1.INTRODUCTION

COVID-19 is an infectious and contagious disease pandemic that spread over 200 countries in the past year. Meanwhile, COVID-19 has become a serious health threat for the entire world that causes respiratory problems, heart infections, and even death.This virus first reported in a human being in December 2019 in Wuhan,China rapidly crossed the continent borders due to intensive travelling among countries. COVID-19 has had an adverse impact on the world economy too. Research studies found out that the COVID-19 virus badly affects the lungs and promptly mutates before the patient receives any diagnostic led medication . The situation becomes more severe when the symptoms match the normal flu, as in the South East and Central Asia cases. Experts found out that the incubation period of COVID-19 virus is approximately 1 week . This observation is crucial because the infected patient acts as a virus carrier during this period and unintentionally transmits it. Due to its rapid contagious nature, its spread is much faster than its detection. Machine learning methods are very popular in healthcare applications. There are various methods used to detect the presence of a COVID-19 virus in patients, such as RT PCR test X-ray imaging , computed tomography (CT) scan , rapid antigen , serological test .

While RT PCR is by far the most effective way of COVID-19 detection. This method is very time consuming (taking hours to even days) and requires special kits that may not be available in remote regions of a country due to geological, social and economic barriers. On the contrary, the rapid antigen test looks for the presence of antigens of the virus from a nasal swab but suffers from a higher rate of false negatives. The serological test looks for the antibodies produced by the immune system against the virus from the blood sample of the patient. However, it only checks the IgM and IgG antibodies during or after recovery and does not help in early virus detection. CT scan and X-ray scans, both use invisible ranges of electro-magnetic spectrum to detect any kind of anomaly, used for early detects and have high clini- cal relevance.

In our research, we found out that the chest X-ray tests are economi- cally affordable and the results are relatively easy to use. Chest X-ray tests are easily available, have portable versions, and a low risk of radiation. On the other hand, CT scans have high risk of radiation, are expensive, need clinical expertise to handle and are non-portable. This makes the use of X-ray scans more convenient than CT scans.

Recently, artificial intelligence (AI) models have provided successful results in analyzing data in a medical context. In fact, new machines have already been manufactured with AI models that perform similarly to experts in specific evaluation tasks . Moreover, a relevant application is the adoption of AI systems to extract information from medical imaging with the ultimate purpose of creating tools to reduce diagnostic errors, enhance efficiency, and reduce costs. These systems are typically embedded in image-based decision-support systems to provide aid for imaging professionals . Medical imaging enables specific capabilities, including risk assessment, detection, diagnosis, prognosis, therapy response, and multi-omics disease discovery . Their inherent automatic processing requires fewer laboratory infrastructure and supplies, as well as less health personnel. Hence, medical image analysis could also be beneficial in the diagnosis of COVID-19 .

In this context, AI-based models that use radiological medical images as input are an alternative to automate the detection of COVID-19 in patients for clinical analysis and diagnostic support. Exams based on radiological images are consolidated and can ease the burdens of highly specialized equipment for data generation (e.g., CXR image). Moreover, because CXR is a standard exam, it is possible to develop and test models based on previously available data. This can avoid the distress of disturbing patients to produce new information.

Even though previous works have developed AI-based models to identify COVID-19 in patients with CT and/or CXR images, most of them have limitations such as:

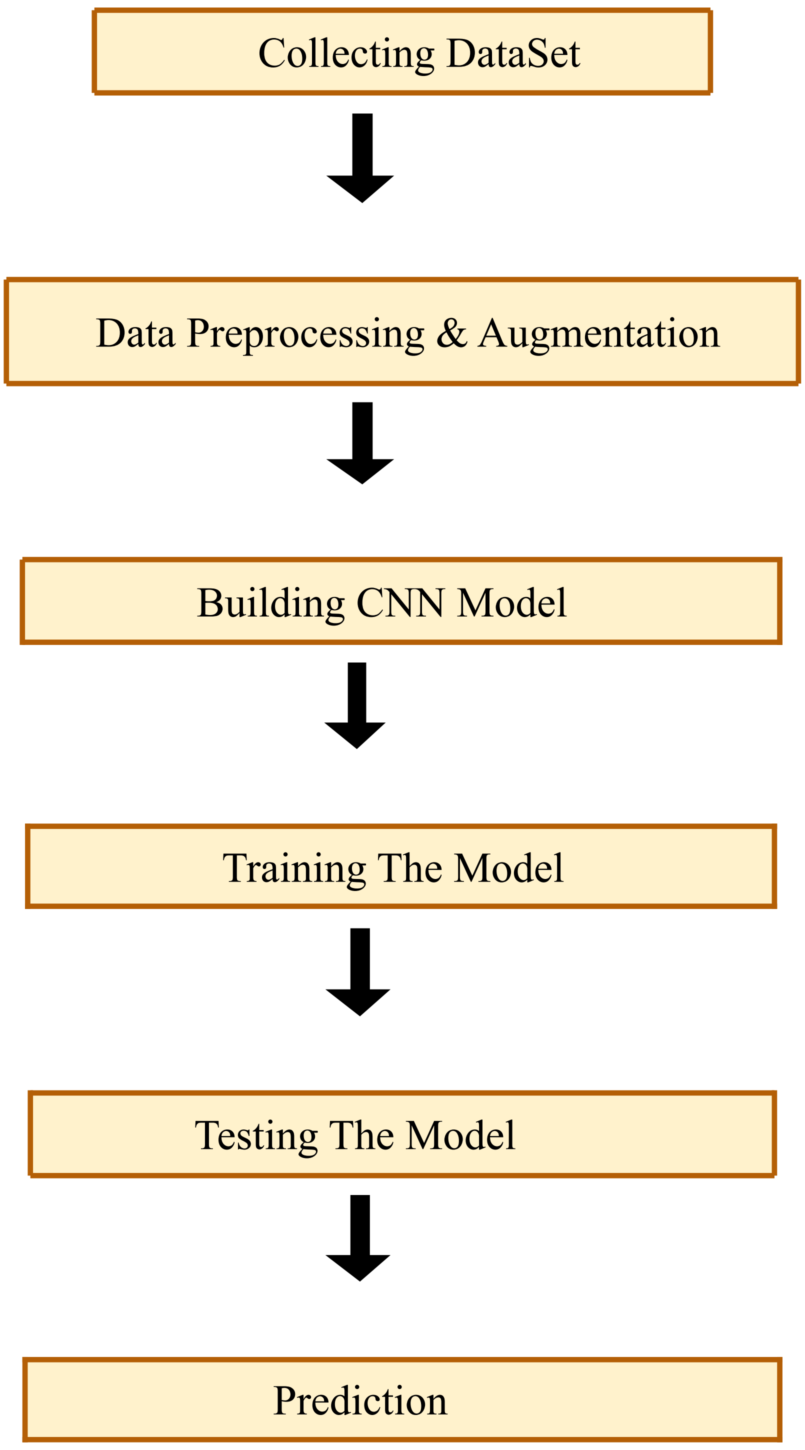
(i) use of private data preventing reproducibility

(ii) creation based on a few databases, which may not be diverse enough to truly detect features from COVID-19 and other illnesses

(iii) not being flexible enough to completely exploit images of superior quality, e.g., of higher dimensions if they become available.

Therefore, we offer here a broader discussion about the importance of adequately using various open datasets to feed the AI model and the application of data augmentation (DA) for classes with a small amount of data (images related to COVID-19, in this work). Thus, we expect to avoid biasing the model towards detecting datasets due to the characteristic features of each data source as well as fully exploiting available images from COVID-19 cases, which are still scarce.Specifically, in this project, we developed convolutional neural network (CNN)-based models to distinguish healthy and infected patients.

2.FLOW CHART


Collecting DataSet
Data Preprocessing & Augmentation
Building CNN Model
Training The Model
Testing The Model
Prediction

3.LITERATURE OF SURVEY

3.1 Introduction to Machine Learning

Machine Learning is one of the most popular approaches in Artificial Intelligence. Artificial intelligence is a technique that enables a machine to mimic human behavior. Machine learning is a technique to achieve AI through algorithms trained with data. Over the past decade, Machine Learning has become one of the integral parts of our life. It is implemented in a task as simple as recognizing human faces or as complex as self-driving cars. With the increasing amounts of data becoming available there is a good reason to believe that Machine Learning will become even more prevalent as a necessary element for technological progress.

