**Visvesvaraya Technological University**

Belagavi



### **A Mini Project Report on**

**“CONVOLUTIONAL NEURAL NETWORK BASED IMAGE CLASSIFICATION FOR MEDICAL APPLICATION”**

SUBMITTED BY:

**JAYANTH S USN:1NH19EC045**

**HUSSAIN PEERA USN:1NH19EC043**

**G NIRANJAN REDDY USN:1NH19EC037**

**K HARSHAVARDHAN USN:1NH19EC144**

Under the guidance of

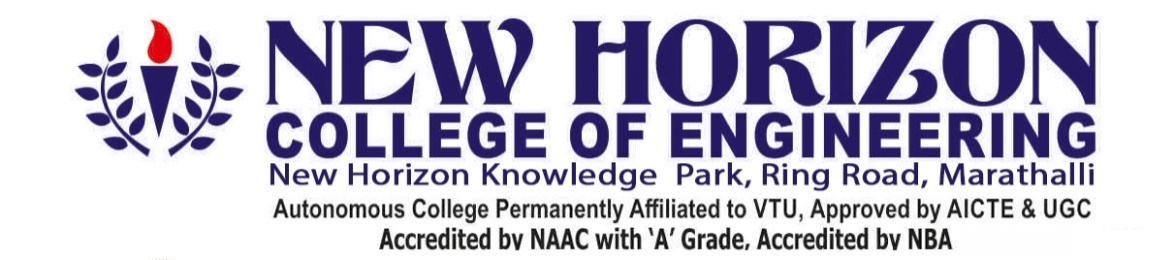
**Ms. Divya Sharma**

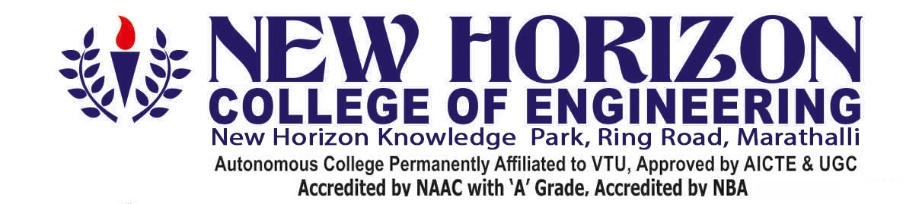
Sr. Assistant Professor, Dept. of ECE, NHCE, Bengaluru.

#### In partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING** IN **ELECTRONICS & COMMUNICATION**

**2021-22**

****

****

**BENGALURU-560103**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

CERTIFICATE

Certified that the mini project work entitled **“CONVOLUTIONAL NEURAL NETWORK BASED IMAGE CLASSIFICATION FOR MEDICAL APPLICATION”** carried out by **Jayanth S (1NH19EC045), HUSSAIN PEERA (1NH19EC043), G NIRANJAN REDDY (1NH19EC037), K HARSHA VARDHAN (1NH19EC144)** bonafide students of Electronics and Communication Department, NHCE, Bengaluru in partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication of the Visvesvaraya Technological University, Belagavi during the year 2020-21. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini-project report has been approved as it satisfies the academic requirements in respect of the mini-project work prescribed for the said degree.

Signature of the Guide Signature of the HoD

**Ms. Divya Sharma** **Dr. Sanjeev Sharma**

Sr. Assistant Professor Professor & HoD Department of ECE Department of ECE NHCE, Bengaluru NHCE, Bengaluru

External Viva

Name of the Examiners Signature with Date

1. 1.

2. 2.

**ACKNOWLEDGEMENT**

The satisfaction that accompanies the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement helped us succeed.

We thank **Dr. Mohan Manghnani**, Chairman of **New Horizon Educational Institution**, for providing the necessary infrastructure and creating a good environment.

We also record here the constant encouragement and facilities extended to us by **Dr.Manjunatha**, Principal, NHCE, and **Dr. Sanjeev Sharma**, head of the Department of Electronics and Communication Engineering. We extend sincere gratitude to them.

We sincerely acknowledge the encouragement, timely help, and guidance to us by our beloved guide **Ms. Divya Sharma** to complete the project within the stipulated time successfully.

Finally, a note of thanks to the teaching and non-teaching staff of the electronics and communication department for their co-operation extended to us, who helped us directly or indirectly in this successful completion of the mini-project.

**JAYANTH S (1NH19EC045)**

**HUSSAIN PEERA (1NH19EC043)**

**G NIRANJAN REDDY (1NH19EC037)**

**K HARSHAVARDHAN (1NH19EC144)**

**CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Chapter No. | Title | Page No. |
| 1 |  | ABSTRACT | Page No. 5 |
| 2 | **CHAPTER 1** | INTRODUCTION | Page No. 8 |
| 3 | **CHAPTER 2** | LITERATURE SURVEY | Page No. 10 |
| 4 | **CHAPTER 3** | 3.1 PROPOSED METHODOLOGY | Page No. 12 |
|  |  | 3.2 PROPOSED CNN MODELING | Page No. 20 |
| 5 | **CHAPTER 4** | PROJECT DESCRIPTION | Page No. 25 |
|  |  | SOFTWARE DESCRIPTION | Page No. 25 |
| 6 | **CHAPTER 5** | RESULTS AND DISCUSSION | Page No. 28 |
| 7 | **CHAPTER 6** | CONCLUSION AND FUTURE SCOPE | Page No. 34 |
|  |  | REFERENCES | Page No. 35 |
|  |  | APPENDIX | Page No. 36 |
|  |  | CODE | Page No. 37 |

**ABSTRACT**

In this mini-project, we work on designing a COVID-19 detection using the Convolutional Neural Network model with support of Open-Source software such as Keras, Python, Google Colab, Google Drive, Kaggle, and Visual Studio. All these Open-Source applications and software allow us to aggregate, design, create, train, visualize, and analyses bulk load of data on the cloud after programing a Deep neural network without a need for high-end processing hardware. The historical data that we will be using is Covid Chest X-Rays.

Here, Python is an interpreted high-level general-purpose programming language and Keras is an open-source software library that provides a Python interface for artificial neural networks. It is an API designed to reduce the complicity in the programming of a neural network program. Kaggle is an online community of data scientists and machine learning practitioners, from where we obtained the dataset of Covid-19 positive and Normal lung scan images for analysis. Google Colab is a free Jupyter notebook environment that runs entirely in the cloud, allows anyone with internet access and a Google account to write and execute, arbitrary Python code to complex Machine Learning and Deep Learning algorithms through the browser. It is well suited for machine learning, data analysis students as it is free. Colad is the platform where we will be modeling out CNN. We collect the data from the Kaggle dataset and process the data as per our requirement using Visual Studio and save it in Google Drive which will be mounted onto the Google Colabe notebook in which we will be modeling our CNN. Once modeled we can train the model using the data that has been saved in Google Drive to get the Weights of the CNN model. This Weights can be used to test, analyze the accuracy, visualize and predict the condition of a lung using chest X-Rays at certain accuracy. This will help in identifying the problems of the patients at a faster rate, thus giving an appropriate treatment at an early stage itself to saving one life.

