Major Project Stage - II

Detection of Covid-19 using ResNet50 and VGG16 in Convolution Neural Network

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

18WH1A0570	Ms. T.VARSHITHA
18WH1A0581	Ms. M.KRISHNAVENI
18WH1A0583	Ms. L.V.NAVYA SREE

under the esteemed guidance of

Ms.M.Shanmuga Sundari

Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2021-2022

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Work report on **"Detection of Covid-19 using ResNet50 and VGG16 in Convolution Neural Network "** is a bonafide work carried out by Ms. T.VARSHITHA (18WH1A0570) ; Ms. M.KRISHNAVENI (18WH1A0581) ; Ms. L.V.NAVYA SREE (18WH1A0583) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.
The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr.K.Srinivasa Reddy	**Ms.M.Shanmuga Sundari**
Professor and HoD,	**Assistant Professor,**
Department of CSE	**Department of CSE**

External Examiner

DECLARATION

We hereby declare that the work presented in this project entitled **"Detection of Covid-19 using ResNet50 and VGG16 in Convolution Neural Network "** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering for Women', Hyderabad is an authentic record of our original work carried out under the guidance of Ms.M.Shanmuga Sundari Assistant Professor, Department of CSE.

Sign:

Ms. T.VARSHITHA
(18WH1A0570)

Sign:

Ms. M.KRISHNAVENI
(18WH1A0581)

Sign:

Ms. L.V.NAVYA SREE
(18WH1A0583)

ACKNOWLEDGEMENTS

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. K.Srinivasa Reddy, Professor and HOD, Department of CSE, BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. M.Shanmuga Sundari, Assistant Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support through out the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of CSE Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. T.VARSHITHA (18WH1A0570)

Ms. M.KRISHNAVENI (18WH1A0581)

Ms. L.V.NAVYA SREE (18WH1A0583)

ABSTRACT

A novel coronavirus (CoV) generally known as ‘SARS-CoV2’ or ‘2019 novel coronavirus’ or ‘COVID-19’ by the WHO which is the outbreak of pneumonia that started at the beginning of 2019 in December in China. COVID-19 is a contagious virus. As COVID-19 is a contiguous disease, early detection of the virus is very important but it can be incurable if the virus is detected later. The identification of COVID-19 disease is done by collecting samples from the throat and nose, which is susceptible to errors which are made by humans. This research proposed a system for identification of the virus utilizing X-Ray images. Dataset used consists of both Covid and Normal X-ray images. Among Convolutional Neural Network(CNN) models, the proposed models are ResNet50 and VGG16. RESNET50 consists of 48 convolutional, 1 MaxPool and Average Pool layers and VGG16 is another convolutional neural network that consists of deep 16 layers. By using these two models detection of COVID-19 is done. This research can be practically helpful to the physicians with the usage of datasets for the successful diagnosis of COVID-19 disease in the medical field.

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
VGG	Visual Geometry Group
ResNet	Residual Network

LIST OF CONTENTS

1.INTRODUCTION

1.1 Objective:

COVID-19 Detection is a method that detects human chest X-Ray images into Normal or COVID-19. This process can be used in practice to help the physicians in faster detection of the virus.

1.2 Problem statement: COVID-19 has been a wide spread disease for which a lot of new medicine and technology is being developed. But one of the major issues still faced by the physicians is in the detection of the virus. Inorder to detect the virus in a fast and precise manner we are proposing a machine learning model for COVID-19 detection using ResNet50 and VGG16 models.

1.3 Proposed system:

It is not possible for a computer to detect the X-Ray image in a faster way on its own.
The proposed Machine Learning model classifies and detects human chest X-Ray images into Normal or COVID-19. The model is trained on the COVID-19 Radiography Database. The input to the model will be an image and the output will be shown as COVID-19 or normal.

The project involves two models of Convolutional Neural Networks(CNN) that is Resnet50 and VGG16. The project used the CNN model in 2 phases:

1. Training phase
2. Testing phase

In the training phase, the CNN model is set up and trained on COVID-19 Radiography Database. Then in the testing phase the model is tested on other images.

2. LITERATURE SURVEY

In [1] this paper, the authors have worked on the CNN model and came up with the detection of COVID-19 virus utilizing COVID-19 Radiography dataset from Kaggle. They used ReLU as the activation function and a Softmax classifier which builds end predictions of the disease. This CNN model is used to conclude whether a given chest X-ray image of a patient has COVID-19 or not with an accuracy of 99.20%.

The authors in [2] have attempted a system for identification of the Coronavirus infected images of human chest X-Ray using CNN model. Local binary patterns(LBP) are used for extracting features and classifiers used are "K-Nearest Neighbours(KNN), Naive Bayes(NB), Random Trees and Random Forests, Support Vector Machine(SVM)." Here, the proposed method uses a dataset which has human chest X-Rays of non infected people as well as patients suffering from pneumonia and Covid19 virus infection.

In [3], the authors have proposed the tensor flow based CNN method which is used to classify chest x-ray images. They used datasets like "Covid-chest Xray-dataset and Chest X-Ray Images (Pneumonia)". The performance of the CNN model is a helpful tool for radiologists for COVID-19 classification and detection. This approach with RELM will help the researchers for further analysis. This model shows the accuracy of 95%.

In [4] ,The authors used radiology which helps to diagnose patients carrying coronavirus symptoms . They worked on the dataset taking two different sources. The first is Covid-chestxray-dataset which consists of covid-positive X-Ray images and second source is ChestX-ray8 database which consists of X-Rays of normal and pneumonia-infected people. Here, they used concepts of CNN as the model and resulted an accuracy of 87%.

In [5] this paper, a database of X-ray, CT-Scan images with pneumonia, Covid-19 infection, and common cases, were used to detect Coronavirus infection. They have used ResNet50, Inception V3 and Inception-ResNetV2 pre-trained models to obtain high accuracy for small X-ray databases. Test results obtained from the chest X-ray and CT dataset show features obtained from DenseNet 121 and later trained by the Bagging tree classifier. This produces accurate predictions of 99.00%.