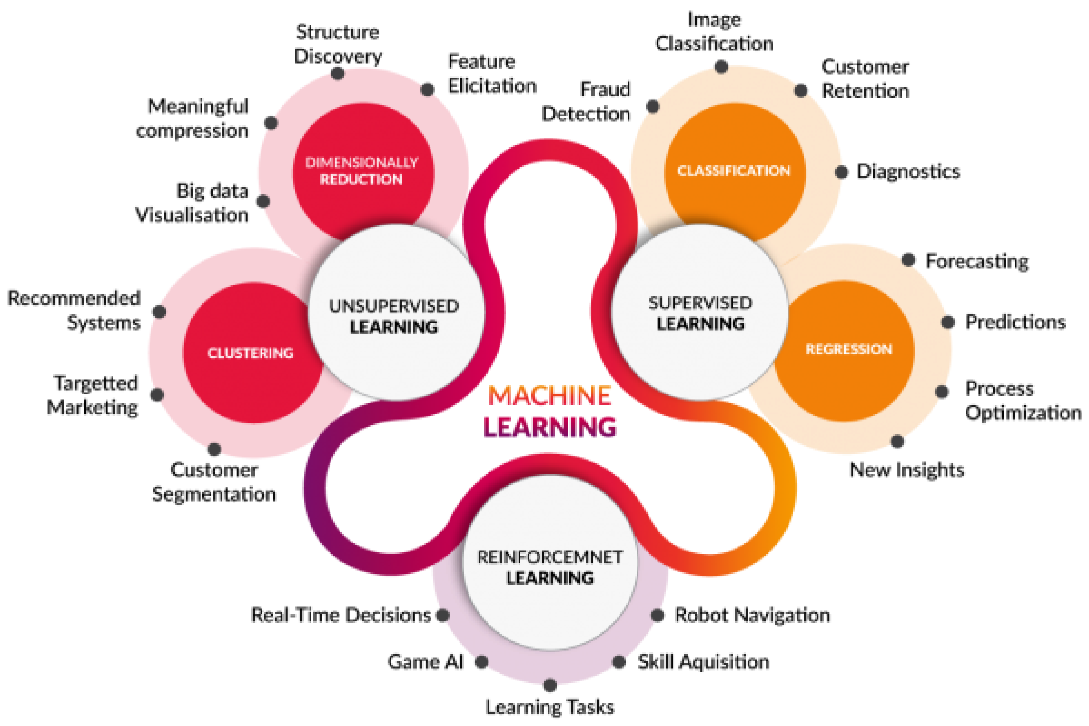


Fig. 1 Classification of Machine Learning

3.2 Types of Machine Learning

As with any method, there are different ways to train machine learning algorithms, each with their own advantages and disadvantages. To understand the pros and cons of each type of machine learning, we must first look at what kind of data they ingest. In ML, there are two kinds of data — labeled data and unlabeled data.

Labeled data has both the input and output parameters in a completely machine-readable pattern, but requires a lot of human labor to label the data, to begin with. Unlabeled data only has one or none of the parameters in a machine-readable form. This negates the need for human labor but requires more complex solutions.

There are also some types of machine learning algorithms that are used in very specific use-cases, but three main methods are used today.

Supervised Learning

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

The algorithm then finds relationships between the parameters given, by establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

Unsupervised Learning

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

Reinforcement Learning

Reinforcement Learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. With every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

3.3 Dataset

The collected Chest X-ray images dataset contains 2295 X-ray images divided into 1449 images for training and 484 for testing.

4.METHODOLOGY

4.1 The proposed CNN model

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a max pooling layer. The final layer is fully connected MLP. ReLu activation function is applied to the output of every convolutional layer and fully connected layer. The first convolutional layer filters the input image with 32 kernels of size 3x3. After max pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is given to the softmax function which produces a probability distribution of the four output classes. The model is trained using adaptive moment estimation (Adam) with batch size of 100 for 1000 epochs.

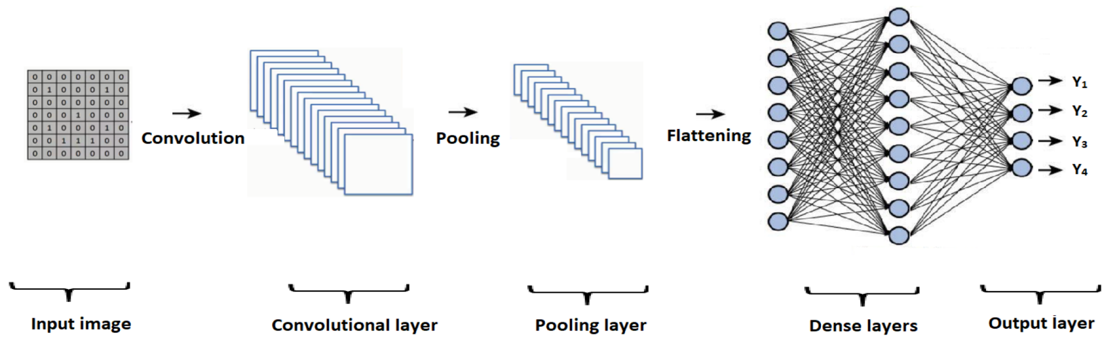


Fig2 :CNN algorithm steps

There are four CNN algorithm steps,

Convolution: The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map.

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

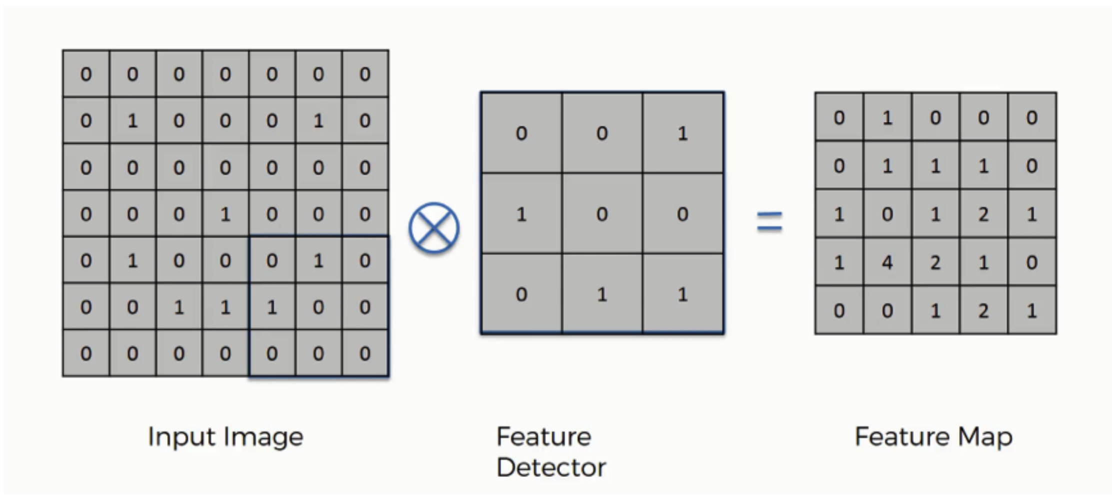


Fig 3 Convolution in CNN

Max pooling: Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned**.**

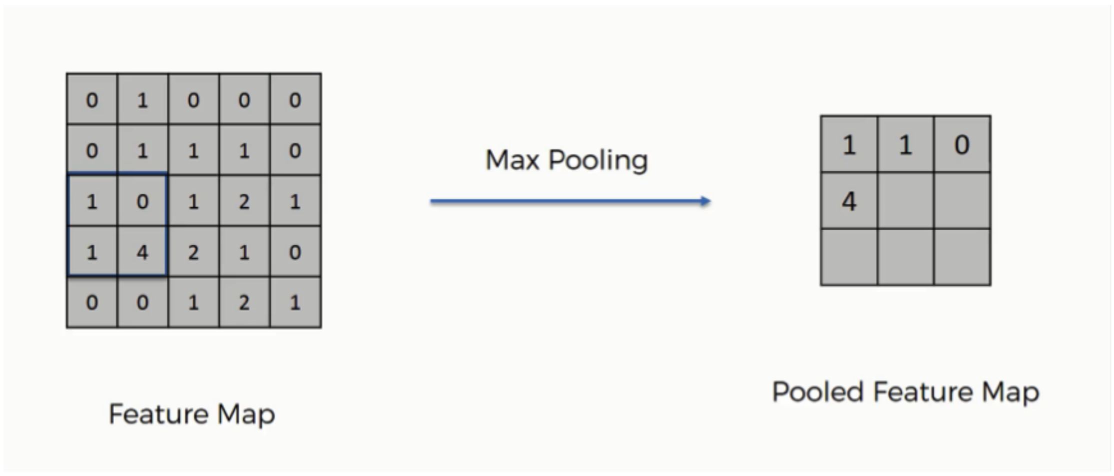


Fig 4: Max Pooling in CNN

Dropout: Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on new data.To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

Flattening: Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector.