**Keywords**: Convolutional Neural Network (CNN), Modelling, Keras API, Google Colab, Kaggle, Google Drive, Open Source, Covid-19 detector, Chest X-Ray

# LIST OF TABLES

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **No** | **Table No** | **TABLE DESCRIPTION** | **Page No** |
| 1 | 2 | Literature Survey | 10 |
| 2 | 3.1 | Data Distribution Table | 16 |
| 3 | 3.2 | Summary of the Model: | 21 |

# LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **No** | **Table No** | **FIGURE DESCRIPTION** | **Page No** |
| 1 | 3.1 | Pictorial Representation of CNN Model Being Designed | 12 |
| 2 | 3.2 | General Artificial Neural Network | 13 |
| 3 | 3.3 | General Convolutional Neural Network | 14 |
| 4 | 3.4 | Block Diagram Being Designed | 15 |
| 5 | 3.5 | RGB to RGB Matrix Conversion Image | 16 |
| 6 | 3.6 | Convolution (Conv2D) Pictorial Representation | 17 |
| 7 | 3.7 | Max-Pooling Pictorial Representation | 17 |
| 8 | 3.8 | ReLU Function Graph | 18 |
| 9 | 3.9 | Sigmoid Function Graph | 18 |
| 10 | 3.10 | Adam Optimizer Formula | 19 |
| 11 | 3.11 | Code implication of Adam optimizer | 19 |
| 12 | 3.12 | Pictorial Representation of CNN model | 20 |
| 13 | 3.13 | Flow Chart of the Program | 22 |
| 14 | 3.14 | Flow Chart of CNN Model | 23 |
| 15 | 3.15 | AlexNet Style Representation of Model | 24 |
| 16 | 5.1 | Interfacing and Predication of Image using CNN Model | 28 |
| 17 | 5.2 | Data classification is done by the program | 29 |
| 18 | 5.3 | Models Loss and Accuracy | 30 |
| 19 | 5.4 | Confusion Matrix | 30 |
| 20 | 5.5 | Wrong Predictions (Mismatching) | 31 |
| 21 | 5.6 | Accuracy V/S Loss Training Dataset | 31 |
| 22 | 5.7 | Accuracy V/S Loss Test Dataset | 32 |
| 23 | 5.8 | Accuracy of Training dataset V/S Test dataset | 32 |
| 24 | 5.9 | Accuracy of Test dataset V/S Test dataset | 32 |
| 25 | 5.10 | Data Sheet of Models Performance on Test Dataset | 33 |

# Chapter 1

**INTRODUCTION**

Syndrome coronavirus 2 (SARS-CoV-2) or shortly COVID-19 are family of viruses that can cause respiratory illness in humans was first discovered in 2019 in China Wuhan province. At present 2022 it is an ongoing global pandemic that has caused more than 300 million cases and 5.47 million deaths, making it one of the deadliest in history. Even though COVID-19 vaccines have been approved and widely distributed all over the world since December 2020, and many more preventive measures have been placing the effect of the virus is still prevailing significantly around the world causing havoc. One of the main reasons for this is the mutative nature of the viruses. Even though other symptoms of the disease vary from mutant to mutant to some extent, but still the most serious illness caused by this virus is related to the lungs such as Pneumonia, thus these cases can most commonly be diagnosed using the X-Ray imaging analysis for the abnormalities.

In short, X-Ray is a type of radiation called electromagnetic waves, which are used in creating X-Ray imaging to create a picture of the inside of our body, in the above-mentioned case lung scan (a type of nuclear imaging test). A tiny amount of radioactive matter called a tracer sends out gamma rays. These rays are picked up by the scanner to take a picture of the lungs.

The X-Ray imaging technique comes with great advantages which include its low cost, easy availability of X-Ray facilities, less time consumption, etc. Thus, X-ray imaging may be considered a better candidate for the mass, easy, and quick diagnosis procedure for a pandemic like COVID-19 considering the current global healthcare crisis, as the serious illness caused by this virus tends to remain constant.

Thus, in this project, we intended to demonstrate how we can apply neural networks i.e., in particular, CNN classification to differentiate a healthy human lung from one that is infected with covid-19 using these X-Ray images. The proposed model is developed to provide accurate diagnostics for binary classification (COVID Positive vs. Negative) predications on the basics of X-Ray images. Our model produced an average classification accuracy of 96.153% for binary classes.

This kind of system can be extensively modified for different end-user applications. For example, it could be used for Interstitial lung disease (ILD) detection, etc... In various places

* Ocular Disease Recognition using CNN
* Heart rate analyzing and diagnosis using CNN
* Health monitoring using Smart device data analysis
* Health care Automated monitoring system…

### Why use CNN in the project?

CNN is used in our project because it gives high accuracy when compared to other Neural Networks, also as CNN is mainly used for image classification, it can be used in medical and various other fields for the application of computer vision. With the help of a data set, we can give the necessary information to train a model to identify and classify the given data by self-learning what is what.

## Relation between CNN, AI, and Deep Learning:

Al in general means getting a computer to mimic human behavior in some way, Machine learning is a subset of Al which consists of the techniques that enable computers to figure out things from the given data and deliver it to Al for processing the information. Deep learning meanwhile is a subset of machine learning that enables computers to solve more complex problems.

In CNN, we have a powerful image processing capability, Al that uses deep learning to perform both generative and descriptive tasks often use as machine vision which comes under Image and Video recognition along with language processing stitch. These all are the application and process where the information used in the medical field for better and high accuracy purpose and help in living a better life.

# Chapter 2

**LITERATURE SURVEY**

**(Table No: 2)**

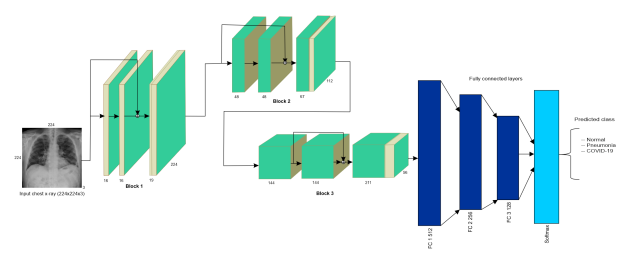
|  |  |  |  |
| --- | --- | --- | --- |
| No. | Article Title | Findings | Outcome |
| 1 | **Deep-chest: multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases** | Reference to architecture and dataset references. | Understood how the architecture should be and got references to datasets |
| 2 | **An Efficient CNN Model for COVID-19 Disease Detection Based on X-Ray Image Classification**  **(**[**https://www.hindawi.com/journals/complexity/2021/6621607/**](https://www.hindawi.com/journals/complexity/2021/6621607/)**)** | Architecture and Dataset references | The basic architecture and essential parameter references |
| 3 | **Coursera**  **Neural Networks and Deep Learning** | Theory of ANN and CNN | Understood the theoretical concepts of ANN and CNN in detail. |
| 4 | **A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way**  (<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>) | Understanding Convolutional Neural network | Learning  theoretical concepts of CNN with e.g. |
| 5 | **Kaggle Dataset.**  [(https://www.kaggle.com/search?q=covid+19+image+dataset)](https://www.kaggle.com/search?q=covid+19+image+dataset) | Datasets of Covid-19 and Normal lungs | We got an image dataset of covid-19 and Normal |
| 6 | Keras  (<https://keras.io/>) | Working and inbuilt function sheet of Keras API | Understanding how Keras work and how to program using it |
| 7 | Matplotlib  (<https://matplotlib.org/>) | Used to get function sheet of matplotlib lib | Application of matplotlib to data |
| 8 | OS  (<https://www.geeksforgeeks.org/os-module-python-examples/>) | How to interact with system OS using python | Understood how to handle files using Python and OS |
| 9 | Numpy  (<https://numpy.org/>) | Data handling functions lib | Handling data using inbuilt functions |
| 10 | scikit-learn  (<https://scikit-learn.org/stable/>) | Features various classification, regression, and clustering algorithms lib | Handling classification and accuracy auto-setup lib |
| 11 | Visual Studio  (<https://visualstudio.microsoft.com/>) | Visual studio software | Used to process data as per the requirement |
| 12 | Google Colab  (<https://colab.research.google.com/?utm_source=scs-index>) | Website for modeling the CNN | Cloud-based Hardware GPU for data processing and analysis |
| 13 | Deep Learning in Medical Imaging  (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6945006/>) | Application of Deep learning in medical image analysis | Advantage and use of applying Deep learning in medical image analysis |

# Chapter 3

# 3.1 Proposed Methodology

Identifying and giving the right diagnosis at a low cost to sickly people at the earliest is one of the biggest problems faced by the healthcare sector, due to the increasing number of problems such as covid-19, pneumonia, and many more in the field of lung disorders alone. Analysis of the medical reports and finding a proper diagnose to the patients it-self-consumes a lot of time, which can be crucial.

To solve this kind of issue while diagnosing COVID-19 concerned patients, we have proposed to use Deep Learning as a solution. Trained a Convolutional Neural Network (CNN), similar to the one referred to in fig 3.1, to identify the problem with high accuracy using chest X-ray images of sickly, we can reduce the time required for report analysis by a huge extent. This will help in identifying the problems of the patients at a faster rate, thus giving an appropriate treatment at an early stage itself to saving one life.