In [6], the authors have proposed a deep transfer learning-based model that improves the current accuracy. They worked on COVID-19 Radiography dataset from Kaggle. The multiple CNN models are used like "AlexNet, VGG, SqueezeNet, GoogleNet". The DenseNet121 and ResNet50 are the two networks which are fine-tuned with the deep classifiers such as "Support vector machine (SVM), random forest, K-Nearest neighbor (KNN), and CNN with softmax classifier" with data augmentation to detect three classes of COVID-19 ,Viral Pneumonia and normal radiographs. This model shows an accuracy of 97.83%.

In this paper[7], the author presents an examination of the Convolutional Neural Network(CNN) for the classification of COVID-19 and normal patients. Binary classification was performed, Sigmoid was chosen as the classifier, Binary-Cross Entropy is used as the loss function. It concentrates on the comparison of APST-Net with the pre-trained models of CNN and also incorporating various epochs with different optimizers which can be executed for the arrangement of other image processing techniques. The 2 datasets used here for this research are Chest X-Ray Images(Pneumonia) from Kaggle and another is "covid-chestxray-dataset" from GitHub. APST-Net trained with Adam optimizer attains the highest training, validation, and F1-score of 98.45%, 98.20%, and 98.18%.

In [8], the authors used datasets like "Covid-chestxray-dataset and Chest X-Ray Images (Pneumonia)". They worked on "Detection And Spread Prediction Using CNN-GMM(Gaussian Mixture Model)". The GMM-based linear regression successfully predicts the disease using the parameters like mean and covariance of the distribution. The model shows the accuracy of 96.6%.

In [9], the authors used X-Ray and SARS-CoV-2 CT scan datasets. They proposed a deep learning model using Transfer learning to detect COVID-19. The developed approach consisted of DenseNet and InceptionV3 models by at least 12 percent, which explains the feasibility and effectiveness of the proposal. Both X-ray and CT scans were considered to estimate the proposed methods.

In [10], authors included a total of 657 chest X-ray images which have been taken for the diagnosis of COVID-19 using deep learning methods like CNN, VGG16, VGG19, and InceptionV3. Here, Softmax Classifier is used and Relu is taken as the activation function. COVID-19, healthy patients, and viral pneumonia cases are classified successfully by the VGG19 model that has a 95% accuracy rate. InceptionV3 is the most doomed(unsuccessful) method for the dataset.

In [11], the main goal of this paper is to provide how various machine learning models are implemented in the real-world. Apart from this, this paper also analyzes the current trend or pattern of Covid-19 transmission in India with the help of datasets from the Ministry of Health and Family Welfare of India. Here ,the data to be studied has been obtained for 154 days. The data can be further analyzed for later references, and more results can be obtained. Through the analysis can be concluded that the Polynomial Regression Algorithm as compared to the SVM Algorithm, shows an accuracy of approximately 93% by predicting the rise in cases for the coming 60 days.

In [12], authors have taken datasets like "Covid-chestxray-dataset and Chest X-Ray Images (Pneumonia)". They used a CNN model to detect COVID-19 and it is evaluated by comparing with two other CNN models. Here, the ReLU activation function is used. This model obtains the Receiver Operating Characteristic (ROC) curve area of 0.976 and F1-score of 97.61 and this can be improved by increasing the dataset for training the model. The proposed model executes with an accuracy of 97.56% and a precision value of 95.34%.

In [13], the authors have utilized X-Ray images of COVID-19 which were extracted from online hosted data by Italian research organization, European Health Care and Chest X-Ray Images(Pneumonia) dataset from Kaggle. After removing noisy images, a dataset with images for COVID-19, PNEUMONIA, and NORMAL in each label was extracted. Hence, the end results are reported only for ResNet-34 and ResNet-50 trained using transfer learning. In the end, ResNet - 50 architecture performed better, with an accuracy of 72.38%.

In this [14] paper, the authors have utilized the concept of artificial intelligence in the deep learning field. The dataset used here is obtained from a GitHub repository owned by UCSD-AI4H and it consists of images of chest CT-Scan from infected and non-infected patients. They proposed FJCovNet which is a new deep learning model i.e based on DenseNet121. FJCovNet results an accuracy of 98.14%, exceeding Xception with an accuracy of 84,24%, VGG19 with an accuracy of 95.25%, and ResNet50 with accuracy of 91.53%. They worked on the Batch Normalization layer for its great benefits for increasing the model's performance.

In this [15], authors have highlighted the importance of speech signal processing in diagnosing the COVID-19 virus with the help of Recurrent Neural Network (RNN) and architecture used here is Long Short-Term Memory (LSTM) for analyzing the acoustic features of cough, breathing, and voice of the victims. They used speech corpus which was collected from 60 Healthy speakers (40 male and 20 female), and 20 COVID-19 patients (12 male and 8 female). When comparing, the best accuracy is obtained for breathing sound i.e. 98.2%, then for cough sounds, an accuracy of 97% is attained. When it comes to voices, the accuracy is only 88.2%. The final analysis predicts that we can rely in the first place on collecting cough and breathing sounds to make a COVID-19 detection system.