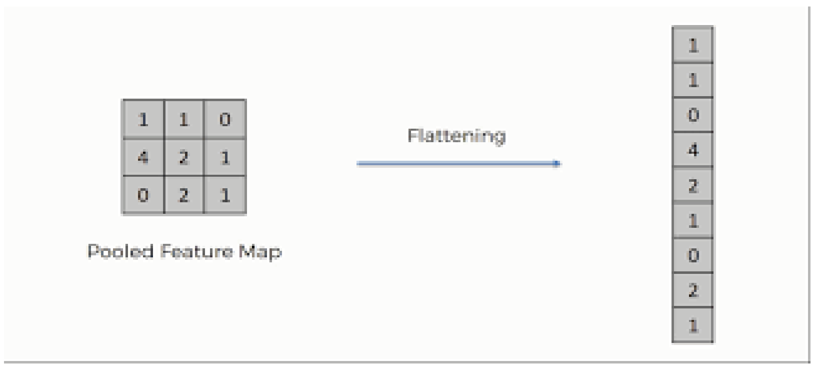


Fig 5: Flattening in CNN

Full Connection: At the end of a CNN, the output of the last Pooling Layer acts as an input to the so-called Fully Connected Layer. There can be one or more of these layers ("fully connected" means that every node in the first layer is connected to every node in the second layer).

As you see from the image below, we have three layers in the full connection step:

- Input layer
- Fully-connected layer
- Output layer

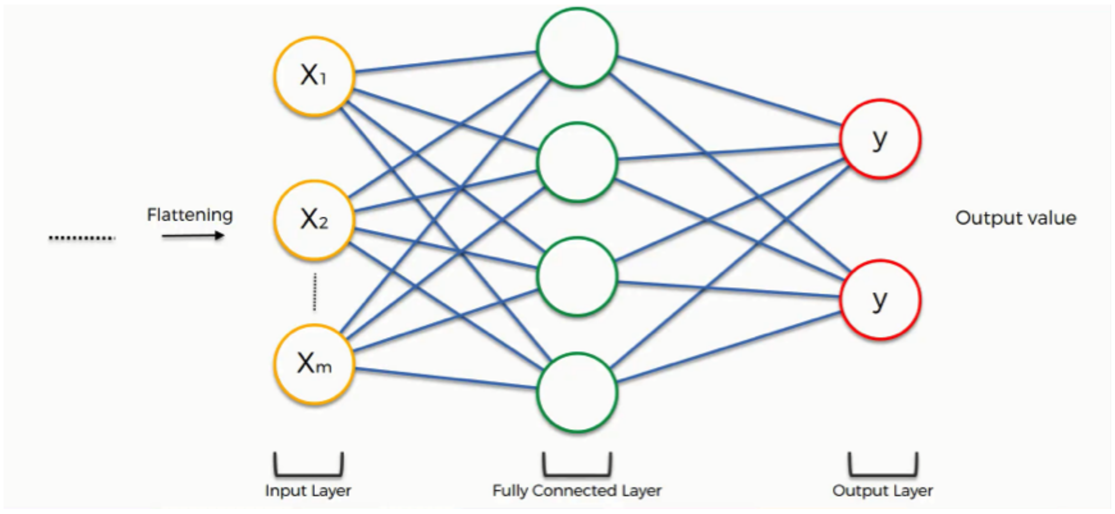


Fig 6: Full Connection in CNN

4.2 Organization of Project

The technique which is developed is taking input as a chest x-ray image and tries to predict if the uploaded image is covid positive or negative using convolutional neural network.

We have three modules in our project.

- Model Training
- Model Testing
- Prediction

5.WORKING

Initialization

- Add first layer (Convolution 2D): We use 64 output filters in the convolution 3*3 filter matrix that wi 11 multiply to input RGB size image 64*64 and use activation = relu.
- Apply (MaxPooling2D), Processing, Hidden Layer 1 (2*2 matrix rotates, tilts) to all the images. Step 1 and 2 are repeated twice.
- Adding Flattening: converts the matrix in a single array.
- Adding full connection (128 final layer of outputs, activation= relu & Dense layer, activation= sigmoid).

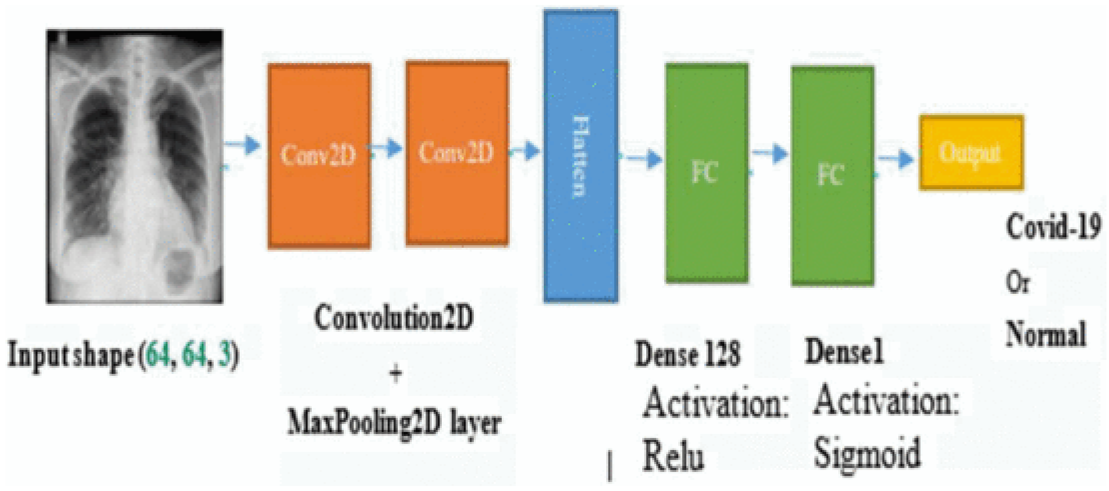
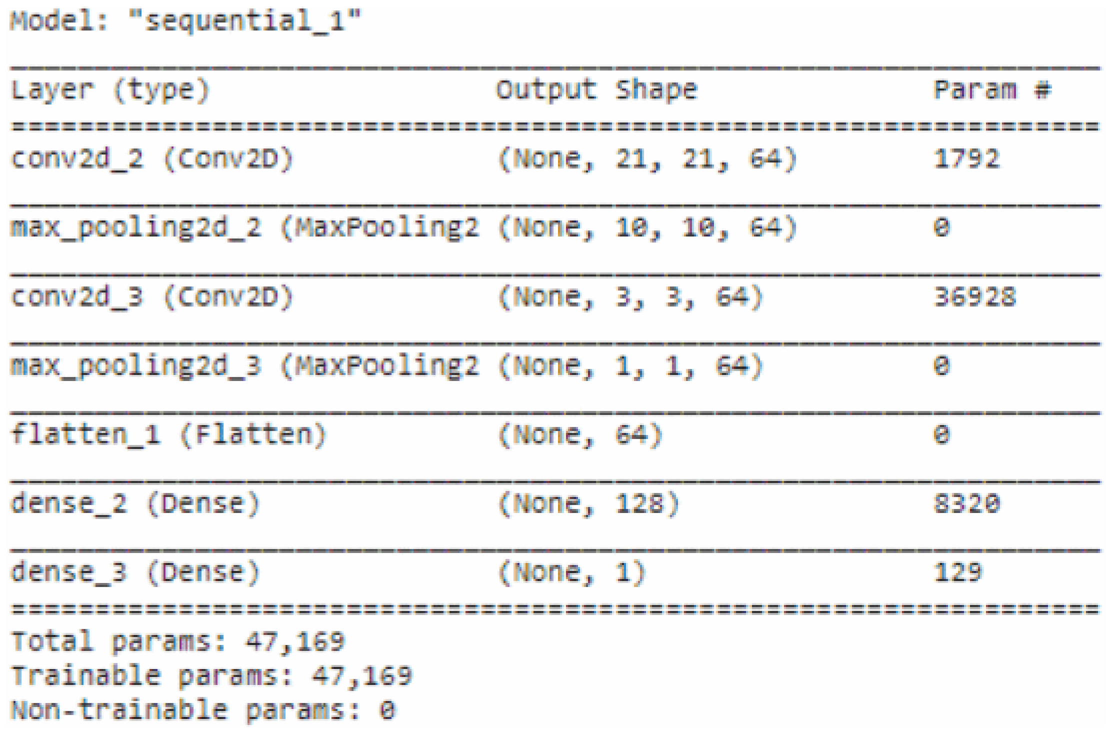


