_width: 993
max_width: 2534
min_height: 617
max_height: 2534
```



Fig 4.2: - Training Normal X-ray set

PNEUMONIA->

Trained data for PNEUMONIA:

```
===== Images in: Data/train/PNEUMONIA
images_count: 517
min_width: 502
max_width: 1944
min_height: 307
max_height: 1944
```

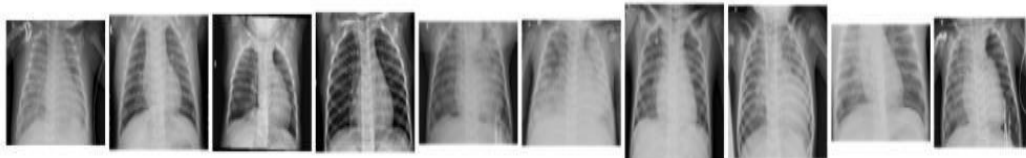


Fig 4.3: - Training Pneumonia X-ray set

COVID-19->

Trained data for COVID19 type disease:

```
===== Images in: Data/train/COVID19
images_count: 460
min_width: 224
max_width: 4757
min_height: 224
max_height: 4757
```

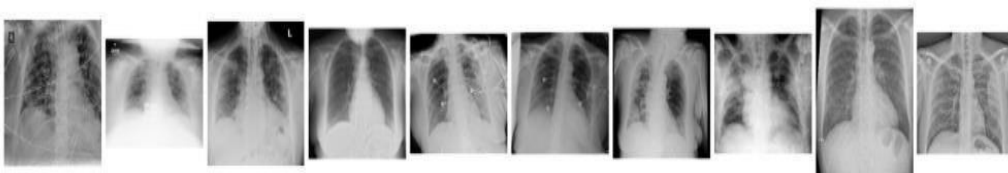


Fig 4.4: - Training Covid-19 X-ray set

4.3 Module 2: To train the dataset by using AlexNet.

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 75, 75, 32)	896
max_pooling2d (MaxPooling2D)	(None, 37, 37, 32)	0
conv2d_1 (Conv2D)	(None, 12, 12, 128)	36992
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
dense_1 (Dense)	(None, 3)	771
Total params: 1,218,563		
Trainable params: 1,218,563		
Non-trainable params: 0		

Fig 4.5: - Sequential Model

4.4 Module 3: To train the model using LeNet:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are handengineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

Input Layer: -

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

Convo Layer: -

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive fields (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

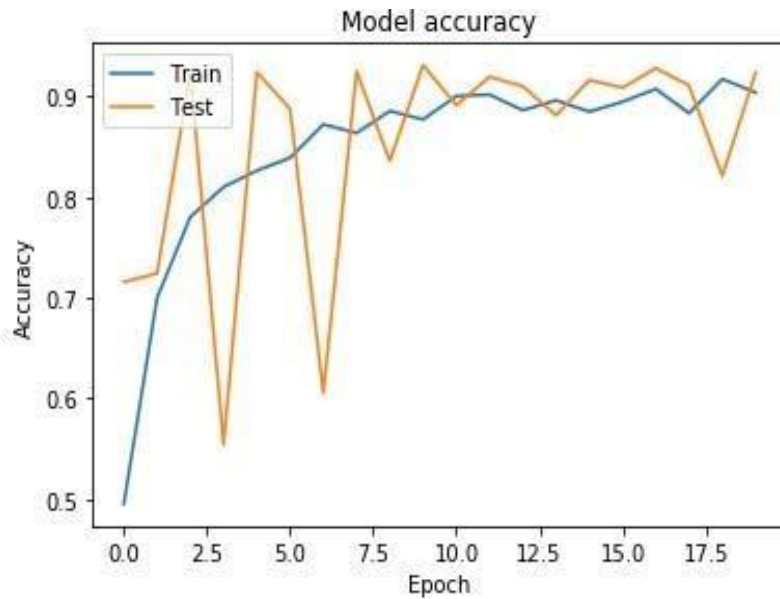


Fig 4.6: - Accuracy Graph

Pooling Layer: -

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

Fully Connected Layer (FC): -

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Logistic Layer: -

Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

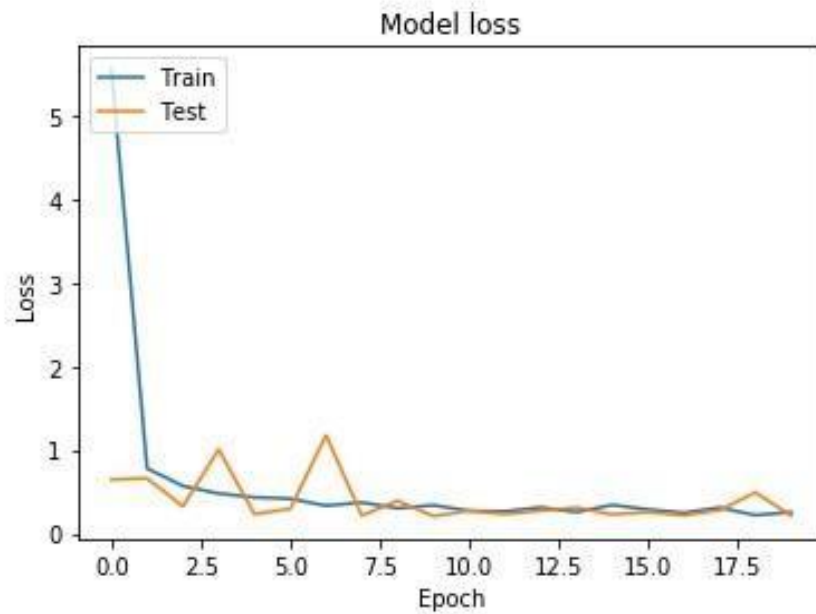


Fig 4.7: - Model Loss Graph

Output Layer: -

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

4.5 Module 4: Deploying the model in Django Framework and predicting output