3. REQUIREMENTS

3.1 Software requirements:

- Windows 10
- Google Colab
- Python 3.8

3.2 Hardware Requirements:

- Intel core i3 processor
- RAM 4GB
- Hard Disk Drive 1 TB

4. METHODOLOGY

4.1 Architecture:

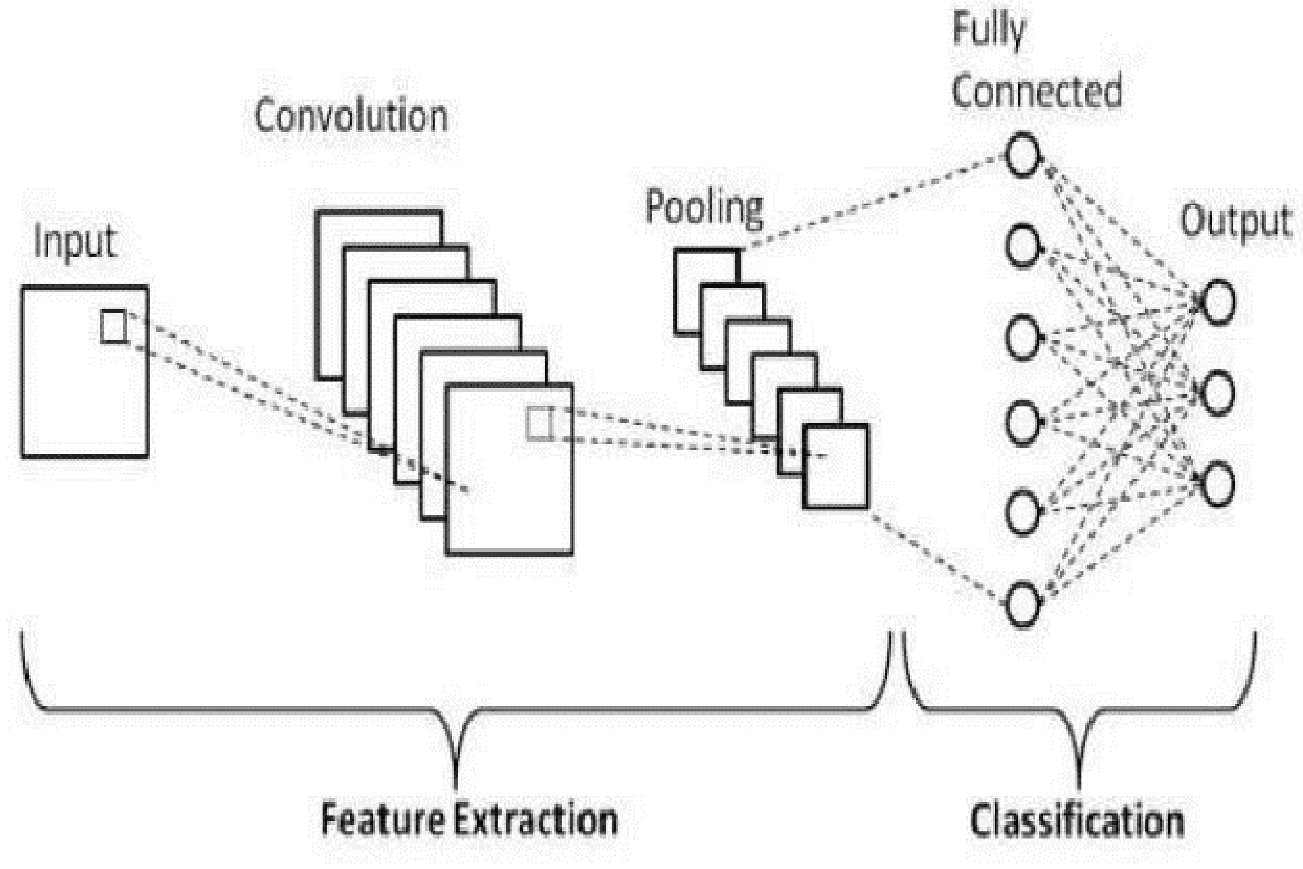


Figure 1. Architecture

Detection of Covid-19 using ResNet50 and VGG16 in Convolution Neural Network:

1. Initially, the images are given as input to the CNN model which is already trained using a dataset which contains images of human chest X-rays.

2. In convolution, filters are applied to the input images that result in activation.

3. In pooling, a feature is imbibed into CNN architecture in order to reduce spatial size, amount of parameters and computation in the network.This layer makes the model more robust.

4. The "input images", "convolution" and "pooling" steps come under feature extraction. "Fully connected" and "output" steps come under classification.

5. In fully connected neural networks all the inputs from a layer are connected to every activation unit of the next layer. Finally, we get the output.

4.2 Flow Diagram:

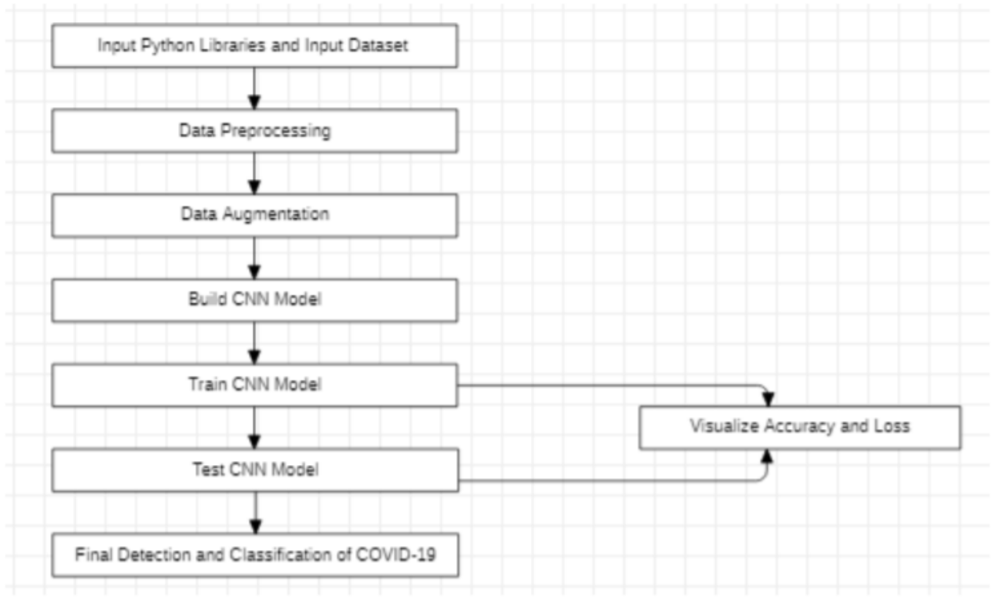

Input Python Libraries and Input Dataset
Data Preprocessing
Data Augmentation
Build CNN Model
Train CNN Model
Visualize Accuracy and Loss
Test CNN Model
Final Detection and Classification of COVID-19

Figure 2. Flow Diagram for Detection of Covid-19 using ResNet50 and VGG16 in Convolution Neural Network

5. TECHNOLOGY STACK

- Python

  A high-level, interpreted, general-purpose programming language.

- Pandas

  A software library written for the Python programming language for data manipulation and analysis.

- NumPy

  A library for adding support for large, multi-dimensional arrays and matrices.

- Keras

  An open-source software library that provides a Python interface for artificial neural networks.

- TensorFlow

  An open source machine learning framework used for carrying out high-performance numerical computations.