**(Fig 3.1) Pictorial Representation of CNN Model Being Designed**

The project aims to build an n-layer Convolutional neural network, which will be trained using the training dataset and get the weights of the network, which can be used on test-dataset for analysis of the CNN model performance and optimize it for better accuracy. Once done we should be able to use that network of medical applications to identify an unhealthy lung using **Binary Classification**.

## Neural Network:

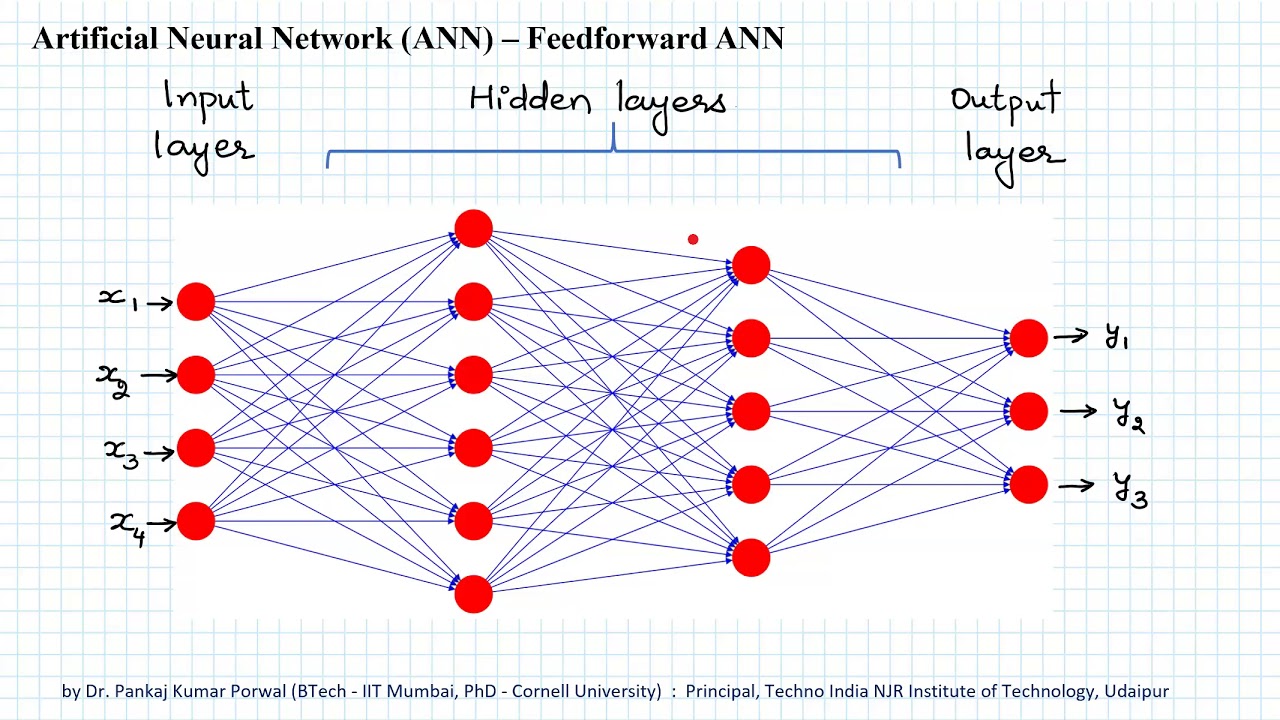
A neural network is a network of neurons (or) in the modern sense, an artificial neural network that is composed of artificial neurons nodes. It is similar to that of the biological neural network is composed of a group of chemically connected to a functioning associated neuron. A single neuron may be connected to many other neurons and form a network. The connection in a neuron is modeled in ANN as weights between nodes.

There are 3 types in a Neural network:

1. Artificial Neural Network or Standard Neural Network
2. Convolutional Neural Network
3. Recurrent Neural Network

In are using Supervised of Unstructured Data in our case.

## Artificial Neural Network:



**(Fig 3.2) General Artificial Neural Network**

As we can see the general representation of the neural network in figure 3.2, where if X is the input that we are given and y is the output that we get out, then all the other things in between these 2 things which are helping to determine the Y using X are called Hidden layers. All of these networks together form Artificial Neural Network.

ANNS are Composed of multiple nodes which imitate biological neurons of the human brain. The neurons are connected by links and interact with each other. The nodes can take input data and perform a simple program. The result is passed to other neurons and the output of each node is activation (or) node valves.

## Convolutional Neural Network:

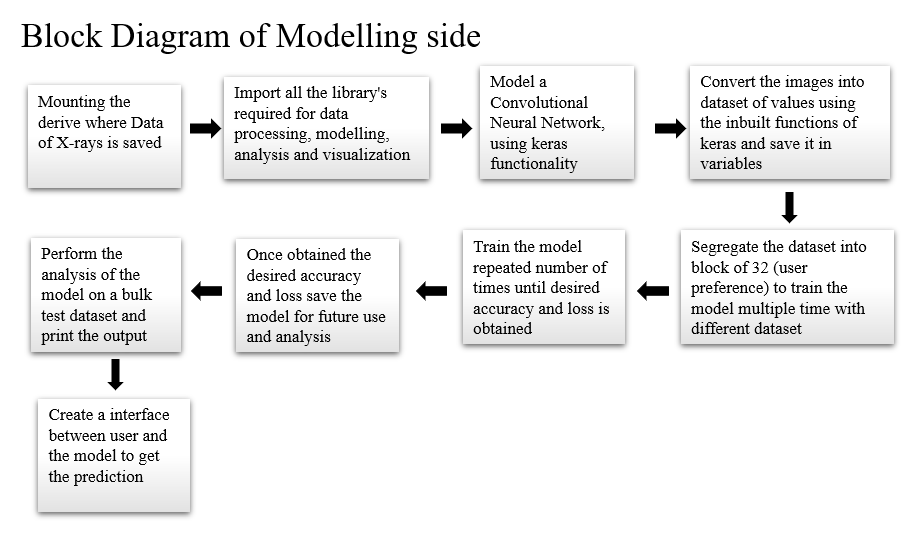
## Building a Convolutional Neural Network: Male 👨 vs Female 👩 | by Shadab Hussain | Towards Data Science

**(Fig 3.3) General Convolutional Neural Network**

A CNN is a type of ANN which is used in image processing and image recognition that is specifically used for processing pixel data. CNN is a powerful image processing tool that can perform both generative and descriptive tasks which are often used by machine learning for image and video analysis.

In CNN we first process the data to lower order through various mathematical tools, then the data is flattened and given to a classical ANN network for further analysis.

## Block Diagram of Proposed System:

****

**(Fig 3.4) Block Diagram**

## Importing Libraries:

The process of creating a link between the libraries we are going to use in the program while programming such as Keras, matplotlib, etc.

Imported Libraries:

1. Numpy
2. Matplotlib
3. OS
4. Keras
   * Layer 🡪 Conv2D, MaxPool2D, Dropout, Flatten, Dense
   * Image 🡪 image, ImageDataGenerator
5. Sklearn
   * Metrics 🡪 classification\_report, Confusion\_matrix
6. Seaborn
7. TensorFlow

## Mounting the Drive to the Program:

Creating a connection between the drive-in which we have saved our image dataset and the program in which we are creating our CNN model

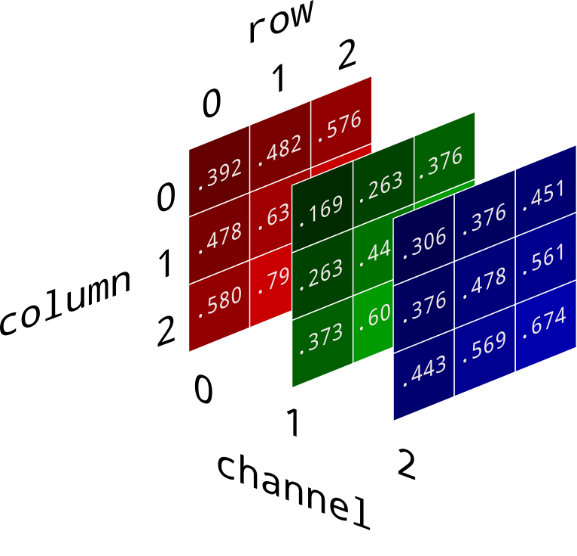
## Data Gathering and Segmentation:

Data gathering is one of the most important aspects of creating a model, that is because the quality and quantity of the dataset impact the output of the model to a great extent. Balancing Dataset is important to improve the performance of the proposed CNN models in the detection of COVID-19, we have gathered a total of 489 lung X-Ray images of two classes, 282 images of COVID-19 positive lung X-Ray images, and 207 images of Normal lung X-Ray images. These 489 images are divided into 2 sets, Training (242 positive, 242 normal) images and Test (39 positive, 39 normal) image dataset. These concatenated extra X-ray images were downloaded from Kaggle.The process of dividing the available dataset into groups such that it can be used to train the model with different datasets for every epoch data segmentation.