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output whether the given chest X-ray is covid19 / pneumonia / normal.

4.6 Architecture: -

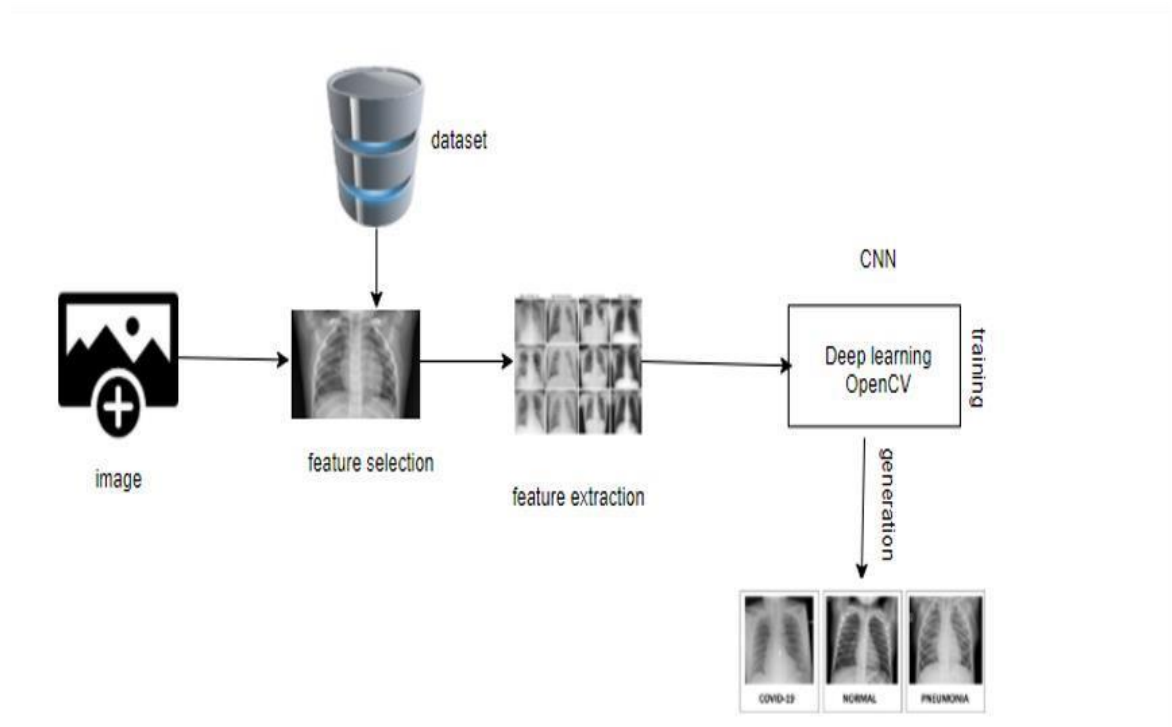


Fig 4.8: - Architecture Diagram

4.7 Requirements: -

4.7.1 Hardware Requirements

- Hard Disk: 50 GB and Above
- RAM: 8GB and above
- Processor: i5 and above (64 bit)

4.7.2 Software Requirements

Technologies

- Python
- Anaconda python

IDE

- Jupiter notebook

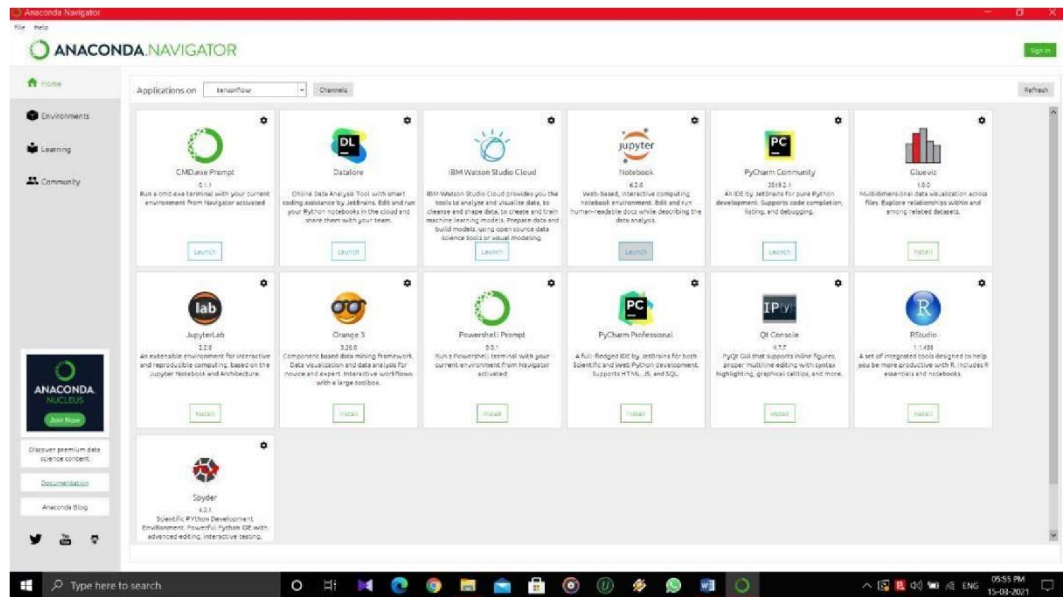


Fig 4.9 – Anaconda Navigator UI

4.7.3 Libraries Required:

- **numpy**: To process the image matrices
- **os**: To access the file system to read the image from the train and test directory from our machines
- **random**: To shuffle the data to overcome the biasing
- **matplotlib**: To display the result of our predictive outcome.
- **Tensor flow**: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.

Numpy:

NumPy is an acronym for "Numeric Python" or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices. The implementation is even aiming at huge matrices and arrays. Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays. It is the

fundamental package for scientific computing with Python. It contains various features including these important ones: -

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Useful linear algebra, Fourier Transform, and random number capabilities

Numpy Array:

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array. The shape of an array is a tuple of integers giving the size of the array along each dimension.

SciPy:

SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others. NumPy is based on two earlier Python modules dealing with arrays. One of these is Numeric. Numeric is like NumPy a Python module for high-performance, numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is built on the code of Numeric and the features of numarray.

Tensorflow:

Tensorflow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of Tensorflow.

4.7.4: -Packages list