- Matplotlib

  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

- Shutil

  It offers high-level operations on a file like copy, create, and removal of files and directories.

- Glob

  Glob is a powerful tool in python which helps to filter through large datasets and pull out only files that are of interest.

- Os

  The os module in python provides functions for interacting with the operating system.

6. LIST OF ALGORITHMS

- **CNN - Convolutional Neural Network -** 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 hand-engineered, 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. A collection of such fields overlap to cover the entire visual area.

  CNN can run directly on a underdone image and do not need any preprocessing. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. CNN is used in image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

7. IMPLEMENTATION AND RESULTS

7.1 CODE

ResNet50

```
import warnings
warnings.filterwarnings("ignore")

"""**Mounting drive**"""

from google.colab import drive
drive.mount('/content/drive')

!unzip "/content/drive/MyDrive/data/MajorProject.zip"

"""**Import Libraries**"""

import pandas as pd
import numpy as np
import os
import shutil
import glob
import matplotlib.pyplot as plt

"""**Going through meta data**"""

covid_imgs = pd.read_excel("./COVID-19_Radiography_Dataset/COVID.metadata.xlsx")
covid_imgs.head(2)

opacity_images = pd.read_excel("./COVID-19_Radiography_Dataset/Lung_Opacity.metadata
.xlsx")
opacity_images.head(2)

normal_images = pd.read_excel("./COVID-19_Radiography_Dataset/Normal.metadata.xlsx")
```

```
normal_images.head(2)

pneumonia_images = pd.read_excel("./COVID-19_Radiography_Dataset/Viral␣Pneumonia.
metadata.xlsx")
pneumonia_images.head(2)

"""**Working with images**"""

ROOT_DIR = "/content/COVID-19_Radiography_Dataset/"
imgs = ['COVID','Lung_Opacity','Normal','Viral␣Pneumonia']

NEW_DIR = "/content/all_images/"

# Copy all my images to a new folder i.e all_images

if not os.path.exists(NEW_DIR):
  os.mkdir(NEW_DIR)

  for i in imgs:
    org_dir = os.path.join(ROOT_DIR, i+"/")

    for imgfile in glob.iglob(os.path.join(org_dir, "*.png")):
      shutil.copy(imgfile, NEW_DIR)

else:
  print("Already␣Exist")

counter = {'COVID':0,'Lung_Opacity':0,'Normal':0,'Viral␣Pneumonia':0}

for image in imgs:
  for count in glob.iglob(NEW_DIR+image+"*"):
    counter[image] += 1

# the number of images i have in each class
```

```
counter

#visualizing the number of images

plt.figure(figsize=(10,5))
plt.bar(x = counter.keys(), height= counter.values())
plt.show()

if not os.path.exists(NEW_DIR+"train_test_split/"):

  os.makedirs(NEW_DIR+"train_test_split/")



  os.makedirs(NEW_DIR+"train_test_split/train/Normal")
  os.makedirs(NEW_DIR+"train_test_split/train/Covid")



  os.makedirs(NEW_DIR+"train_test_split/test/Normal")
  os.makedirs(NEW_DIR+"train_test_split/test/Covid")



  os.makedirs(NEW_DIR+"train_test_split/validation/Normal")
  os.makedirs(NEW_DIR+"train_test_split/validation/Covid")

  # Train Data

  for i in np.random.choice(replace= False , size= 3000 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 3900 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Normal" )
    os.remove(i)
  for i in np.random.choice(replace= False , size= 900 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Covid" )
    os.remove(i)

  # Validation Data

  for i in np.random.choice(replace= False , size= 308 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 500 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Normal" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 200 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Covid" )
    os.remove(i)

  # Test Data

  for i in np.random.choice(replace= False , size= 300 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 300 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Normal" )
    os.remove(i)
  for i in np.random.choice(replace= False , size= 200 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Covid" )
    os.remove(i)

train_path  = "/content/all_images/train_test_split/train"

valid_path  = "/content/all_images/train_test_split/validation"

test_path   = "/content/all_images/train_test_split/test"


import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.resnet import preprocess_input, ResNet50
from keras.models import Model
from keras.layers import Dense, MaxPool2D, Conv2D

train_data_gen = ImageDataGenerator(preprocessing_function= preprocess_input,
                                    zoom_range= 0.2,
                                    horizontal_flip= True,
                                    shear_range= 0.2,
                                    )

train = train_data_gen.flow_from_directory(directory= train_path,
                                           target_size=(224,224))

validation_data_gen = ImageDataGenerator(preprocessing_function= preprocess_input )

valid = validation_data_gen.flow_from_directory(directory= valid_path,
                                                target_size=(224,224))

test_data_gen = ImageDataGenerator(preprocessing_function= preprocess_input )

test = train_data_gen.flow_from_directory(directory= test_path ,
                                              target_size=(224,224),
                                              shuffle= False)

# Covid +ve X-Ray is represented by 0 and Normal is represented by 1

class_type = {0:'Covid',  1 : 'Normal'}

# to visualize the images in the traing data denerator

t_img , label = train.next()

# function when called will prot the images

def plotImages(img_arr, label):
  """
  input  :- images array
  output :- plots the images
  """

  for im, l in zip(img_arr,label) :
    plt.figure(figsize= (5,5))
    plt.imshow(im, cmap = 'gray')
    plt.title(im.shape)
    plt.axis = False
    plt.show()

# function call to plot the images

plotImages(t_img, label)

"""**We will be using our model Resnet 50**"""

from keras.applications.resnet import ResNet50
from keras.layers import Flatten , Dense, Dropout , MaxPool2D
```

```
res = ResNet50( input_shape=(224,224,3), include_top= False) # include_top will consider
the new weights

for layer in res.layers:           # Dont Train the parameters again
  layer.trainable = False

x = Flatten()(res.output)
x = Dense(units=2 , activation='sigmoid', name = 'predictions' )(x)

# creating our model.

model = Model(res.input, x)

model.summary()

model.compile( optimizer= 'adam' , loss = 'categorical_crossentropy',
metrics=['accuracy'])