Fig 7.The layer of CNN architecture

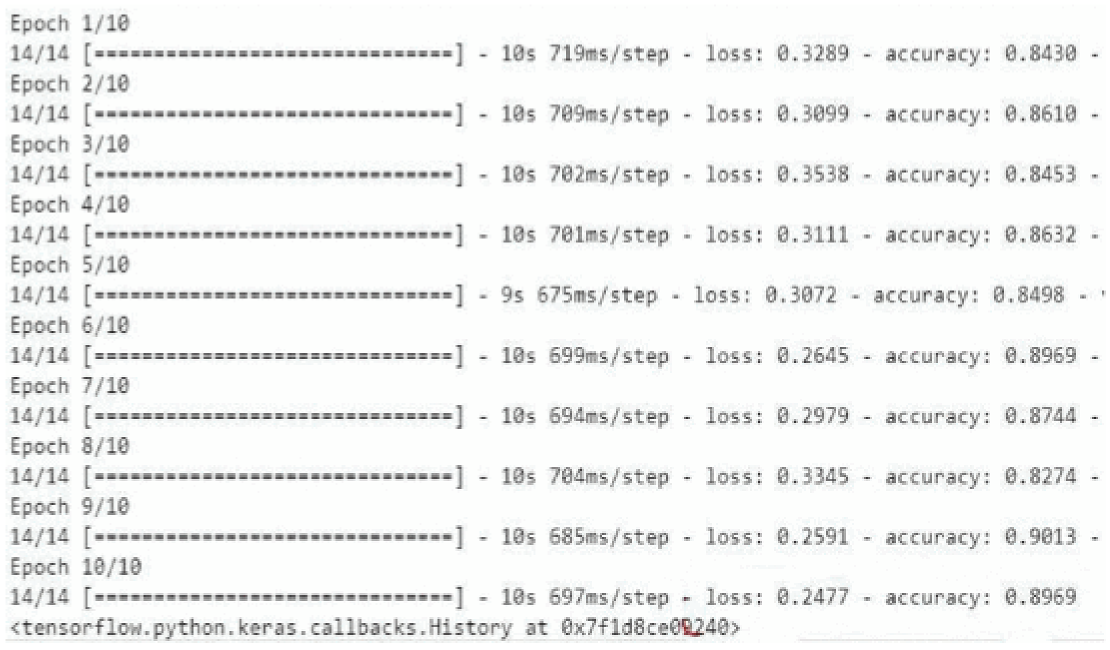


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 21, 21, 64)        1792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 47,169
Trainable params: 47,169
Non-trainable params: 0
```

Fig.8.CNN model summary.

Fit the Cnn Model

In the field of deep learning, a popular Adam algorithm is used because it delivers good results quickly, and it is an optimization algorithm to update iterative network weights based on training data instead of the classic random gradient descent procedure .