```
pip install tensorflow==2.2
```

```
pip install keras==2.2.4
```

pip install pandas

pip install flask

pip install numpy

pip install image

pip install opencv-python

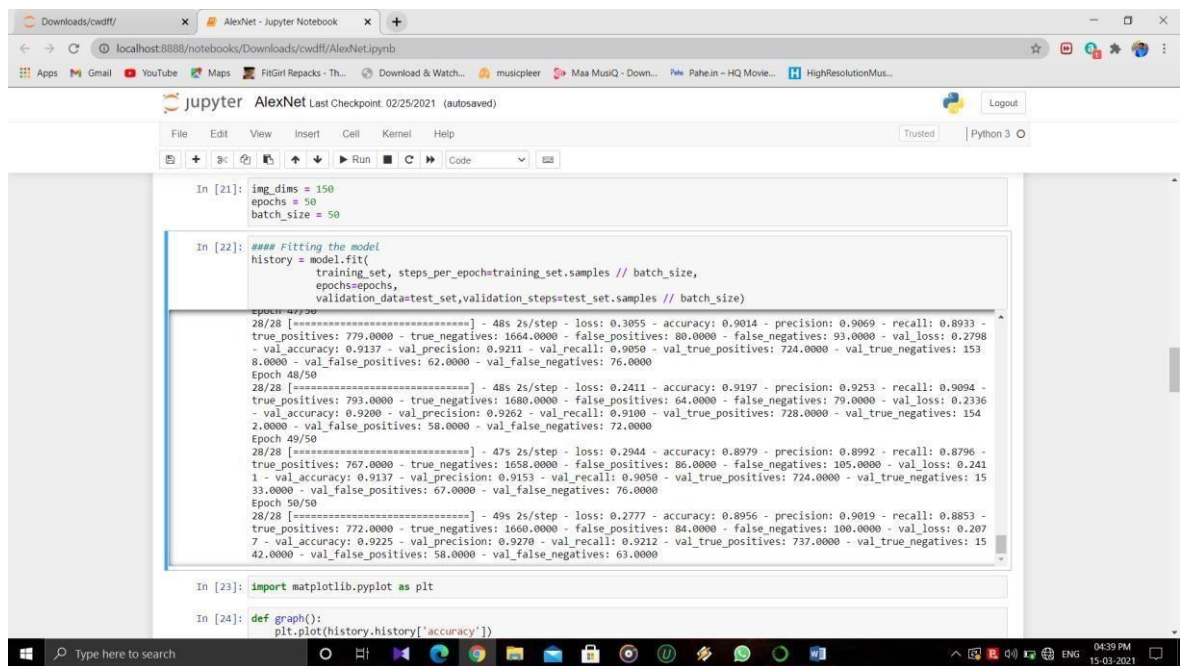
pip install sklearn

CHAPTER 5

RESULTS AND DISCUSSION

5.1 OUTPUT: -

After the model is trained successfully, the software can identify the disease if the X-ray is contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the disease.