**(Table No: 3.1)** **Data Distribution Table**:

|  |  |  |
| --- | --- | --- |
|  | **Covid-19 Positive X-Ray Image** | **Normal X-Ray Image** |
| **Training set** | 242 | 242 |
| **Test set** | 39 | 39 |

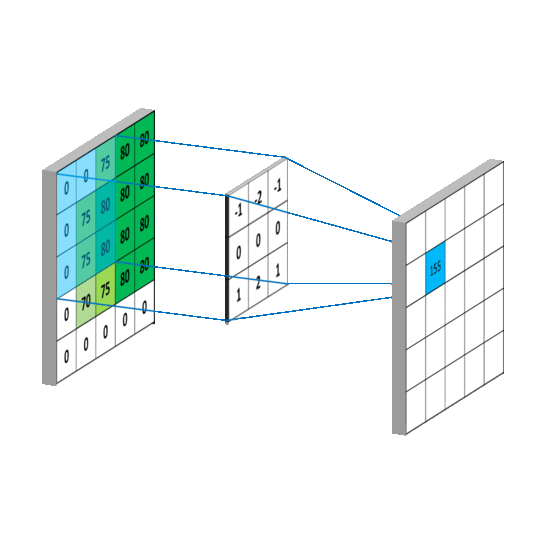
## Processing the Data:

1. ***Image to data conversion***:

**(Fig 3.5) Image Conversion from RGB to RGB Matrix**

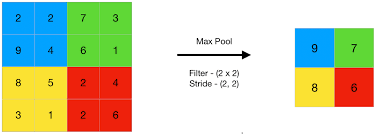
The process of converting the image (RGB image) into a matrix of shape (width, height, depth) dataset with a respective pixel value between 0 - 255 can be seen in figure 3.6

1. ***Convolutional (Conv2D):***



**(Fig 3.6) Convolution (Conv2D) Pictorial Representation**

The process of simple application of a filter onto an input that results in activation, which is simply known as Convolution is shown in figure 3.7. Repeated application of the same filter to an input, results in a map of activations called a feature map, indicating the locations and strength of a detected feature in input, such as an image.

1. **Max-Pooling:**

**(Fig 3.7) Max-Pooling Pictorial Representation**

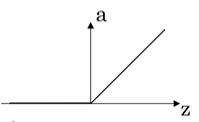
Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map. One such example is shown in figure 3.8.

## **Flatten Layer:**

The process of converting a matrix of order M\*N into (M, N\*1) that is into a column matrix.

Example: 🡪

## **ReLU activation functions:**

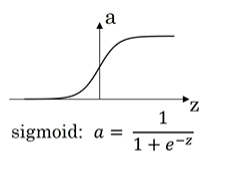


The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The graphical of ReLU is given the figure 3.9

ReLU: **(Fig 3.8) ReLU Function Graph**

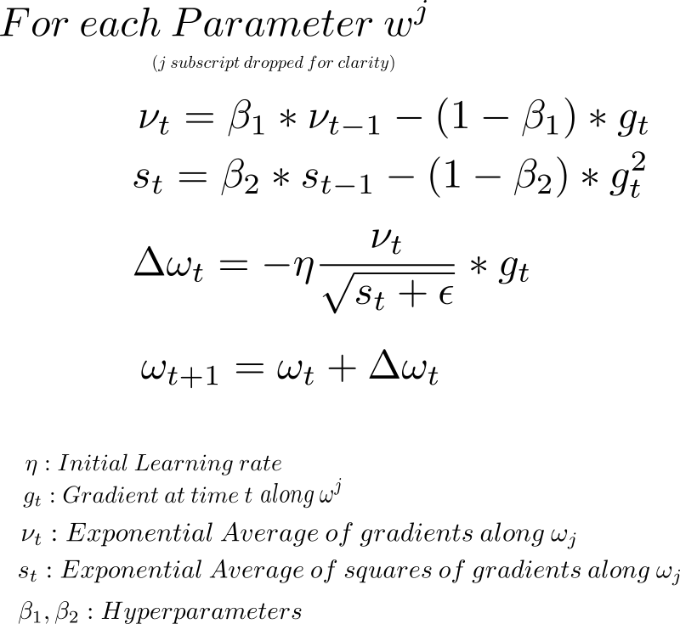
Derivative:

## **Sigmoid activation function:**

The main reason why we use the sigmoid function is that it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. As our output is a yes or no type of classification that is binary classification in the last stage, we make it a sigmoid function in the last layer of CNN. The Graph and function of the Sigmoid activation function is given in figure 3.10

**(Fig 3.9) Sigmoid Function Graph**

## **Adam Optimizer:**

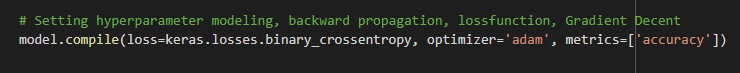


**(Fig 3.10) Adam Optimizer Formula**

Adam optimizer involves a combination of two gradient descent methodologies: Momentum: This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima at a faster pace.

## Parameter Setup:

Varies parameters have to be set up while modeling CNN before training it such as input size is set to (150, 150,3) i.e., any image input will be resized to this configuration, the parameter of Initial, Learning rate, Exponential Average of gradients, Exponential average of squares of gradients, Hyperparameters are auto-configure by the **Adam** optimizer in the program.



**(Fig 3.11) Code implication of Adam optimizer**

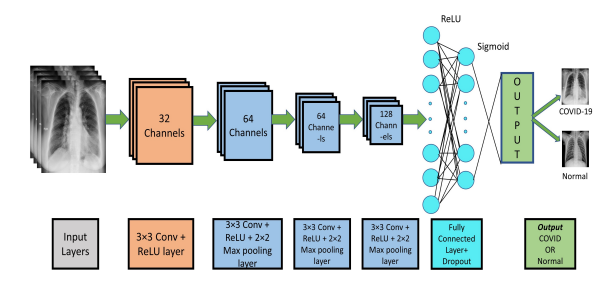
## Model analysis:

The process of analyzing the accuracy and the reliance of the model using the test dataset for a better understanding of the trained model and visualization of the output model is known as model analysis.

## Creating an Interface:

Creating a method where the use a give his image to get the predication whether it is covid-19 positive or Negative, and the predication is displayed on to the screen along with the image.

# Proposed Methodology



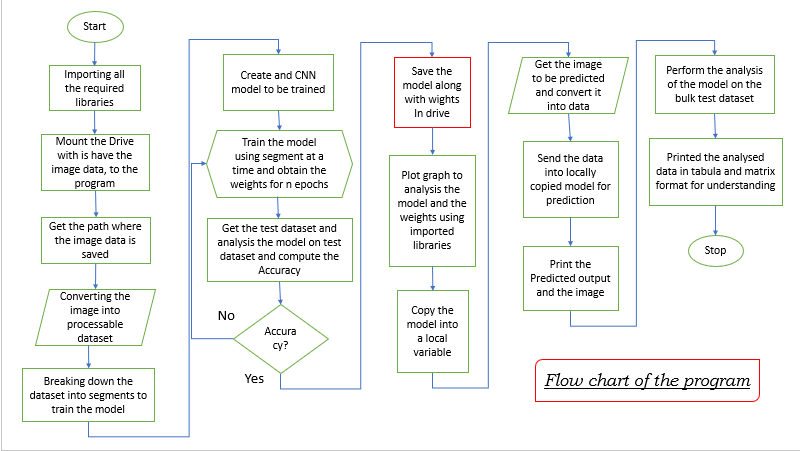
**(Fig 3.12) Pictorial Representation of CNN model**

The Process of Creating a Model of Convolutional Neural network by setting the flow of the model as shown in the figure and setting the essential parameters to train the model is known as modeling CNN.