# implementing early stopping and model check point

from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

es = EarlyStopping(monitor= "val_accuracy" , min_delta= 0.01, patience= 3, verbose=1)

mc = ModelCheckpoint(filepath="bestmodel.h5", monitor="val_accuracy", verbose=1,
 save_best_only= True)

hist = model.fit_generator(train, steps_per_epoch= 10, epochs= 30,
validation_data= valid , validation_steps= 16, callbacks=[es,mc])

"""**Load only the best model**"""

from keras.models import load_model
model = load_model("bestmodel.h5")
```

```
"""**Seeing how our model has performed**"""

h = hist.history
h.keys()

plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'] , c = "red")
plt.title("acc␣vs␣v-acc")
plt.show()

plt.plot(h['loss'])
plt.plot(h['val_loss'] , c = "red")
plt.title("loss␣vs␣v-loss")
plt.show()

"""**Checking out the accuracy of our model**"""

acc = model.evaluate_generator(generator= test)[1]

print(f"The␣accuracy␣of␣your␣model␣is␣=␣{acc}␣%")

"""**Accuracy**"""

from keras.preprocessing import image

def get_img_array(img_path):
  """
  Input : Takes in image path as input
  Output : Gives out Pre-Processed image
  """
  path = img_path
  img = image.load_img(path, target_size=(224,224,3))
  img = image.img_to_array(img)
  img = np.expand_dims(img , axis= 0 )
```

```
    return img

# path for that new image. ( you can take it either from google or any other scource)

path = "/content/all_images/COVID-1786.png"      # you can add any image path

#predictions: path:- provide any image from google or provide image from all image
  folder

img = get_img_array(path)

res = class_type[np.argmax(model.predict(img))]
print(f"The␣given␣X-Ray␣image␣is␣of␣type␣=␣{res}")
print()
print(f"The␣chances␣of␣image␣being␣Covid␣is␣:␣{model.predict(img)[0][0]*100}␣percent")
print()
print(f"The␣chances␣of␣image␣being␣Normal␣is␣:␣{model.predict(img)[0][1]*100}␣percent")

# to display the image

plt.imshow(img[0]/255, cmap = "gray")
plt.title("input␣image")
plt.show()

"""**Grad CAM Visualization**"""

import tensorflow as tf

# this function is udes to generate the heat map of aan image

def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):

    # First, we create a model that maps the input image to the activations
    # of the last conv layer as well as the output predictions
```

```
grad_model = tf.keras.models.Model(
    [model.inputs], [model.get_layer(last_conv_layer_name).output, model.output]
)

# Then, we compute the gradient of the top predicted class for our input image
# with respect to the activations of the last conv layer

with tf.GradientTape() as tape:
    last_conv_layer_output, preds = grad_model(img_array)
    if pred_index is None:
        pred_index = tf.argmax(preds[0])
    class_channel = preds[:, pred_index]

# This is the gradient of the output neuron (top predicted or chosen)
# with regard to the output feature map of the last conv layer

grads = tape.gradient(class_channel, last_conv_layer_output)

# This is a vector where each entry is the mean intensity of the gradient
# over a specific feature map channel

pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

# We multiply each channel in the feature map array
# by "how important this channel is" with regard to the top predicted class
# then sum all the channels to obtain the heatmap class activation

last_conv_layer_output = last_conv_layer_output[0]
heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
heatmap = tf.squeeze(heatmap)

# For visualization purpose, we will also normalize the heatmap between 0 & 1

heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
```

```
    return heatmap.numpy()

"""**Now we will mask the heat map on the image**"""

import matplotlib.cm as cm

from IPython.display import Image, display

def save_and_display_gradcam(img_path , heatmap, cam_path="cam.jpg", alpha=0.4):
    """
    img input shoud not be expanded
    """

    # Load the original image

    img = keras.preprocessing.image.load_img(img_path)
    img = keras.preprocessing.image.img_to_array(img)

    # Rescale heatmap to a range 0-255

    heatmap = np.uint8(255 * heatmap)

    # Use jet colormap to colorize heatmap

    jet = cm.get_cmap("jet")

    # Use RGB values of the colormap

    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]

    # Create an image with RGB colorized heatmap

    jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)

    # Superimpose the heatmap on original image

    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)

    # Save the superimposed image

    superimposed_img.save(cam_path)

    # Display Grad CAM

    display(Image(cam_path))


# function that is used to predict the image type and the ares that are affected by
covid

def image_prediction_and_visualization(path,last_conv_layer_name = "conv5_block3_3_conv"
, model = model):
  """
  input:  is the image path, name of last convolution layer , model name
  output : returs the predictions and the area that is effected
  """

  img_array = get_img_array(path)

  heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)

  plt.title("the heat map of the image is ")
  plt.imshow(heatmap)
  plt.show()
  print()
  img = get_img_array(path)
```

```
    res = class_type[np.argmax(model.predict(img))]
    print(f"The given X-Ray image is of type = {res}")
    print()
    print(f"The chances of image being Covid is : {model.predict(img)[0][0]*100} %")
    print(f"The chances of image being Normal is : {model.predict(img)[0][1]*100} %")

    print()
    print("image with heatmap representing region on interest")

    # function call

    save_and_display_gradcam(path, heatmap)

    print()
    print("the original input image")
    print()

    a = plt.imread(path)
    plt.imshow(a, cmap = "gray")
    plt.title("Original image")
    plt.show()

"""**Prediction of COVID-19 and Normal**"""

# provide the path of any image from google or any other scource
# the path is already defigned above , but you can also provide the path here to avoid
scrolling up

# for covid image :  path:- provide any image from google or provide image from all
image folder

path = "/content/all_images/COVID-1786.png"

image_prediction_and_visualization(path)
```

```
# for normal image :  path:- provide any image from google or provide image from all
image folder

path = "/content/all_images/train_test_split/validation/Normal/Normal-10094.png"

image_prediction_and_visualization(path)

# for a healthey chest x-Ray heap map will be white thus the x-ray will look blue
```