```
In [21]: img_dims = 150
epochs = 50
batch_size = 50

In [22]: ### Fitting the model
history = model.fit(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set, validation_steps=test_set.samples // batch_size)

28/28 [=====] - 48s 2s/step - loss: 0.3055 - accuracy: 0.9014 - precision: 0.9069 - recall: 0.8933 -
true_positives: 779.0000 - true_negatives: 1664.0000 - false_positives: 80.0000 - false_negatives: 93.0000 - val_loss: 0.2798
- val_accuracy: 0.9137 - val_precision: 0.9211 - val_recall: 0.9050 - val_true_positives: 724.0000 - val_true_negatives: 153
8.0000 - val_false_positives: 62.0000 - val_false_negatives: 76.0000
Epoch 48/50
28/28 [=====] - 48s 2s/step - loss: 0.2411 - accuracy: 0.9197 - precision: 0.9253 - recall: 0.9094 -
true_positives: 793.0000 - true_negatives: 1680.0000 - false_positives: 64.0000 - false_negatives: 79.0000 - val_loss: 0.2336
- val_accuracy: 0.9200 - val_precision: 0.9262 - val_recall: 0.9100 - val_true_positives: 728.0000 - val_true_negatives: 154
2.0000 - val_false_positives: 58.0000 - val_false_negatives: 72.0000
Epoch 49/50
28/28 [=====] - 47s 2s/step - loss: 0.2944 - accuracy: 0.8979 - precision: 0.8992 - recall: 0.8796 -
true_positives: 767.0000 - true_negatives: 1658.0000 - false_positives: 86.0000 - false_negatives: 105.0000 - val_loss: 0.241
1 - val_accuracy: 0.9137 - val_precision: 0.9153 - val_recall: 0.9050 - val_true_positives: 724.0000 - val_true_negatives: 15
33.0000 - val_false_positives: 67.0000 - val_false_negatives: 76.0000
Epoch 50/50
28/28 [=====] - 49s 2s/step - loss: 0.2777 - accuracy: 0.8956 - precision: 0.9019 - recall: 0.8853 -
true_positives: 772.0000 - true_negatives: 1660.0000 - false_positives: 84.0000 - false_negatives: 100.0000 - val_loss: 0.207
7 - val_accuracy: 0.9225 - val_precision: 0.9270 - val_recall: 0.9212 - val_true_positives: 737.0000 - val_true_negatives: 15
42.0000 - val_false_positives: 58.0000 - val_false_negatives: 63.0000

In [23]: import matplotlib.pyplot as plt

In [24]: def graph():
plt.plot(history.history['accuracy'])
```

Fig 5.1: -Batches Epoching

An **epoch** is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed. Datasets are usually grouped into batches (especially when the amount of data is very large).

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 96)	34944
activation (Activation)	(None, 54, 54, 96)	0
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
flatten (Flatten)	(None, 69984)	0
dense (Dense)	(None, 256)	17916160
activation_1 (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 96)	24672
activation_2 (Activation)	(None, 96)	0
dropout_1 (Dropout)	(None, 96)	0
dense_2 (Dense)	(None, 32)	3104
activation_3 (Activation)	(None, 32)	0
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 3)	99
activation_4 (Activation)	(None, 3)	0
Total params:	17,978,979	
Trainable params:	17,978,979	
Non-trainable params:	0	

Fig 5.2: - Sequential Model Output

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

```

In [28]: confusion_matrix=[[true_positives,false_positives],[false_negatives,true_negatives]]

In [29]: print(f"Accuracy          : {scores[1]*100}")
print("")
print("Confution Matrix      : ",confution_matrix)
print("")
print("precision Score          : ",precision)
print("")
print("Recall or Sensitivity Score : ",recall)
print("")
f1=(precision*recall)/(precision+recall)
F1_Score=2*f1
print("F1 Score                : ",F1_Score)
print("")
Specificity=true_negatives/true_negatives+false_positives
print("Specificity Score        : ",Specificity)

Accuracy          : 91.69254899024963
Confution Matrix : [[2474.0, 108.0], [108.0, 102.0]]
precision Score   : 0.9161490797996521
Recall or Sensitivity Score : 0.9161490797996521
F1 Score          : 0.9161490797996521
Specificity Score  : 100.0