```
Epoch 1/10
14/14 [==============================] - 10s 719ms/step - loss: 0.3289 - accuracy: 0.8430 -
Epoch 2/10
14/14 [==============================] - 10s 709ms/step - loss: 0.3099 - accuracy: 0.8610 -
Epoch 3/10
14/14 [==============================] - 10s 702ms/step - loss: 0.3538 - accuracy: 0.8453 -
Epoch 4/10
14/14 [==============================] - 10s 701ms/step - loss: 0.3111 - accuracy: 0.8632 -
Epoch 5/10
14/14 [==============================] - 9s 675ms/step - loss: 0.3072 - accuracy: 0.8498 -
Epoch 6/10
14/14 [==============================] - 10s 699ms/step - loss: 0.2645 - accuracy: 0.8969 -
Epoch 7/10
14/14 [==============================] - 10s 694ms/step - loss: 0.2979 - accuracy: 0.8744 -
Epoch 8/10
14/14 [==============================] - 10s 704ms/step - loss: 0.3345 - accuracy: 0.8274 -
Epoch 9/10
14/14 [==============================] - 10s 685ms/step - loss: 0.2591 - accuracy: 0.9013 -
Epoch 10/10
14/14 [==============================] - 10s 697ms/step - loss: 0.2477 - accuracy: 0.8969
<tensorflow.python.keras.callbacks.History at 0x7f1d8ce09240>
```

Fig .9 Result of model training in 10 epoch

The optimizer which is called Adam, is an efficient variant of gradient descent which generally does not require hand-tuning of the learning rate. Throughout training, the optimizer uses the gradients of the loss to try reducing the error ("optimize") of the model output by adjusting the parameters. According to Kingma et al. , "the method is computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

Apply fitting to the training set (steps_per_epoch:100, no. epoch: 10, Validation data: test-set, nb.val.samples (60), callbacks= [early_stop]). shows the results of model training inl0 Epochs.

Evaluating and Testing the Model

For evaluation the CNN model, loss (binary_crossentropy) and accuracy metrics are used. The Loss Function which is known as a prediction error of Neural Net, is one of the most important components of Neural Networks, and it is also considered as the method to calculate the loss.

Simply, we can describe the way that a Neural Net is trained as follows: The Loss is used to calculate the gradients and then, gradients are used to update the weights of the Neural Net .

While binary Crossentropy (BCE) loss is used for the binary classification tasks. We use BCE loss function just when we need one output node and this for classifying the data into two classes, hence the output value should be passed through a sigmoid activation function and the range of output should be (0 - 1). The Cross-Entropy (CE) loss can be expressed as explained by.

In order to verify our Model, some samples of cells are given to detect covid-19. The pictures were loaded and converted into an array, then the model can predict if the image is covid-19 or normal. The below fig shows a model of testing for new X-Ray images.

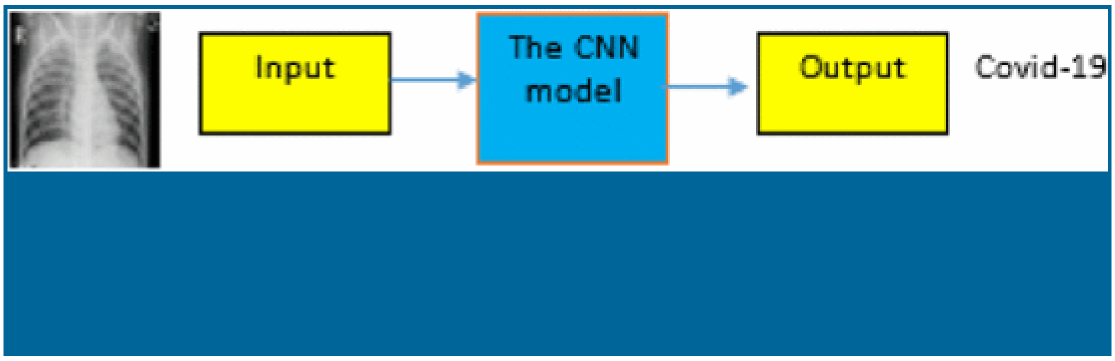


Fig .10.New prediction Model

6. SOFTWARE REQUIREMENTS

Programming Language	:	Python 3.6
Dataset	:	Chest X-Ray images Dataset
Packages	:	Tensorflow, Numpy, Matplotlib
Tool	:	Google Colaboratory

7. TECHNOLOGIES DESCRIPTION

7.1 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

7.2 Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these 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

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

7.3 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools,libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

7.4 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

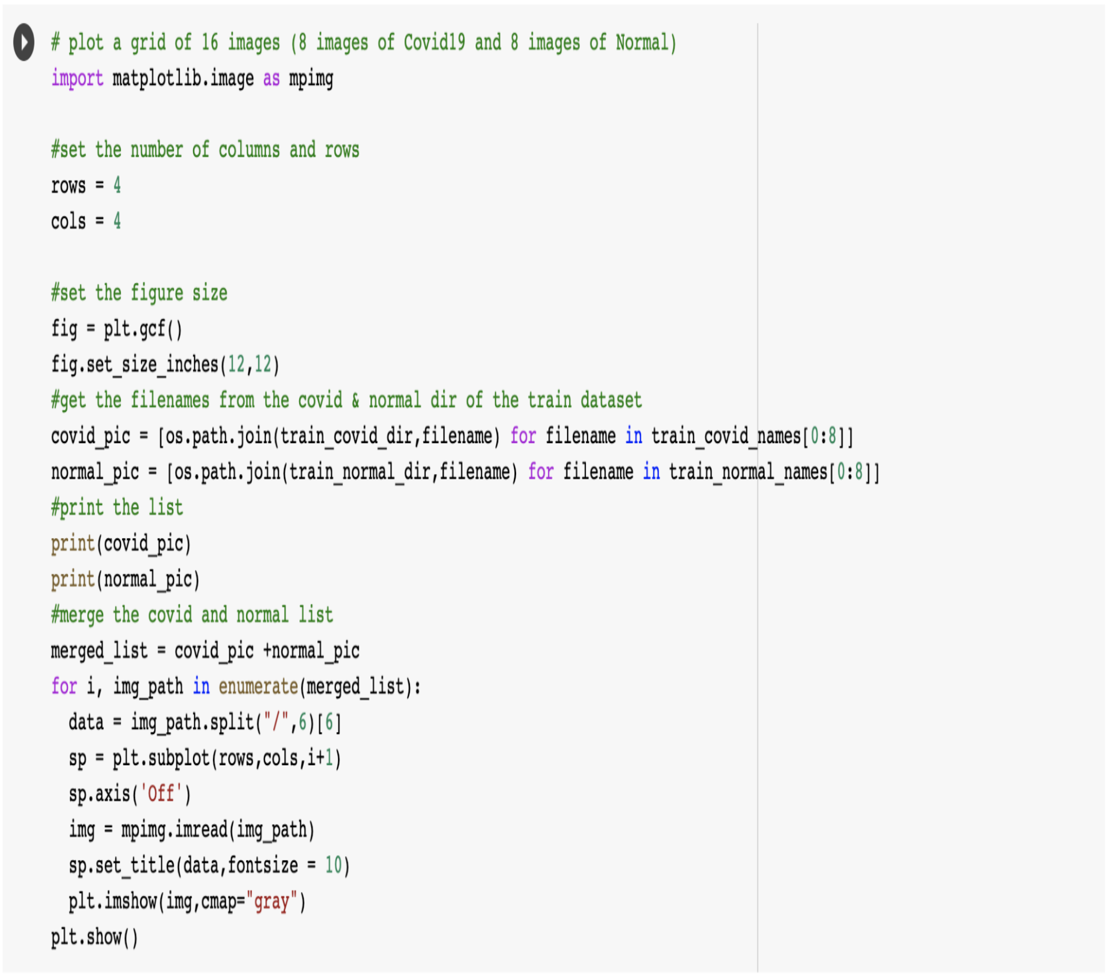
For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

7.5 Google Colaboratory

Colab notebooks are Jupyter notebooks that are hosted by Colab. Colab notebooks allow us to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When we create our own Colab notebooks, they are stored in your Google Drive account. We can easily share Colab notebooks with co-workers or friends, allowing them to comment on our notebooks or even edit them.

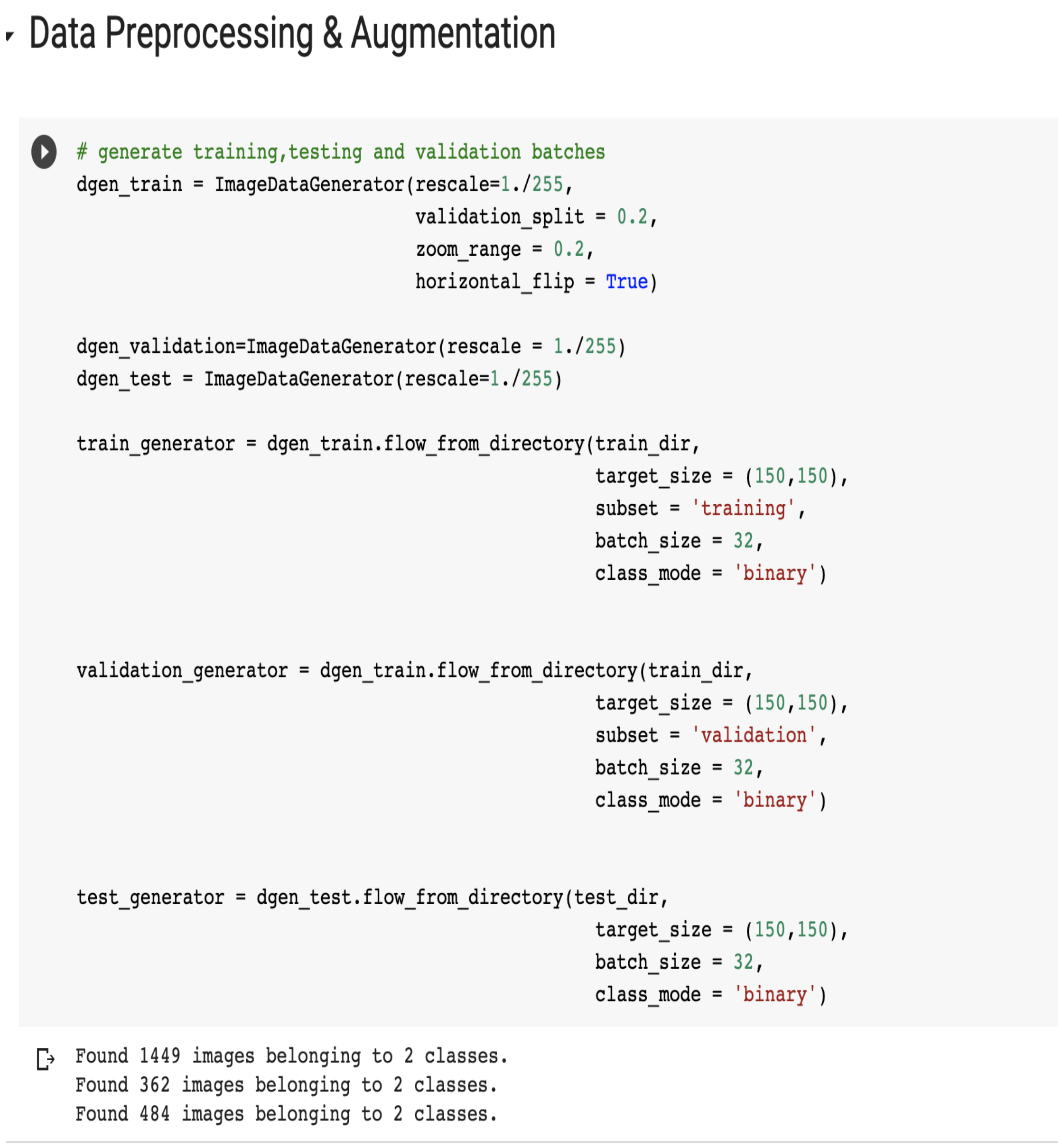
8.CODE

Data Visualization



```python
# plot a grid of 16 images (8 images of Covid19 and 8 images of Normal)
import matplotlib.image as mpimg