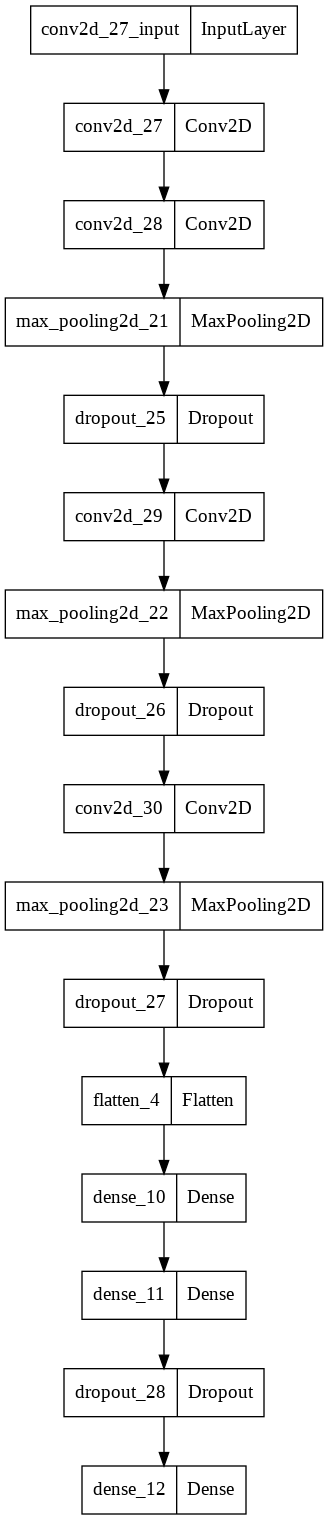
The Proposed CNN model consists of 16 layers in which 4 are convolutional (Conv2D), 4 max-pooling layers, 4 dropout layers, 6 activation layers (5 ReLU layers and final one sigmoid layer), 56 batch normalization layer for the training set and 1 flatten layer, and 2 fully connected layers; CNN model input image shape is (150, 150, 3), i.e., 150-by-150 RGB image. In all the convolutional layers, a 3\*3 size kernel has been used by the filter size after every layer change (32, 64, 128, 256) respectively. After each con2D layer, the Max-pooling layer with a 2\*2 pooling size has been used. The activation layer has been used with the ReLU function, and the dropout layer has been used with a 25% dropout rate. The output of 65536 neurons of the final Con2D layer is flattened into a column matrix, a flattening layer has been used do this process, this flattened layer will be the input next 3 dense layers of order 256, 128, 1 respectively. Since only one output node is needed to classify the data to one of 2 classes we make use of Binary classification by giving the output to a sigmoid activation function. The output given by the sigmoid activation function lies between 0 and 1.

**(Table No: 3.2)** ***Summary of the Model:***

|  |  |  |
| --- | --- | --- |
| Model: "sequential" | | |
| Layer (type) | Output Shape | Param # |
| conv2d (Conv2D) | (None, 148, 148, 32) | 896 |
| conv2d\_1 (Conv2D) | (None, 146, 146, 64) | 18496 |
| max\_pooling2d (MaxPooling2D) | (None, 73, 73, 64) | 0 |
| dropout (Dropout) | (None, 73, 73, 64) | 0 |
| conv2d\_2 (Conv2D) | (None, 71, 71, 128) | 73856 |
| max\_pooling2d\_1 (MaxPooling2D) | (None, 35, 35, 128) | 0 |
| dropout\_1 (Dropout) | (None, 35, 35, 128) | 0 |
| conv2d\_3 (Conv2D) | (None, 33, 33, 128) | 295168 |
| max\_pooling2d\_2 (MaxPooling2D) | (None, 16, 16, 256) | 0 |
| dropout\_2 (Dropout) | (None, 16, 16, 256) | 0 |
| flatten (Flatten) | (None, 65536) | 0 |
| dense (Dense) | (None, 256) | 16777472 |
| dense (Dense) | (None, 128) | 32896 |
| dropout\_2 (Dropout) | (None, 128) | 0 |
| dense\_1 (Dense) | (None, 1) | 129 |
| Total params: 17,198,913  Trainable params: 17,198,913  Non-trainable params: 0 | | |

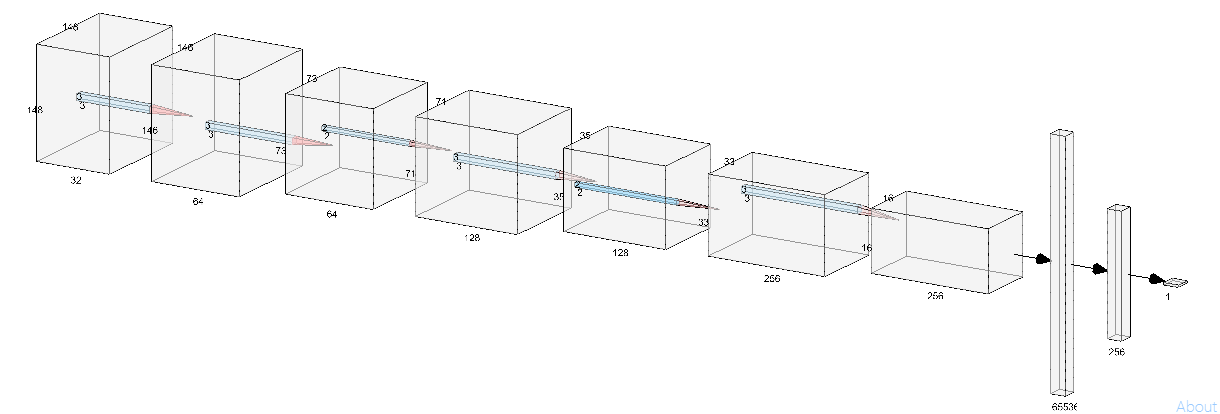
***Flow Chart of the Program:***

**(Fig 3.13) Flow Chart of the Program**

***Flow Chart of the CNN Model***.

**(Fig 3.14) Flow Chart of CNN Model**

***AlexNet Style Representation of Model*** :



**(Fig 3.15) AlexNet Style Representation of Model**

# Chapter 4

# Project Description

4.1 Software Description:

Google Colab

Google Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute code in python, document our code that supports mathematical equations, and is especially well suited to machine learning, data analysis, and education. Colab is a hosted Jupyter notebook service that requires no setup to use, just need a google account to login to the service and we can start working on it. It provides free time-restricted access to computing resources such as GPUs and TPU.

It comes with Pre-Installed Libraries, Saved on the Cloud, Collaboration. Free GPU and TPU use Anaconda distribution of Jupyter Notebook along with libraries, such as Pandas, NumPy, Matplotlib, and also pre-installed machine learning libraries such as Keras, TensorFlow, and PyTorch, etc. We can execute our entire model in this platform (Google Colab), utilizing the GPU provided by the platform with ease.

Google Colab: <https://research.google.com/colaboratory/>

Keras

Keras is an open-source software Neural Network library with high-level APIs written in python, it is a simple, flexible and powerful library for implementing Deep Neural Networks. It is a high-level neural network library that is running on top of TensorFlow, CNTK, and Theano. Using Keras in Deep Learning modeling allows for fast prototyping as well as running seamlessly on CPU and GPU. The framework is written in python code which is easy to debug and program.

Keras allows us to design deep neural models and allows distributed training of those models on GPUs or TPUs.

Keras: <https://keras.io/>

Numpy

Numpy is a core library used for working with array and scientific computing in python, its community-driven open-source project. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. It provides a high-performance multidimensional array of objects and tools. It plays a role in the machine learning library as it has a lot of useful inbuilt libraries.

Numpy: <https://numpy.org/>

Matplotlib

Matplotlib is an amazing Open-Source visualization, plotting library in python programming language for 2D plots of arrays. It is a multiple-platform data visualization library, it comprehensive for creating static, animated, and interactive visualizations in python.