In [30]: import h5py

In [31]: model.save('covid.h5')

```

Fig 5.3: - Specificity Score & Accuracy

Specificity is defined as the proportion of actual negatives, which got predicted as the negative (or true negative).

True positives are when you predict an observation belongs to a class and it actually does belong to that class.

True negatives are when you predict an observation does not belong to a class and it actually does not belong to that class.

False positives occur when you predict an observation belongs to a class when in reality it does not.

False negatives occur when you predict an observation does not belong to a class when in fact it does.

The Specificity is calculated as;

$$\text{Specificity} = \frac{\text{true_negatives}}{\text{true_negatives} + \text{false_positives}}$$

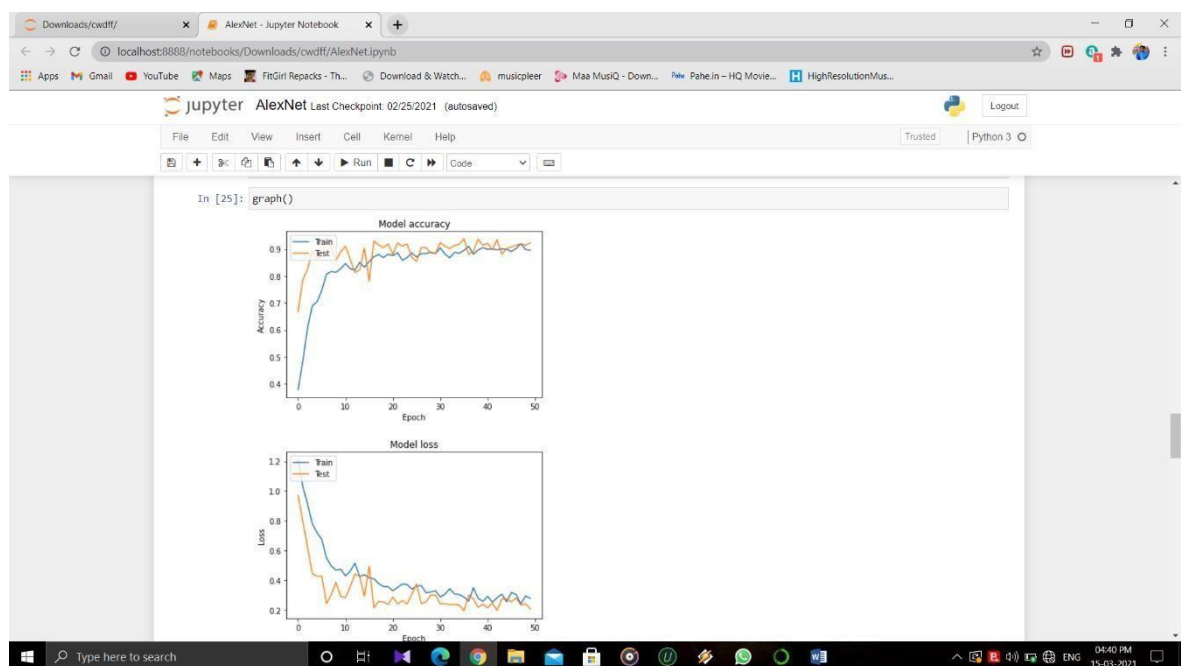


Fig 5.4: - Model Accuracy and Loss

Loss can be seen as a distance between the true values of the problem and the values predicted by the model. Greater the loss is, huger the errors you made on the data. Accuracy can be seen as the number of error you made on the data.

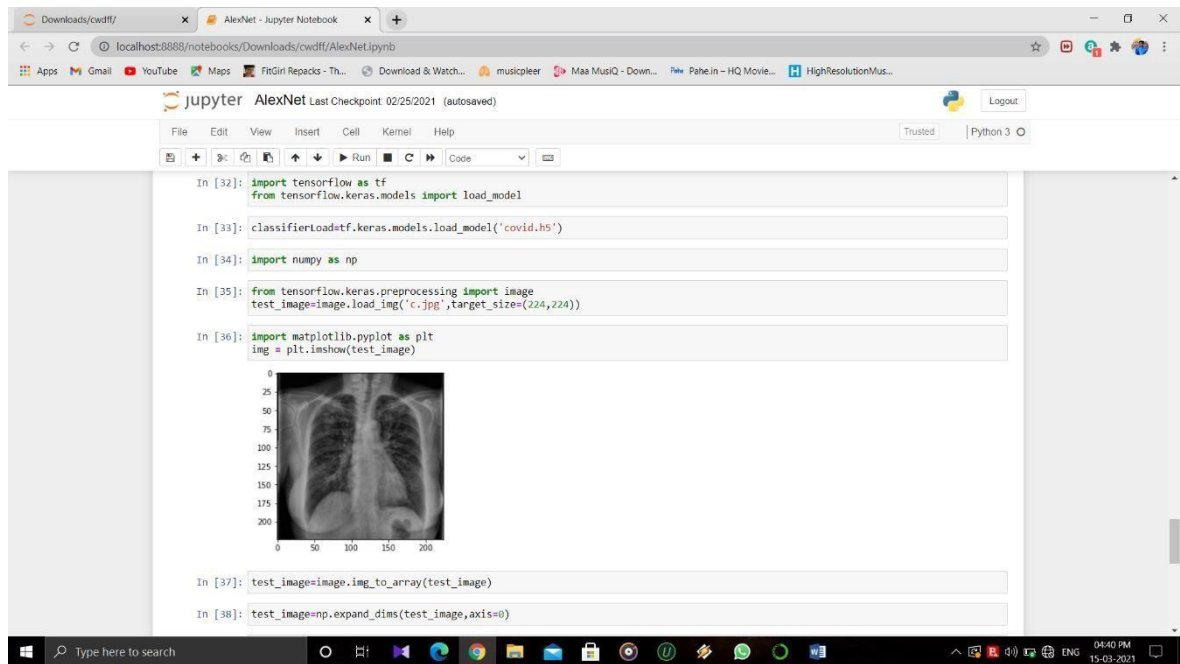


Fig 5.5: - Input X-ray Image

Here, An External X-Ray has been given as input to predict the condition of the person it belongs to. A neat scanned x-ray should be processed for input, as the accuracy should be high and good.

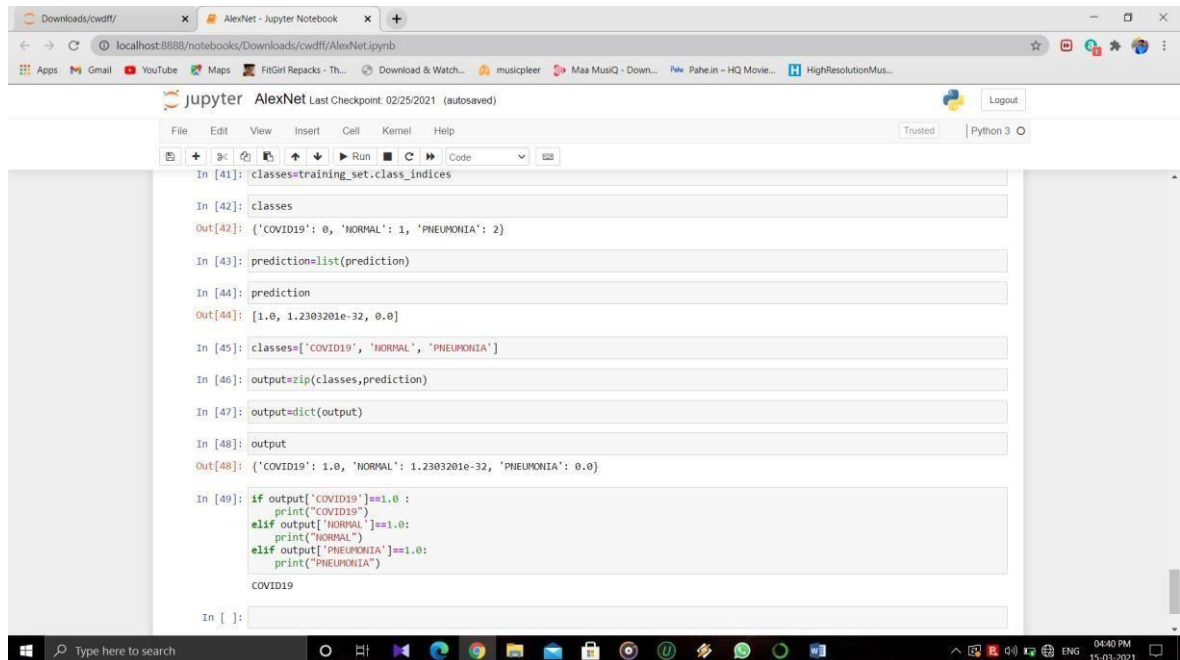


Fig5.6: -Output of the X-ray As we

can see,

After the Calculations the Covid19 value is "1" and remaining two conditions are 0 & 1.2. We already coded that the condition is something, which comes exactly "1". So for the given X-ray, the condition is **"Covid19 Pneumonia"**

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION: -

A large amount of chest scan data has been trained on artificial neural network.

Using CNN, we successfully developed a code to check the condition of the x-ray,

The Accuracy is as high as we expected and the given X-ray has been processed without any errors and the output is correct.

6.2 FUTURE WORK: -

To deployment real time this process by show the prediction result in web application or desktop application.

To optimize the work to implement in Artificial Intelligence environment.

To deploy this model to AI on web application.

To Develop the code, so that it runs much faster and with utmost accuracy.

REFERENCES

- [1] N. Zhu, D. Zhang, W. Wang, X. Li, B. Yang, J. Song, X. Zhao, B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao, and W. Tan, "A novel coronavirus from patients with pneumonia in China, 2019," *N. Engl. J. Med.*, vol. 382, pp. 727–733, 2020.