#set the number of columns and rows
rows = 4
cols = 4

#set the figure size
fig = plt.gcf()
fig.set_size_inches(12,12)
#get the filenames from the covid & normal dir of the train dataset
covid_pic = [os.path.join(train_covid_dir,filename) for filename in train_covid_names[0:8]]
normal_pic = [os.path.join(train_normal_dir,filename) for filename in train_normal_names[0:8]]
#print the list
print(covid_pic)
print(normal_pic)
#merge the covid and normal list
merged_list = covid_pic +normal_pic
for i, img_path in enumerate(merged_list):
  data = img_path.split("/",6)[6]
  sp = plt.subplot(rows,cols,i+1)
  sp.axis('Off')
  img = mpimg.imread(img_path)
  sp.set_title(data,fontsize = 10)
  plt.imshow(img,cmap="gray")
plt.show()
```

Fig 11.Data Visualization



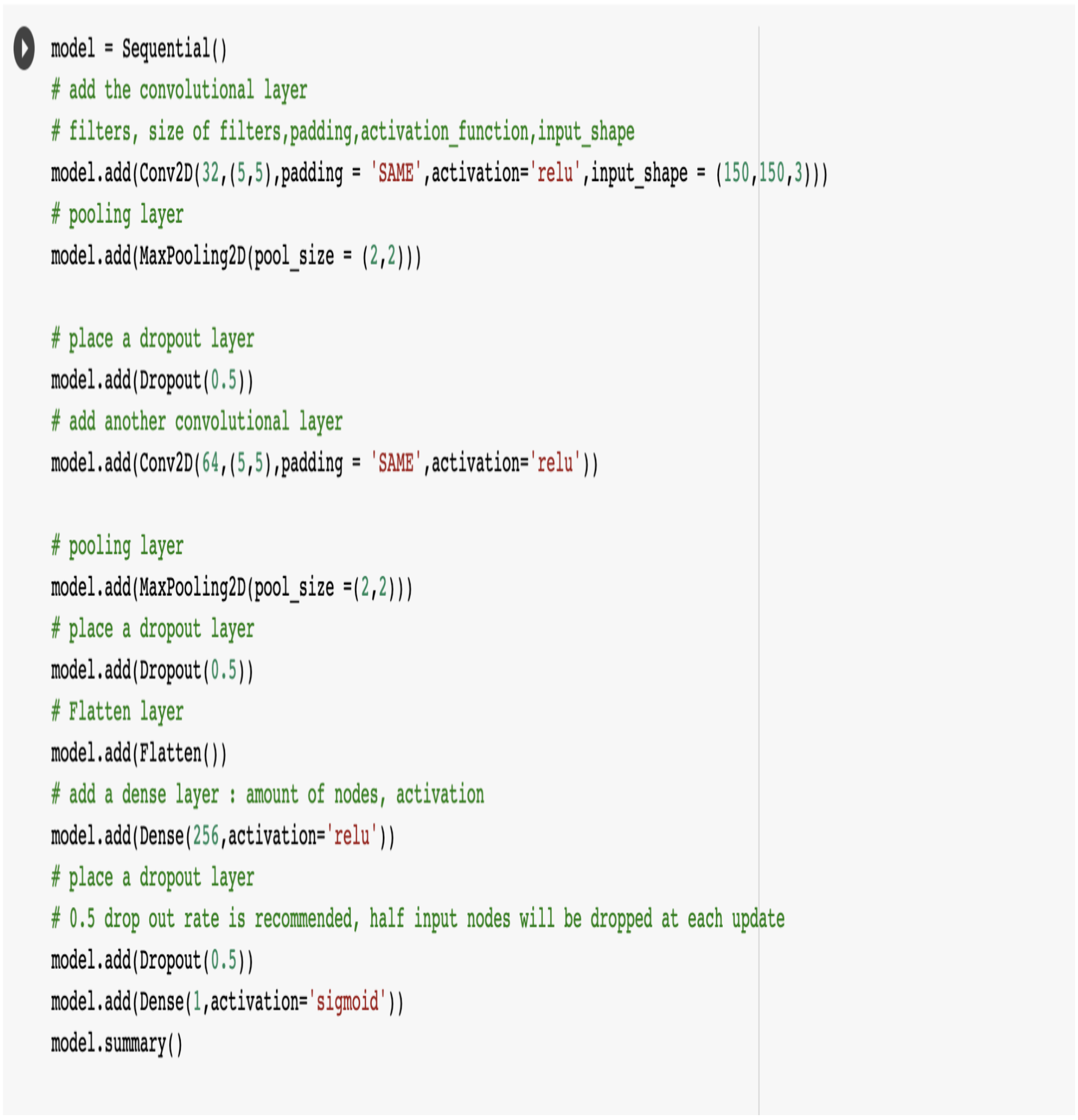
Data Preprocessing & Augmentation

```
# generate training,testing and validation batches
dgen_train = ImageDataGenerator(rescale=1./255,
                                validation_split = 0.2,
                                zoom_range = 0.2,
                                horizontal_flip = True)

dgen_validation=ImageDataGenerator(rescale = 1./255)
dgen_test = ImageDataGenerator(rescale=1./255)

train_generator = dgen_train.flow_from_directory(train_dir,
                                                 target_size = (150,150),
                                                 subset = 'training',
                                                 batch_size = 32,
                                                 class_mode = 'binary')


validation_generator = dgen_train.flow_from_directory(train_dir,
                                                 target_size = (150,150),
                                                 subset = 'validation',
                                                 batch_size = 32,
                                                 class_mode = 'binary')


test_generator = dgen_test.flow_from_directory(test_dir,
                                                 target_size = (150,150),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 1449 images belonging to 2 classes.
Found 362 images belonging to 2 classes.
Found 484 images belonging to 2 classes.
```

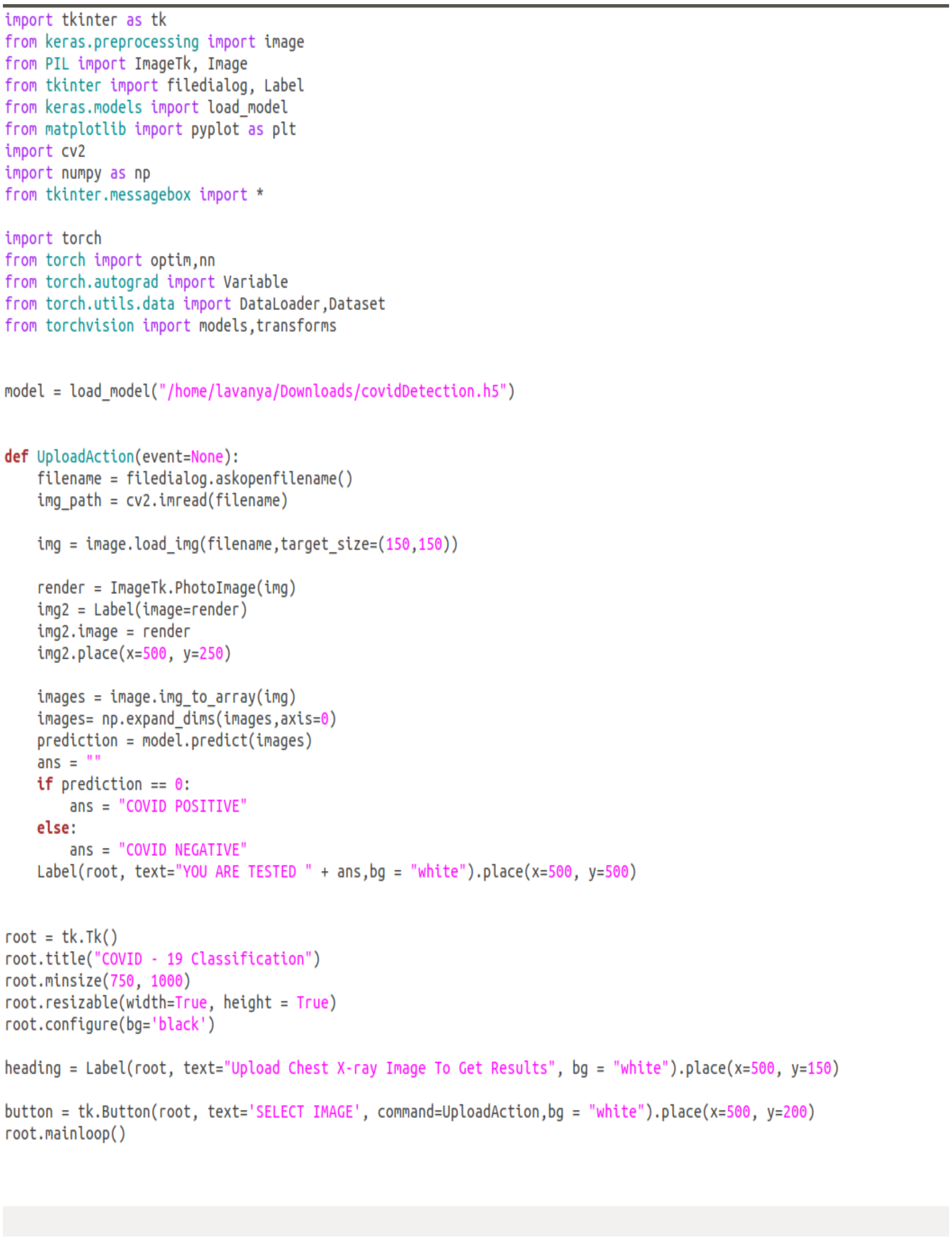
Fig 12.Data Preprocessing

Build Convolutional Neural Network Model