VGG16

```
# to ignore the warnings

import warnings
warnings.filterwarnings("ignore")

"""**Mounting drive**"""

from google.colab import drive
drive.mount('/content/drive')

!unzip "/content/drive/MyDrive/data/MajorProject.zip"

import pandas as pd
import numpy as np
import os
import shutil
import glob
import matplotlib.pyplot as plt

"""**Going Through Meta** **Data**"""

covid_imgs = pd.read_excel("./COVID-19_Radiography_Dataset/COVID.metadata.xlsx")
covid_imgs.head(2)

opacity_images = pd.read_excel("./COVID-19_Radiography_Dataset/Lung_Opacity.metadata
.xlsx")
opacity_images.head(2)

normal_images = pd.read_excel("./COVID-19_Radiography_Dataset/Normal.metadata.xlsx")
normal_images.head(2)

pneumonia_images = pd.read_excel("./COVID-19_Radiography_Dataset/Viral␣Pneumonia.
metadata.xlsx")
```

```
pneumonia_images.head(2)

"""**Working with** **images**"""

ROOT_DIR = "/content/COVID-19_Radiography_Dataset/"
imgs = ['COVID','Lung_Opacity','Normal','Viral Pneumonia']

NEW_DIR = "/content/all_images/"

# Copy all my images to a new folder i.e NEW_DIR

if not os.path.exists(NEW_DIR):
  os.mkdir(NEW_DIR)

  for i in imgs:
    org_dir = os.path.join(ROOT_DIR, i+"/")

    for imgfile in glob.iglob(os.path.join(org_dir, "*.png")):
      shutil.copy(imgfile, NEW_DIR)

else:
  print("Already Exist")

"""**Visualizing the number of Images in each categories**"""

counter = {'COVID':0,'Lung_Opacity':0,'Normal':0,'Viral Pneumonia':0}

for image in imgs:
  for count in glob.iglob(NEW_DIR+image+"*"):
    counter[image] += 1

counter

plt.figure(figsize=(10,5))
plt.bar(x = counter.keys(), height= counter.values())
plt.show()

"""**We will be ignoring the lung opacity and Viral Pneumonia as its not related to
our problem statement**"""

#Creating the folder

if not os.path.exists(NEW_DIR+"train_test_split/"):

  os.makedirs(NEW_DIR+"train_test_split/")

  os.makedirs(NEW_DIR+"train_test_split/train/Normal")
  os.makedirs(NEW_DIR+"train_test_split/train/Covid")

  os.makedirs(NEW_DIR+"train_test_split/test/Normal")
  os.makedirs(NEW_DIR+"train_test_split/test/Covid")

  os.makedirs(NEW_DIR+"train_test_split/validation/Normal")
  os.makedirs(NEW_DIR+"train_test_split/validation/Covid")


  # Train Data

  for i in np.random.choice(replace= False , size= 3000 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 3900 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Normal" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 900 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/train/Covid" )
    os.remove(i)

  # Validation Data

  for i in np.random.choice(replace= False , size= 308 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 500 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Normal" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 200 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/validation/Covid" )
    os.remove(i)

  # Test Data

  for i in np.random.choice(replace= False , size= 300 , a = glob.glob(NEW_DIR+imgs[0]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Covid" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 300 , a = glob.glob(NEW_DIR+imgs[2]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Normal" )
    os.remove(i)

  for i in np.random.choice(replace= False , size= 200 , a = glob.glob(NEW_DIR+imgs[3]
+"*") ):
    shutil.copy(i , NEW_DIR+"train_test_split/test/Covid" )
    os.remove(i)

train_path  = "/content/all_images/train_test_split/train"
valid_path  = "/content/all_images/train_test_split/validation"
test_path   = "/content/all_images/train_test_split/test"

from keras.preprocessing.image import ImageDataGenerator
from keras.applications import vgg16
from keras.models import Model
from keras.layers import Dense, MaxPool2D, Conv2D
import keras

train_data_gen = ImageDataGenerator(preprocessing_function= vgg16.preprocess_input ,
 zoom_range= 0.2, horizontal_flip= True, shear_range= 0.2 , rescale= 1./255)
train = train_data_gen.flow_from_directory(directory= train_path ,
target_size=(224,224))

validation_data_gen = ImageDataGenerator(preprocessing_function= vgg16.preprocess_input,
 rescale= 1./255 )
valid = validation_data_gen.flow_from_directory(directory= valid_path ,
target_size=(224,224))

test_data_gen = ImageDataGenerator(preprocessing_function= vgg16.preprocess_input,
rescale= 1./255 )
test = train_data_gen.flow_from_directory(directory= test_path ,
target_size=(224,224), shuffle= False)

train.class_indices

class_type = {0:'Covid', 1 : 'Normal'}

t_img , label = train.next()

def plotImages(img_arr, label):
  for im, l in zip(img_arr,label) :
    plt.figure(figsize= (5,5))
    plt.imshow(im, cmap = 'gray')
    plt.title(im.shape)
    plt.axis = False
    plt.show()

plotImages(t_img, label)

"""**We will be using our model vgg16**"""

from keras.applications.vgg16 import VGG16
from keras.layers import Flatten , Dense, Dropout , MaxPool2D

vgg = VGG16( input_shape=(224,224,3), include_top= False) # include_top will consider
 the new weights

for layer in vgg.layers:           # Dont Train the parameters again
  layer.trainable = False

x = Flatten()(vgg.output)
x = Dense(units=2 , activation='sigmoid', name = 'predictions' )(x)

model = Model(vgg.input, x)

model.summary()

model.compile(optimizer='adam', loss = 'categorical_crossentropy', metrics=['accuracy'])

# implementing early stopping and model check point

from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

es = EarlyStopping(monitor= "val_accuracy" , min_delta= 0.01, patience= 3, verbose=1)
```

```
mc = ModelCheckpoint(filepath="bestmodel.h5", monitor="val_accuracy", verbose=1,
save_best_only= True)

#hist = model.fit_generator(train, steps_per_epoch= 10, epochs= 8,

validation_data= valid , validation_steps= 32)
hist = model.fit_generator(train, steps_per_epoch= 10, epochs= 30,