- [2] D. Benvenuto, M. Giovanetti, M. Salemi, M. Prosperi, C. De Flora, L. C. Junior Alcantara, S. Angeletti, and M. Ciccozzi, "The global spread of 2019-nCoV: A molecular evolutionary analysis," *Pathog. Glob. Health*, pp. 1–4, 2020.
- [3] F. Shi, J. Wang, J. Shi, Z. Wu, Q. Wang, Z. Tang, K. He, Y. Shi, and D. Shen, "Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation and Diagnosis for COVID19," *IEEE Rev. Biomed. Eng.*, vol. 3333, no. c, pp. 1–13, 2020.
- [4] WHO, "Coronavirus disease 2019 (COVID-19) Situation Report - 81," 2020. [Online]. Available: https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200410-sitrep-81-covid-19.pdf?sfvrsn=ca96eb84_2
- [5] M.-Y. Ng, E. Y. Lee, J. Yang, F. Yang, X. Li, H. Wang, M. M.-s. Lui, C. S.-Y. Lo, B. Leung, P.-L. Khong, C. K.-M. Hui, K.-y. Yuen, and M. D. Kuo, "Imaging profile of the COVID-19 infection: Radiologic findings and literature review," *Radiol. Cardiothorac. Imaging*, vol. 2, no. 1, p. e200034, 2020.
- [6] L. Huang, R. Han, T. Ai, P. Yu, H. Kang, Q. Tao, and L. Xia, "Serial quantitative chest CT assessment of COVID-19: Deep-learning approach," *Radiol. Cardiothorac. Imaging*, vol. 2, p. e200075, 2020.
- [7] J. Lei, J. Li, X. Li, and X. Qi, "CT imaging of the 2019 novel coronavirus (2019-nCoV) pneumonia," *Radiology*, p. 200236, 2020.
- [8] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, J. Bai, Y. Lu, Z. Fang, Q. Song, K. Gao, D. Liu, G. Wang, Q. Xu, X. Fang, S. Zhang, J. Xia, and J. Xia, "Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest CT," *Radiology*, p. 200905, 2020.
- [9] F. Shan, Y. Gao, J. Wang, W. Shi, N. Shi, M. Han, Z. Xue, and Y. Shi, "Lung infection quantification of COVID-19 in CT images with deep learning," *arXiv*, p. 2003.04655, 2020.
- [10] Y. Cao, Z. Xu, J. Feng, C. Jin, X. Han, H. Wu, and H. Shi, "Longitudinal assessment of COVID19 using a deep learning-based quantitative CT pipeline: Illustration of two cases," *Radiol. Cardiothorac. Imaging*, vol. 2, no. 2, p. e200082, 2020.
- [11] H. Wang, S. Zhao, Q. Dong, Y. Cui, Y. Chen, J. Han, L. Xie, and T. Liu, "Recognizing brain states using deep sparse recurrent neural network," *IEEE Trans. Med. Imaging*, vol. 38, no. 4, pp. 1058–1068, 2019. [12] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annu. Rev. Biomed. Eng.*, vol. 19, no. 1, pp. 221–248, 2017.
- [13] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels : exploring techniques and remedies in medical image analysis," *arXiv:1912.02911*, pp. 1–17, 2020.