Matplotlib: <https://matplotlib.org/>

Visual Studio

Visual Studio Code is a standalone source code editor i.e., integrated development environment (IDE) from Microsoft. It is used for different types of software developments such as websites, web pages, applications, analysis tools, etc, Features such as IntelliSense suggestions and syntax checking in the editor make it popular.

Visual Studio: <https://visualstudio.microsoft.com/>

Scikit-learn

Scikit-learn is the most useful Open-Source and robust library built on Numpy, SciPy, and matplotlib for machine learning in python which provides efficient tools for predictive data analysis, machine learning, and statistical modeling including classification, clustering, regression, and dimensionality reduction via a consistence interface in python.

Scikit-Learn: <https://scikit-learn.org/stable/>

Seaborn

Seaborn is a data visualization tool i.e., a library built on top of matplotlib, and is integrated with pandas data structures in python. It helps in the exploration and understanding of data. One has to be familiar with Numpy and Matplotlib and pandas use seaborn.

TensorFlow

TensorFlow is an Open-Source platform for machine learning and artificial intelligence library, using data flow and graphs to build models. It is mainly used for Classification, Perception, Understanding, Discovering, Prediction and Creation. It allows developers to create large-Scale Neural networks with many layers with a comprehensive, flexible ecosystem of tools, libraries, and interfaces.

TensorFlow: <https://www.tensorflow.org/>

Kaggle

Kaggle platform which allows users to find the published data sets offers a no-setup, customizable, Jupyter Notebooks environment. Users can explore and build models in a web-based data-science environment, work with other machine learning engineers, data scientists, and enter competitions to solve data science challenges.

Kaggle: <https://www.kaggle.com/>

**Chapter 5**

# Results and Discussion

A CNN model is designed and trained using Keras in Google Colab. The accuracy and precession are verified through testing the trained model using. A sample image of both covid positive and normal lung X-Ray image is passed and the prediction is verified. Confusion Matrix, Accuracy v/s Loss dataset graphs for both test and training data set, for better understanding and analysis of the model.

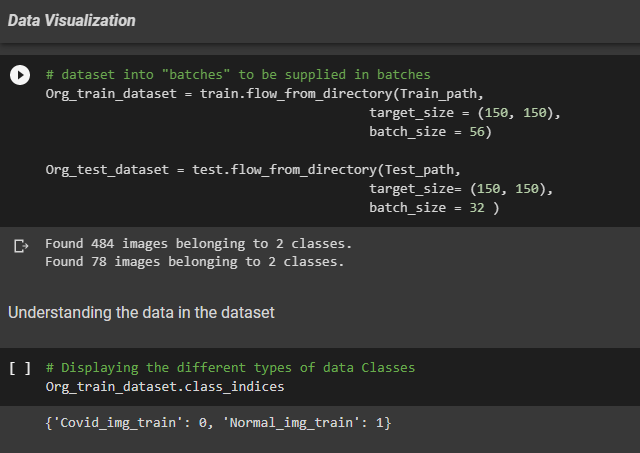
# 

**(Fig 5.1) Interfacing and Predication of Image using CNN Model**

The proposed CNN model is capable of classification or detecting Covid-19 X-Ray images and Normal lung X-Ray images when the trained model is supplied with the X-Ray image of the patients for predicting, which we can see in figure 5.1 where it is predicting the conduction of the given X-Ray lung image.

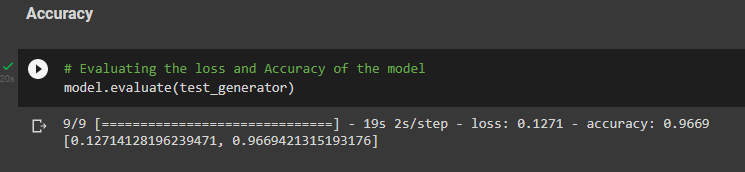
Varies parameters are calculated while training the model and testing, they are shown in the following report.

***Data Classification:***



**(Fig 5.2) Data classification is done by the program**

While pre-processing the dataset of around 562 X-Ray Images, for testing and training the proposed CNN, the dataset was divided into 2 classes. The training dataset contains 242 COVID-19 positive X-Ray images and 242 Normal X-Ray images, making a total of 484 training images. The test dataset similarly, contained 39 COVID-19 positive and Normal images each, making a total of 78 test dataset images. We can see how the program is understanding the classification of training and test dataset given by use in figure 5.2.

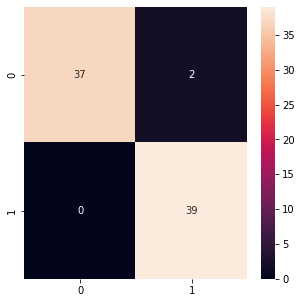


**(Fig 5.3) Models Loss and Accuracy**

In figure 5.3 we see that the CNN model achieved performance with an accuracy of 96.69% with the test validation dataset, during the process of training the model and testing it with the validation dataset. We can also see that the loss of the model at end of the training has reduced to 0.127.

## ***Confusion matrix:***

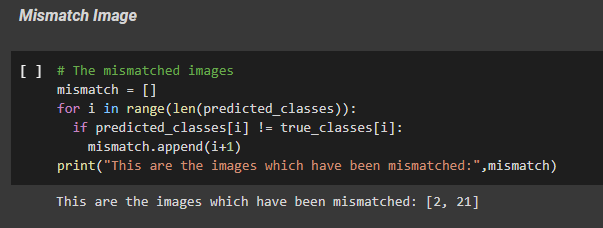
Figure 5.4 shows the Confusion matrix of the model, according to which the CNN model tested for 78 images, out of which 2 have been predicted wrongly.

****

**(Fig 5.4)** Confusion Matrix

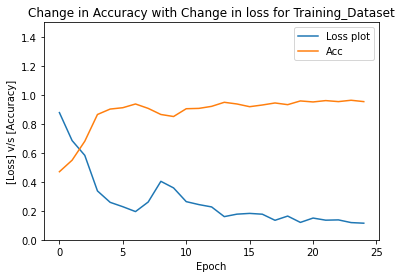
The Y-axis represents the True classification and X-axis Predicated classification where 0 indicates COVID-19 positive and 1 indicates COVID-19 negative. As we can see the model was able to predict all the True COVID-19 negative samples correctly, were as coming to COVID-19 positive samples it predicted 37 cases correctly but gave 2 false predictions saying that COVID-19 positive cases as negative.

Figure 5.5 shows the Cases which have been predicted wrongly by the model. In the output, we can see the image identity numbers, which have been predicted wrongly.



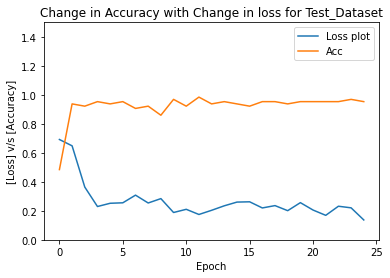
**(Fig 5.5) Wrong Predictions (Mismatching)**

***Graphical Analysis***:



**(Fig 5.6) (Accuracy V/S Loss Training Dataset)**

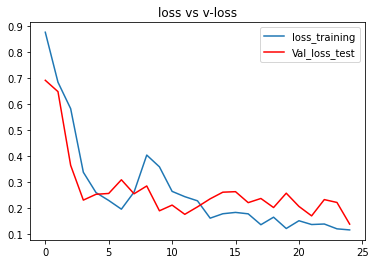
Figures 5.6, 5.7, 5.8, and 5.9 are Graphical Visualization of the curve of the accuracy and loss for training and test dataset and their interrelation, we can see how as the loss decreases the accuracy of the system is increasing respectively with each epoch, and also with the increase in an epoch the accuracy is also increasing respectively, after a certain epoch the accuracy tends to get saturated which can be observed in the graphs after 15 epochs.