validation_data= valid , validation_steps= 32, callbacks=[es,mc])

# load only the best model

from keras.models import load_model
model = load_model("bestmodel.h5")

"""**Seeing how our model has performed**"""

h = hist.history
h.keys()

plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'] , c = "red")
plt.title("acc vs v-acc")
plt.show()

plt.plot(h['loss'])
plt.plot(h['val_loss'] , c = "red")
plt.title("loss vs v-loss")
plt.show()

"""**Visualizing the number of Images in each categories**"""

acc = model.evaluate_generator(generator= test)[1]

print(f"The accuracy of your model is = {acc} %")
```

```
from keras.preprocessing import image

def get_img_array(img_path):
  """
  Input : Takes in image path as input
  Output : Gives out Pre-Processed image
  """
  path = img_path
  img = image.load_img(path, target_size=(224,224,3))
  img = image.img_to_array(img)/255
  img = np.expand_dims(img , axis= 0 )

  return img

# path for that new image. ( you can take it either from google or any other scource)

path = "/content/all_images/COVID-881.png"        # you can add any image path

#predictions: path:- provide any image from google or provide image from all image
 folder

img = get_img_array(path)

res = class_type[np.argmax(model.predict(img))]
print(f"The given X-Ray image is of type = {res}")
print()
print(f"The chances of image being Covid is : {model.predict(img)[0][0]*100} percent")
print()
print(f"The chances of image being Normal is : {model.predict(img)[0][1]*100} percent")

# to display the image

plt.imshow(img[0], cmap = "gray")
plt.title("input image")
plt.show()
```

```
"""**Grad CAM Visualization**"""

import tensorflow as tf

# this function is udes to generate the heat map of aan image

def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):

    # First, we create a model that maps the input image to the activations
    # of the last conv layer as well as the output predictions

    grad_model = tf.keras.models.Model(
        [model.inputs], [model.get_layer(last_conv_layer_name).output, model.output]
    )

    # Then, we compute the gradient of the top predicted class for our input image
    # with respect to the activations of the last conv layer

    with tf.GradientTape() as tape:
        last_conv_layer_output, preds = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(preds[0])
        class_channel = preds[:, pred_index]

    # This is the gradient of the output neuron (top predicted or chosen)
    # with regard to the output feature map of the last conv layer

    grads = tape.gradient(class_channel, last_conv_layer_output)

    # This is a vector where each entry is the mean intensity of the gradient
    # over a specific feature map channel

    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
    # We multiply each channel in the feature map array
    # by "how important this channel is" with regard to the top predicted class
    # then sum all the channels to obtain the heatmap class activation

    last_conv_layer_output = last_conv_layer_output[0]
    heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)

    # For visualization purpose, we will also normalize the heatmap between 0 & 1

    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()

import matplotlib.cm as cm

from IPython.display import Image, display

# put the heatmap to image to understand the area of interest

def save_and_display_gradcam(img_path , heatmap, cam_path="cam.jpg", alpha=0.4):
    """
    img input shoud not be expanded
    """

    # Load the original image

    img = keras.preprocessing.image.load_img(img_path)
    img = keras.preprocessing.image.img_to_array(img)

    # Rescale heatmap to a range 0-255

    heatmap = np.uint8(255 * heatmap)

    # Use jet colormap to colorize heatmap
    jet = cm.get_cmap("jet")

    # Use RGB values of the colormap

    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]

    # Create an image with RGB colorized heatmap

    jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)

    # Superimpose the heatmap on original image

    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)

    # Save the superimposed image

    superimposed_img.save(cam_path)

    # Display Grad CAM

    display(Image(cam_path))


# function that is used to predict the image type and the ares that are affected by
 covid

def image_prediction_and_visualization(path,last_conv_layer_name = "block5_conv3",
model = model):
  """
  input:  is the image path, name of last convolution layer , model name
  output : returs the predictions and the area that is effected
```

```
"""



img_array = get_img_array(path)

heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)

plt.title("the heat map of the image is ")
plt.imshow(heatmap)
plt.show()
print()
img = get_img_array(path)

res = class_type[np.argmax(model.predict(img))]
print(f"The given X-Ray image is of type = {res}")
print()
print(f"The chances of image being Covid is : {model.predict(img)[0][0]*100} %")
print(f"The chances of image being Normal is : {model.predict(img)[0][1]*100} %")

print()
print("image with heatmap representing the covid spot")

# function call

save_and_display_gradcam(path, heatmap)

print()
print("the original input image")
print()

a = plt.imread(path)
plt.imshow(a, cmap = "gray")
plt.title("Original image")
plt.show()
```

```
#predictions

# provide the path of any image from google or any other scource
# the path is already defigned above , but you can also provide the path here to avoid
 scrolling up

# for covid image :

path = "/content/all_images/COVID-881.png"

image_prediction_and_visualization(path)

# for normal image :

path = "/content/all_images/train_test_split/test/Normal/Normal-10026.png"

image_prediction_and_visualization(path)
```