- [14] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *AAAI*, 2017, pp. 1919–1925.
- [15] H. Zhu, J. Shi, and J. Wu, "Pick-and-Learn : Automatic quality evaluation for noisy-labeled image segmentation," in *MICCAI*, 2019, pp. 576–584. [16] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.
- [17] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *NeurIPS*, 2018, pp. 8778–8788.

- [18] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *IC3DV*, 2016, pp. 565–571.
- [19] Z. Mirikharaji, Y. Yan, and G. Hamarneh, "Learning to segment skin lesions from noisy annotations," in *MICCAI MIL3D Work.*, 2019, pp. 207–215.
- [20] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017, pp. 1195–1204.
- [21] L. Yu, S. Wang, X. Li, C. W. Fu, and P. A. Heng, "Uncertainty-aware self-ensembling model for semi-supervised 3D left atrium segmentation," in *MICCAI*, 2019, pp. 605–613.
- [22] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in *ICLR*, 2018, pp. 1–13.
- [23] C. S. Perone, P. Ballester, R. C. Barros, and J. Cohen-Adad, "Unsupervised domain adaptation for medical imaging segmentation with selfensembling," *Neuroimage*, vol. 194, no. January, pp. 1–11, 2019.
- [24] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," *MICCAI Work. DLMIA*, vol. 11045 LNCS, pp. 3–11, 2018.
- [25] S. Jin, B. Wang, H. Xu, C. Luo, L. Wei, W. Zhao, X. Hou, W. Ma, Z. Xu, Z. Zheng, W. Sun, L. Lan, W. Zhang, X. Mu, C. Shi, Z. Wang, J. Lee, Z. Jin, M. Lin, H. Jin, L. Zhang, J. Guo, B. Zhao, Z. Ren, S. Wang, Z. You, J. Dong, X. Wang, J. Wang, and W. Xu, "AI-assisted CT imaging analysis for COVID-19 screening: Building and deploying a medical AI system in four weeks," *medRxiv*, 2020.
- [26] J. Chen, L. Wu, J. Zhang, L. Zhang, D. Gong, Y. Zhao, S. Hu, Y. Wang, X. Hu, B. Zheng, K. Zhang, H. Wu, Z. Dong, Y. Xu, Y. Zhu, X. Chen, L. Yu, and H. Yu, "Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography: a prospective study," *medRxiv*, 2020.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778. [28] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, "SELF: Learning to filter noisy labels with self-ensembling," *CoRR abs/1910.01842*, pp. 1–15, 2019. [29] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," *ICML*, vol. 10, pp. 6900–6909, 2018.
- [30] C. Xue, Q. Dou, X. Shi, H. Chen, and P. A. Heng, "Robust learning at noisy labeled medical images: Applied to skin lesion classification," in *ISBI*, 2019, pp. 1280–1283.

APPENDICES

Sample Code: -

To build a model for Alex Net Convolution neural network:-