```
model = Sequential()
# add the convolutional layer
# filters, size of filters,padding,activation_function,input_shape
model.add(Conv2D(32,(5,5),padding = 'SAME',activation='relu',input_shape = (150,150,3)))
# pooling layer
model.add(MaxPooling2D(pool_size = (2,2)))

# place a dropout layer
model.add(Dropout(0.5))
# add another convolutional layer
model.add(Conv2D(64,(5,5),padding = 'SAME',activation='relu'))

# pooling layer
model.add(MaxPooling2D(pool_size =(2,2)))
# place a dropout layer
model.add(Dropout(0.5))
# Flatten layer
model.add(Flatten())
# add a dense layer : amount of nodes, activation
model.add(Dense(256,activation='relu'))
# place a dropout layer
# 0.5 drop out rate is recommended, half input nodes will be dropped at each update
model.add(Dropout(0.5))
model.add(Dense(1,activation='sigmoid'))
model.summary()
```

Fig 13. Build Model code



```python
import tkinter as tk
from keras.preprocessing import image
from PIL import ImageTk, Image
from tkinter import filedialog, Label
from keras.models import load_model
from matplotlib import pyplot as plt
import cv2
import numpy as np
from tkinter.messagebox import *

import torch
from torch import optim,nn
from torch.autograd import Variable
from torch.utils.data import DataLoader,Dataset
from torchvision import models,transforms


model = load_model("/home/lavanya/Downloads/covidDetection.h5")


def UploadAction(event=None):
    filename = filedialog.askopenfilename()
    img_path = cv2.imread(filename)

    img = image.load_img(filename,target_size=(150,150))

    render = ImageTk.PhotoImage(img)
    img2 = Label(image=render)
    img2.image = render
    img2.place(x=500, y=250)

    images = image.img_to_array(img)
    images= np.expand_dims(images,axis=0)
    prediction = model.predict(images)
    ans = ""
    if prediction == 0:
        ans = "COVID POSITIVE"
    else:
        ans = "COVID NEGATIVE"
    Label(root, text="YOU ARE TESTED " + ans,bg = "white").place(x=500, y=500)


root = tk.Tk()
root.title("COVID - 19 Classification")
root.minsize(750, 1000)
root.resizable(width=True, height = True)
root.configure(bg='black')

heading = Label(root, text="Upload Chest X-ray Image To Get Results", bg = "white").place(x=500, y=150)