**(Fig 5.7) (Accuracy V/S Loss Test Dataset)**

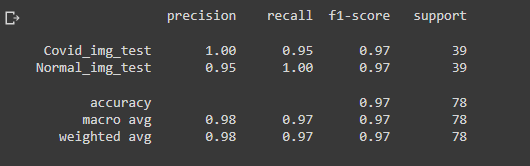


**(Fig 5.8) (Accuracy of Training dataset V/S Test dataset)**



**(Fig 5.9) (Accuracy of Test dataset V/S Test dataset)**

***Data Sheet****:*

**

**(Fig 5.10) Data Sheet of Models Performance on Test Dataset**

The above give report (figure 5.10) we can see shows the model is performing on the bulk dataset of 78 images, out of which 39 are COVID-19 positive and 39 are negative. The model can predict the COVID-19 positive images at 100% precision but when coming to the Normal or COVID-19 negative images the precision is only 95%. The overall accuracy of the model on the test dataset is around 97%.

# Chapter 6

**Conclusion and Future Scope**

In a world dominated by digital technology, Artificial intelligence built on top of Machine learning and Deep Learning plays a prominent role in our lives. It has created an ecosystem that links many to give a smart and impressive performance in every task. Thus, in such a case using this type of rising technology in the medical field can save many lives. We don’t need sophisticated, costly equipment to diagnose the patient.

In a pandemic like this in which we are leaving (2022), diagnosing the patient at the early stages of the problem or spreading of the virus present in one’s body can be greatly helpful, and also the system that is been used should be able to adapt with ever-changing mutations from time to time. Thus, using Artificial intelligence built on top of Machine learning and Deep Learning can be used to solve these problems at a fast rate and high accuracy, thus in the process helping millions of lives in million ways, directly and as well as indirectly to save lives.

Hence, we conclude CNN models proposed, can derive meaningful information based on data analytics, share the data on the cloud, and analyze it safely to give the respective output at a lightning speed, thus making a huge difference in how we tackle a problem and solve it to help millions.

This mini project is a simple demonstration of how we can implement this Deep Learning Convolutional Neural Network (CNN) using a few Open-Source libraries for analysis of lung X-Ray scans. This has a variety of applications like Ocular Disease Recognition using CNN, Heart rate analysis and diagnosis using CNN, Health monitoring using Smart device data analysis, Health care Automated monitoring system, and many more. This can be applied in other filed too where image analysis is required.

We can further implement this modeling scheme in other fields of medical application too, and also, we can improve the accuracy and loss rates. Other more powerful CNN models, such as ResNetv2 and ensemble of the multiple CNN models have not been evaluated, but they could improve the results, thus we can utilize this model to do predictions also and check the model’s performance; Visualization can also be added to improve the understanding and explanation of the results of the CNN based system because those are essential for the adoption of a CNN based system in real clinical application.

# References

[1] “<https://colab.research.google.com/?utm_source=scs-index>”, use this link to model, train, analysis and visualize the datasets, Google Colab.

[2] “<https://visualstudio.microsoft.com/>”, Refer to this link for installation and setup of Visual Studio.

[3] “**Coursera**-**Neural Networks and Deep Learning**”, Refer the video of the to understand Deep Learning and CNN

[4] “<https://www.kaggle.com/search?q=covid+19+image+dataset>”, the website from where we got the dataset of Covid-19 X-ray lung images and Normal X-ray lung images

[5] “[**https://www.hindawi.com/journals/complexity/2021/6621607/**](https://www.hindawi.com/journals/complexity/2021/6621607/)”, Refer to the architecture and modeling.

[6] “<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>”, visit the official website better understanding of CNN.

[7] “<https://link.springer.com/article/10.1007/s12065-020-00540-3#:~:text=Researchers%20have%20successfully%20applied%20CNNs,%5D%2C%20blood%20cancer%2C%20anomalies%20of>”, visit the “Springer Link” website for the information about the application of CNN in medical image understanding

[8] “<https://github.com/Jayanth9601/COVID-19_Detection_using_CNN>”, for the dataset and code used in this project, refer to the link given.

**Appendix**

*Google Colab with Keras*

**Keywords**

Here are some keywords that are used while modeling CNN using Keras.

***ReLU*** – It is an activation function that is used after every convolution layer.

***Sigmoid***– It is an activation function that is used at the end of the model for Binary classification networks.

***Convolution*** – The process of applying the filter to a matrix for activation of it.

***Dense*** – It is a keyword used to create an ANN, this command auto-tunes the model of getting approximate weights, learning rate, forward propagation, and backward propagation.

***flatten*** – The process of converting a Metrix into a column matrix to apply to the Dense ANN.

***Accuracy*** – the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

***Loss*** – Loss is the penalty for a bad prediction. That is, the loss is a number indicating how bad the model's prediction was on a single example.

***Precision --*** Precision is one indicator of a machine learning model's performance, Precision refers to the number of true positives divided by the total number of positive predictions.