```
import tensorflow as tf import tensorflow.keras.backend as K from
tensorflow.keras.models import Model from tensorflow.keras.models
import Sequential from tensorflow.keras.layers import Input from
tensorflow.keras.layers import Dense from tensorflow.keras.layers import
Flatten from tensorflow.keras.layers import Conv2D from
tensorflow.keras.layers import MaxPooling2D from
tensorflow.keras.layers import Dropout from tensorflow.keras.layers
import LeakyReLU from tensorflow.keras.optimizers import Adam from
tensorflow.keras.preprocessing.image import ImageDataGenerator from
tensorflow.keras.callbacks import ModelCheckpoint from
tensorflow.keras.callbacks import ReduceLROnPlateau from
tensorflow.keras.callbacks import EarlyStopping import warnings
warnings.filterwarnings('ignore') model = Sequential() # 1st Convolutional
Layer model.add(Conv2D(filters=96, input_shape=(224,224,3),
kernel_size=(11,11), strides=(4,4), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# Passing it to a Fully Connected layer
model.add(Flatten()) # 1st Fully
Connected Layer
model.add(Dense(256))
model.add(Activation('relu')) # Add
Dropout to prevent overfitting
model.add(Dropout(0.4)) # 2nd Fully
```

```

Connected Layer model.add(Dense(96))
model.add(Activation('relu'))
# Add Dropout model.add(Dropout(0.4)) # 3rd Fully Connected Layer
model.add(Dense(32)) # Add Dropout model.add(Dropout(0.4)) #
Output Layer model.add(Dense(3)) model.add(Activation('softmax'))
model.summary() # Compile the model model.compile(loss =
'categorical_crossentropy', optimizer='adam',
metrics=['accuracy', 'AUC', 'Precision', 'Recall', 'TruePositives', 'TrueNegatives', 'False
Positives', 'FalseNegatives'])
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) test_datagen=ImageDataGenerator(rescale=1./255)
training_set=train_datagen.flow_from_directory('Data/train', target_size=(224,224),
batch_size=32, class_mode='categorical')
test_set=test_datagen.flow_from_directory('Data/test', target_size=(224,224), batch
_size=32, class_mode='categorical') img_dims = 150 epochs = 50
batch_size = 50 ##### Fitting the model history = model.fit(
training_set, steps_per_epoch=training_set.samples // batch_size,
epochs=epochs,
validation_data=test_set, validation_steps=test_set.samples //
batch_size)

import matplotlib.pyplot as plt
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy') plt.ylabel('Accuracy')
    plt.xlabel('Epoch') plt.legend(['Train', 'Test'],
loc='upper left') plt.show()
    # Plot training & validation loss values
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss']) plt.title('Model

```

```
loss') plt.ylabel('Loss') plt.xlabel('Epoch')
plt.legend(['Train',
```

```
'Test'], loc='upper left') plt.show()
#Plot training & validation AUC values
plt.plot(history.history['auc'])
plt.plot(history.history['val_auc'])
plt.title('Model AUC Curve')
plt.ylabel('AUC') plt.xlabel('Epoch')
plt.legend(['Train', 'Valid'], loc='upper left')
plt.show() graph()
```

```
print("[INFO] Calculating model accuracy")
scores = model.evaluate(test_set) print(f"Test
Accuracy: {scores[1]*100}") precision=scores[3]
recall=scores[3] true_positives=scores[5]
true_negatives=scores[6]
false_positives=scores[7]
false_negatives=scores[8]
confution_matrix=[[true_positives,false_positi
ves],[false_negatives,true_negatives]
]
print(f"Accuracy: {scores[1]*100}") print("")
print("Confution Matrix: ",confution_matrix)
print("") print("precision Score: ",precision) print("")
print("Recall or Sensitivity Score :
",recall) print("")
f1=(precision*recall)/(precision+recall
) F1_Score=2*f1 print("F1 Score:
",F1_Score) print("")
Specifisity=true_negatives/true_negatives+false_positives print("Specifisity Score:
",Specifisity)
```

```
import h5py model.save('COVID_and_pneumonia.h5') import tensorflow
```

```

as tf from tensorflow.keras.models import load_model
classifierLoad=tf.keras.models.load_model('COVID_and_pneumonia.h5')
import numpy as np from tensorflow.keras.preprocessing import image
test_image=image.load_img('pne1.jpg',target_size=(225,225)) import
matplotlib.pyplot as plt img = plt.imshow(test_image)
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=classifierLoad.predict(test_image) prediction = result[0]
classes=training_set.class_indices classes prediction=list(prediction)
prediction classes=['COVID19', 'NORMAL', 'PNEUMONIA']
output=zip(classes,prediction) output=dict(output) output if
output['COVID19']==1.0 :    print("COVID19") elif
output['NORMAL']==1.0:    print("NORMAL") elif
output['PNEUMONIA']==1.0:    print("PNEUMONIA")

```