button = tk.Button(root, text='SELECT IMAGE', command=UploadAction,bg = "white").place(x=500, y=200)
root.mainloop()
```

Fig 14. Prediction code

9.Result

In the first experiment, 2295 Chest X-ray images dataset was used for training and testing, 1449 images for training and 484 for testing . The model was trained with a set of 1449 x-rays of people with Covid-19 and normal, and a test set containing 484chest radiographs divided between people with Covid-19 and people without infection. The complete data set must pass multiple times to the same Neural Network to improve the learning process.

Through experience, it has been shown that with an increase in the training sample, certain models can achieve a more accurate identification of COVID-19 in human samples, allowing the identification of the patient.

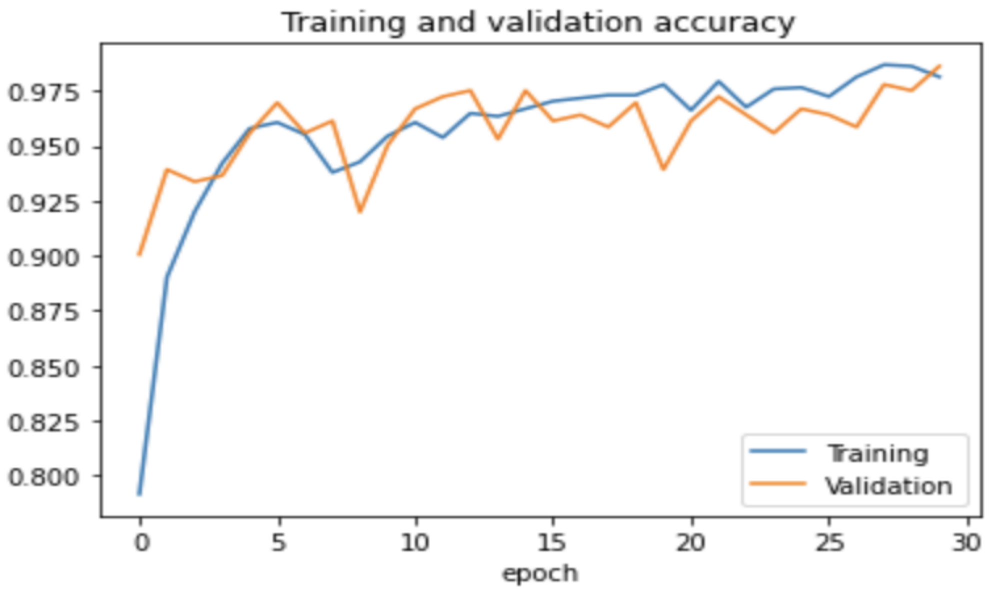


Fig 15. Training and Validation Accuracy Graph

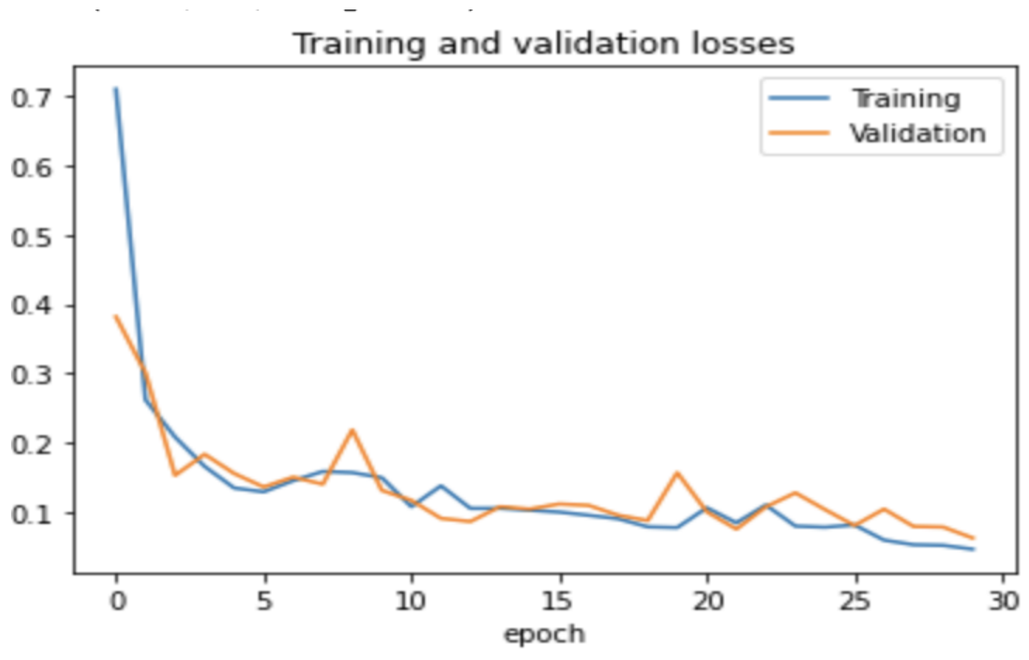
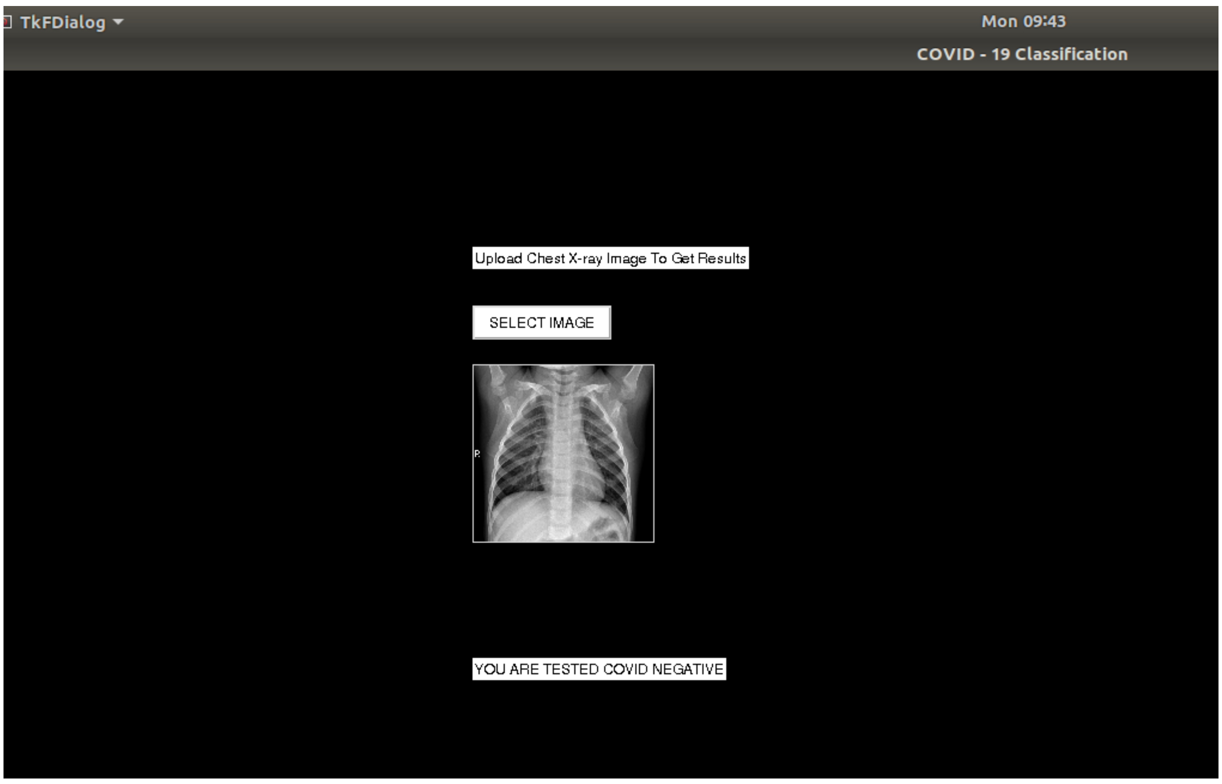


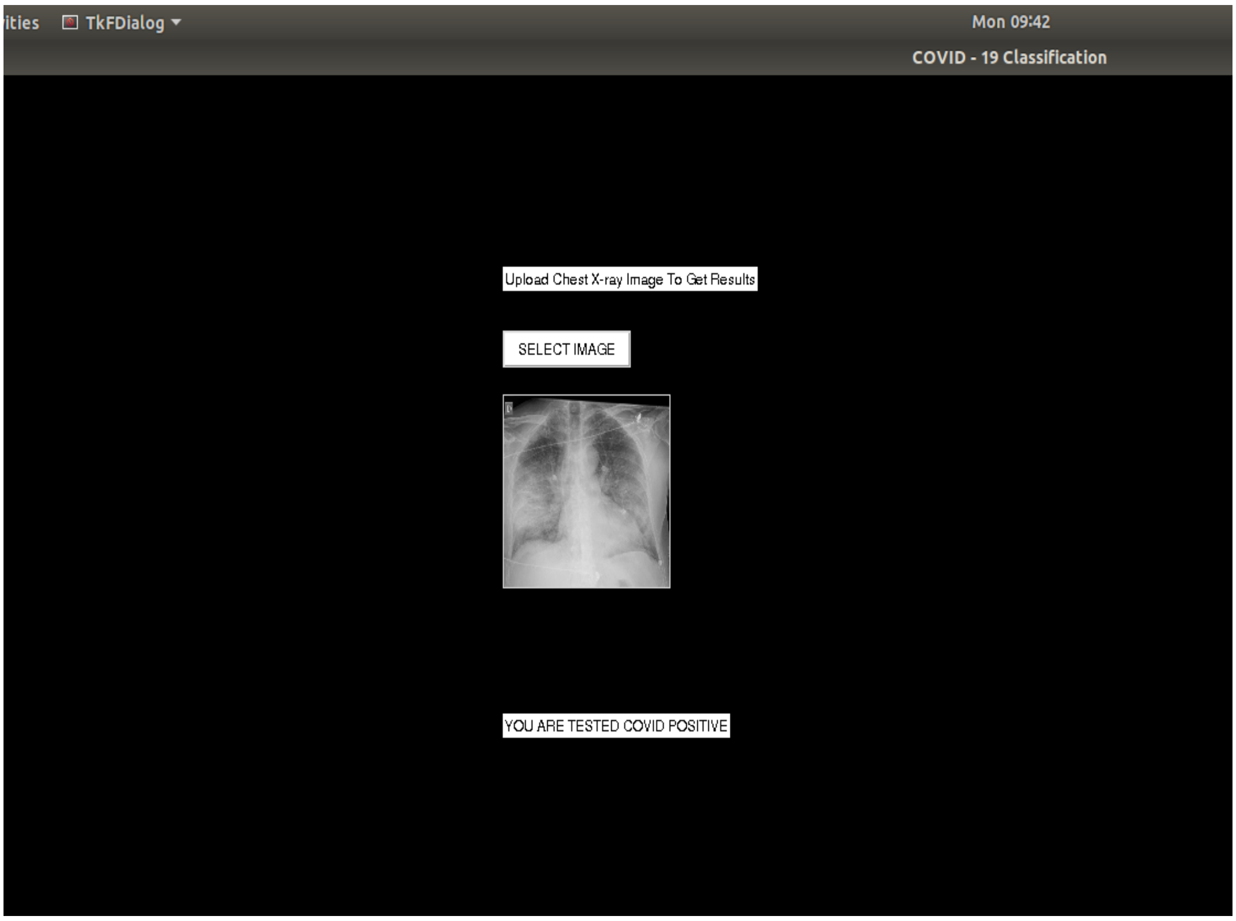
Fig 16. Training and Validation Losses Graph



Upload Chest X-ray Image To Get Results

YOU ARE TESTED COVID NEGATIVE

Fig 17. COVID NEGATIVE chest x-ray image prediction


ities
TkFDialog
Mon 09:42
COVID - 19 Classification
Upload Chest X-ray Image To Get Results
SELECT IMAGE

YOU ARE TESTED COVID POSITIVE

Fig 18. COVID POSITIVE chest x-ray image prediction

10.ADVANTAGES

- It decreases the diagnostic time
- It reduces the financial costs.
- This provides valuable assistance to doctors who are diagnosing Covid-19
- Not painful whereas the existing methods are a bit painful.

11.DISADVANTAGES

The proposed model is intended as a support tool. Thus, a detailed clinical study involving pieces of evidence of different sources is necessary to provide a final medical diagnosis.

Additionally, despite the use of DA to artificially generate more than 10 000 images, the number of original images related to COVID-19 cases is rather small (i.e., 573). More images would be desirable and would provide more visual information about COVID-19 features.

Note that evaluating images is not as intensive as training. The assessment of one single image takes less than a second, even with a CPU. However, training the proposed models is a computer-intensive task, which took about 5 to 6 hours to train using a GPU for high image dimensions. Therefore, training new models based on the proposed methodology may not be feasible in low-end computers.

12.FUTURE SCOPE

- At present our model just predicts whether the chest x-ray image of a patient uploaded is covid positive or covid negative.

- This model can further be improved to even detect the areas where the covid is actually affected and to what extent that particular area in lungs is affected.

- This extension to the current version of our project can actually tell whether the patient is infected by covid and to what extent the patient is infected by covid.

- This helps the doctors as well as patients to take measures and treatment as required so that the patient can be cured.

13.CONCLUSION

In conclusion, the results of this unique research show a potential role of a very accurate Artificial Intelligence algorithm to quickly identify patients, which could be useful and effective in combating the current outbreak of Covid-19. We are almost certain that it is possible for the proposed CNN model, which shows the equivalent of the highest score for the accuracy of a specialized chest radiologist, represents a very effective examination tool for the rapid diagnosis of many infectious diseases such as the COVID-19 epidemic that do not require the introduction of a radiologist or physical examinations.

In future studies, we recommend addressing other topics such as outbreak escalates , as well as trying to explore different approaches to convolutional neural networks, including deep learning models and improved interpretation of CNN models.

14.REFERENCES

- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://www.webmd.com/lung/what-does-covid-do-to-your-lungs#1
- https://www.who.int/health-topics/coronavirus#tab=tab_1
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7187882/
- https://medium.com/analytics-vidhya/a-complete-guide-for-visualising-and-understanding-convolutional-networks-dc26f71c979f
- https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a