*Code Review:*

## Code of Modelling, Training, Testing, Predicting and Visualizing dataset

|  |
| --- |
| """\*\*\*Covid-19 Detection Using CNN\*\*\*"""  # numpy used to Data Handling  import numpy as np  # Used for Graph Generation  import matplotlib.pyplot as plt  # Used for file handling  import os  """library for Deeplearning keras"""  import keras  # Used to creat model in concatination (one\_layer\_ofter\_other)  from keras.models import Sequential  # All the needed tooles to creat model  from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense  # Used conver image into dataset  from keras.preprocessing.image import image, ImageDataGenerator  import sklearn.metrics as metrics  # Used to set the parameters and hyperparameters  from sklearn.metrics import classification\_report, confusion\_matrix  # Python data visualization library  import seaborn as sns  # Used to save and load the developed models  import tensorflow as tf  """Path of the folders where the trainging and test dataset are saved"""  # loading the path were the training dataset is saved  Train\_path = "< Location\_of\_the\_train\_dataset >"  # loading the path were the test dataset is saved  Test\_path = "< Location\_of\_the\_test\_dataset >"  """Mounting the Drive where the Image data has been saved"""  from google.colab import drive  drive.mount('/content/drive')  """\*\*\*Image and Data visualization\*\*\*"""  """\*\*\*Image Visualization\*\*\*"""  # Displaying a image form our dataset  # path of the location where the image is saved  path = "< Location\_of\_the\_image\_to\_be\_displayed (lung image) >"  # Path, target\_size = {size at which the image is displayed}  img = image.load\_img(path, target\_size=(256,256))  # showing the image  plt.imshow(img)  """\*\*\*Image to data Conversition\*\*\*"""  # Converting RGB image into Dataset (array) and scaling them down by (1/255)  # Taining dataset  train = ImageDataGenerator(rescale=1/255)  # Test dataset  test = ImageDataGenerator(rescale=1/255)  """\*\*\*Data Visualization\*\*\*"""  # Seperating our dataset into "batches" to be supplied in groups  Org\_train\_dataset = train.flow\_from\_directory(Train\_path,  target\_size = (150, 150),  batch\_size = 32)    Org\_test\_dataset = test.flow\_from\_directory(Test\_path,  target\_size= (150, 150),  batch\_size = 32 )  """Understanding the data in the dataset"""  # Displaying the different types of data  Org\_train\_dataset.class\_indices  # Visualizing the difference between the data  Org\_train\_dataset.classes  """\*\*\*CNN Basic Model in Keras\*\*\*"""  """Creating a CNN model"""  # Creating a model named "model" which will be sequential  model = Sequential()  # Input layer (Raw Image)  # First layer (Input layer), 1st Convolution layer with 32 filters of size 3\*3, relu activation  model.add(Conv2D(32,kernel\_size=(3,3), activation='relu',  input\_shape=(150,150,3)))    # Second layer of Convolution  # 2nd layer, 2st Convolution layer with 64 filters of size 3\*3, relu activation  model.add(Conv2D(64,(3,3), activation='relu'))  # Pooldown layer of size 2\*2  model.add(MaxPool2D(2,2))  # fitting the model  model.add(Dropout(0.25))  # Third layer of Convolution  # 3rd layer, 3rd Convolution layer with 128 filters of size 3\*3, relu activation  model.add(Conv2D(128,(3,3), activation='relu'))  model.add(MaxPool2D(2,2))  model.add(Dropout(0.25))  # Fourth layer of Convolution  # 4rd layer, 4rd Convolution layer with 256 filters of size 3\*3, relu activation  model.add(Conv2D(256,(3,3), activation='relu'))  model.add(MaxPool2D(2,2))  model.add(Dropout(0.2))  """Data Flattening"""  #flattening the data into Column matrix  # Converting the matrix into column matrix  model.add(Flatten())  # Dense layer with relu activation  model.add(Dense(256,activation='relu'))  model.add(Dense(128,activation='relu'))  # fitting the model  model.add(Dropout(0.5))  # Final output layer with sigmoid activation "Binary Classification"  model.add(Dense(1,activation='sigmoid'))  # Setting hyperparameter modeling, backward propagation, lossfunction, Gradient Decent  model.compile(loss=keras.losses.binary\_crossentropy,  optimizer='adam', metrics=['accuracy'])  # Getting the Summary of the designed model with parameter of respective layers  model.summary()  """Flowchart"""  # Getting the Flow chart of the designed model  from keras.utils.vis\_utils import plot\_model  plot\_model(model)  """\*\*\*Training the Model\*\*\*"""  #Train from scratch  train\_datagen = image.ImageDataGenerator(  rescale = 1./255,  shear\_range = 0.2,  zoom\_range = 0.2,  horizontal\_flip = True,  )  test\_dataset = image.ImageDataGenerator(rescale=1./255)  # Conveting RGB images into dataset in batch of 56 images at a time  train\_generator = train\_datagen.flow\_from\_directory(  Train\_path,  target\_size=(150,150),  batch\_size=56,  class\_mode='binary',  )  # Conveting RGB images into dataset in batch of 32 images at a time  test\_generator = test\_dataset.flow\_from\_directory(  Test\_path,  target\_size=(150,150),  batch\_size=32,  class\_mode='binary'  )  hist = model.fit(  train\_generator,  # Total number of steps (batches of samples) to yield from generator  steps\_per\_epoch=8,  # an instant in time chosen as the origin of a particular calendar era  epochs=15,  # test the test datset  validation\_data = test\_generator,  validation\_steps=2,  )  """\*\*Accuracy\*\*"""  # Evaluating the loss and Accuracy of the model  model.evaluate(test\_generator)  """\*\*\*Saving the generated model\*\*\*"""  # Creating a folder to save the model generated  model.save('< Location\_where\_the\_train\_model\_to\_be\_save\_in\_".h5"\_formate >')  print("model Save with name <model.h5>.")  """\*\*\*Graphical Representation of Ouput\*\*\*"""  # Used to Generat the graph  h = hist.history  # Checking different parametes that we obtained after training the model  h.keys()    """Accuracy Graph With respect to no of iterations"""  # Grpah of the Accuracy of Training dataset v/s Test dataset  plt.plot(h['accuracy'], label="Accuracy\_training")  plt.plot(h['val\_accuracy'] , c = "red", label="Val\_accuracy\_test")  plt.title("acc vs v-acc")  plt.legend()  plt.show()  """Loss Graph With respect to no of iterations"""  # Grpah of the Loss of Training dataset v/s Test dataset  plt.plot(h['loss'],label="loss\_training")  plt.plot(h['val\_loss'] , c = "red",label="Val\_loss\_test")  plt.title("loss vs v-loss")  plt.legend()  plt.show()  """\*\*\*Accuracy V/S Loss Graph\*\*\*"""  #Graph of Accuracy v/s loss of Training dataset  epochs=15  plt.figure()  plt.title("Change in Accuracy with Change in loss for Training\_Dataset")  plt.xlabel('Epoch')  plt.ylabel('[Loss] v/s [Accuracy]')  plt.plot([i for i in range(epochs)], h['loss'], label='Loss plot')  plt.plot([i for i in range(epochs)], h['accuracy'], label = 'Acc')  plt.legend()  plt.ylim([0,1.5])  #Graph of Accuracy v/s loss of Test dataset  plt.figure()  plt.title("Change in Accuracy with Change in loss for Test\_Dataset")  plt.xlabel('Epoch')  plt.ylabel('[Loss] v/s [Accuracy]')  plt.plot([i for i in range(15)], h['val\_loss'], label='Loss plot')  plt.plot([i for i in range(15)], h['val\_accuracy'], label = 'Acc')  plt.legend()  plt.ylim([0,1.5])  """Output checking"""  """\*\*Loading the saved Model\*\*"""  # Loading the saved model from memory  loaded\_model = tf.keras.models.load\_model('/content/saved\_model\_Moredata/covid.h5')  #printing the size of the input layer of the model  loaded\_model.layers[0].input\_shape  """\*\*\*prediction models\*\*\*"""  # Path of the image which has to be tested  image\_path = "< Location\_of\_the\_Image\_which\_has\_to\_pridicted >"  img = image.load\_img(image\_path, target\_size=(150, 150))  plt.imshow(img)  img = np.expand\_dims(img, axis=0)  result=loaded\_model.predict(img)  # printing the output whether the image is of Covid or Non-Covid image  if result[0][0] == 0.0:  print("Covid-19 Positive.")  else:  print(result[0][0])  print("Covid-19 Negative.")  plt.show()  """\*\*\*Model Visulization of Test Dataset\*\*\*"""  # Predicated output values from the image  predicted\_classes = []  positive = "< Location\_of\_the\_test\_COVID\_positive\_dataset >"  normal = "< Location\_of\_the\_test\_COVID\_Negative\_dataset >"  for filename in os.listdir(positive):  img = image.load\_img(positive + '/' + filename, target\_size=(150, 150))  img = np.expand\_dims(img, axis=0)  result=loaded\_model.predict(img)  predicted\_classes.append(int(result[0][0]))  for filename in os.listdir(normal):  img = image.load\_img(normal + '/' + filename, target\_size=(150, 150))  img = np.expand\_dims(img, axis=0)  result=loaded\_model.predict(img)  predicted\_classes.append(int(result[0][0]))  print(predicted\_classes)  predicted\_classes = np.array(predicted\_classes)  # Orginal output values  true\_classes = Org\_test\_dataset.classes  true\_classes  """\*\*\*Mismatch Image\*\*\*"""  # The mismatched images  mismatch = []  for i in range(len(predicted\_classes)):  if predicted\_classes[i] != true\_classes[i]:  mismatch.append(i+1)  print("This are the images which have been mismatched:",mismatch)  """\*\*\*Report\*\*\*"""  class\_labels = list(Org\_test\_dataset.class\_indices.keys())  # Report of the tested dataset analysis  report = metrics.classification\_report(true\_classes, predicted\_classes, target\_names=class\_labels)  print(report)  """\*\*\*Confusion Matrix\*\*\*"""  conf\_matrix = confusion\_matrix(true\_classes, predicted\_classes)  plt.figure(figsize = (5,5))  sns.heatmap(conf\_matrix, annot=True)  """\*\*\*Thank You\*\*\*""" |

## Code for Data segregation

|  |
| --- |
| # Used to Read data from CSV files  import pandas as pd  # Used to handle files and DIR  import os  # Used to copy the Images  import shutil  # Create the data for positive samples  # Path where Metadata CSV file is saved  File\_path = "< Location\_of\_CSV\_file >"  # Path were the Images are saved  Images\_path = "< Location\_where\_the\_images\_are\_saved >"  # Reading the Matadata CSV file for Processing  df = pd.read\_csv(File\_path)  # Printing number of Rows and colums in the metadata table  print(df.shape)  # WE CAN CHECK THE CSV FILE USING THE COMMAND  # "df.head()"  # Path (folder) where the Images will be saved  Target\_DIR = "< Location\_where\_the\_image\_data\_should\_be\_saved >"  # Checking if the file were the images are to be copyed exists  if not os.path.exists(Target\_DIR):  # If doesn't exist Creating one  os.mkdir(Target\_DIR)  print("Covid folder created.")  co = 0  for (i,row) in df.iterrows():  # For matadata selecting only covide PA images  if row["finding"] == "COVID-19" and row["view"] == "PA":  # Copying filename  filename = row["filename"]  # Copying the Path were Image is saved  image\_path = os.path.join(Images\_path,filename)  # Making the Path were the Image has to be Copyed  image\_copy\_path = os.path.join(Target\_DIR,filename)  # Copying the Image to the Desired location (path\_in\_previous\_step)  shutil.copy2(image\_path,image\_copy\_path)  # Counting the number of Images being copyed  co += 1  print("Copied",co," to DIR.") |