7.2 OUTPUT (ResNet50)

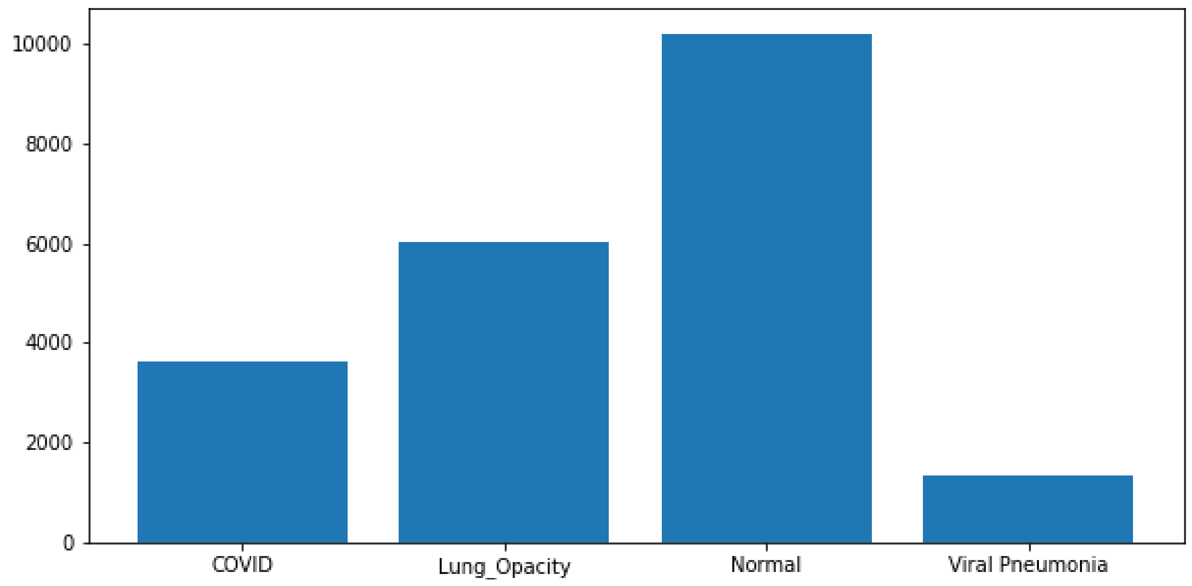


Figure 3. Graph representing count of each class

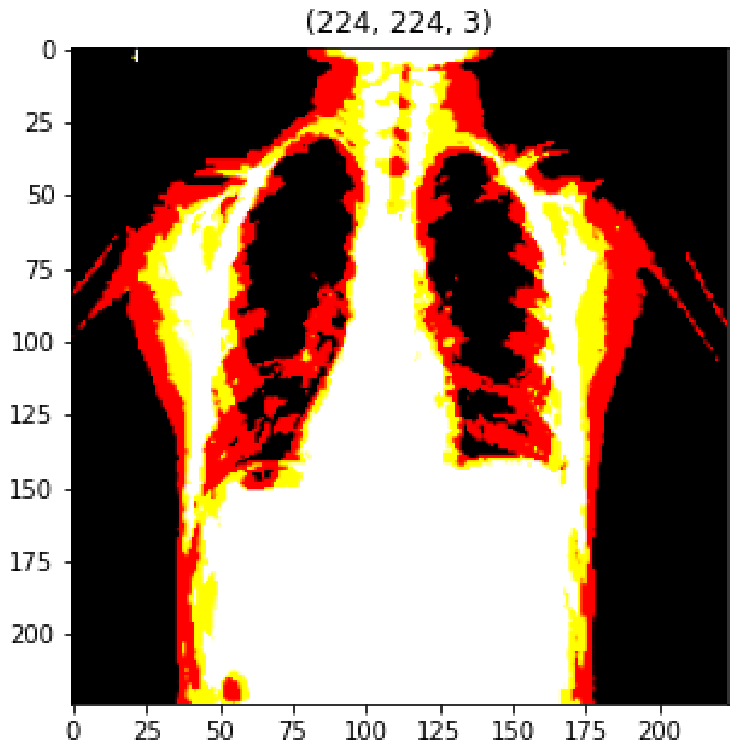


Figure 4. Clipping input data to the valid range for imshow with RGB data

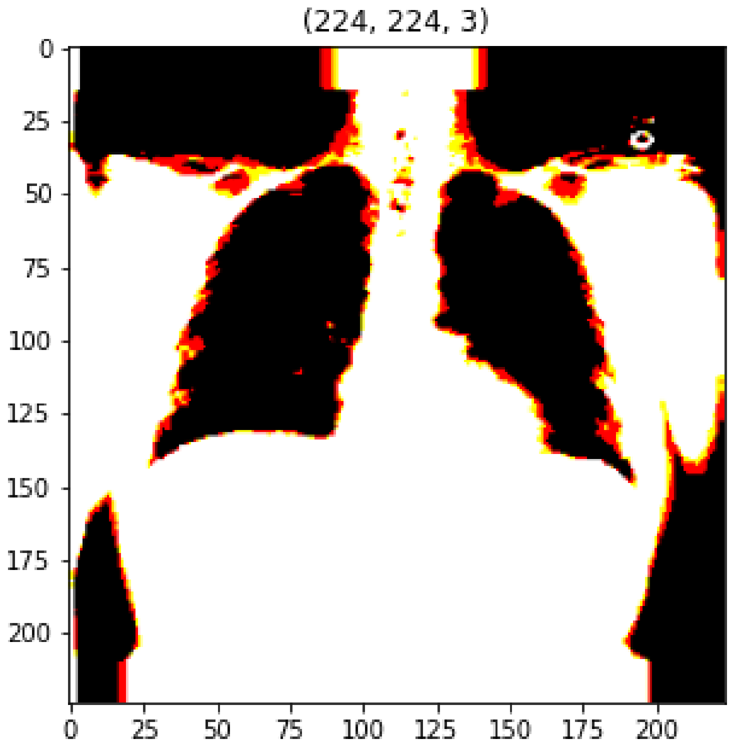


Figure 5. Clipping input data to the valid range for imshow with RGB data

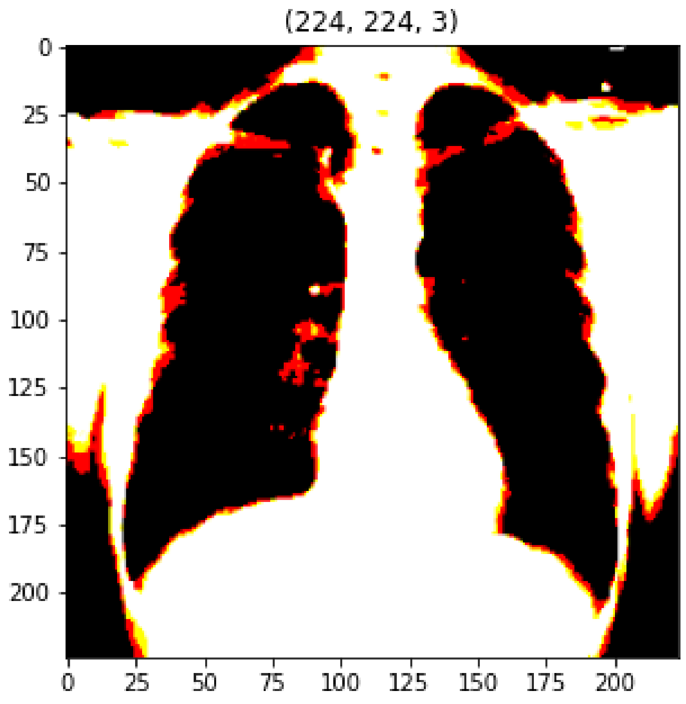


Figure 6. Clipping input data to the valid range for imshow with RGB data

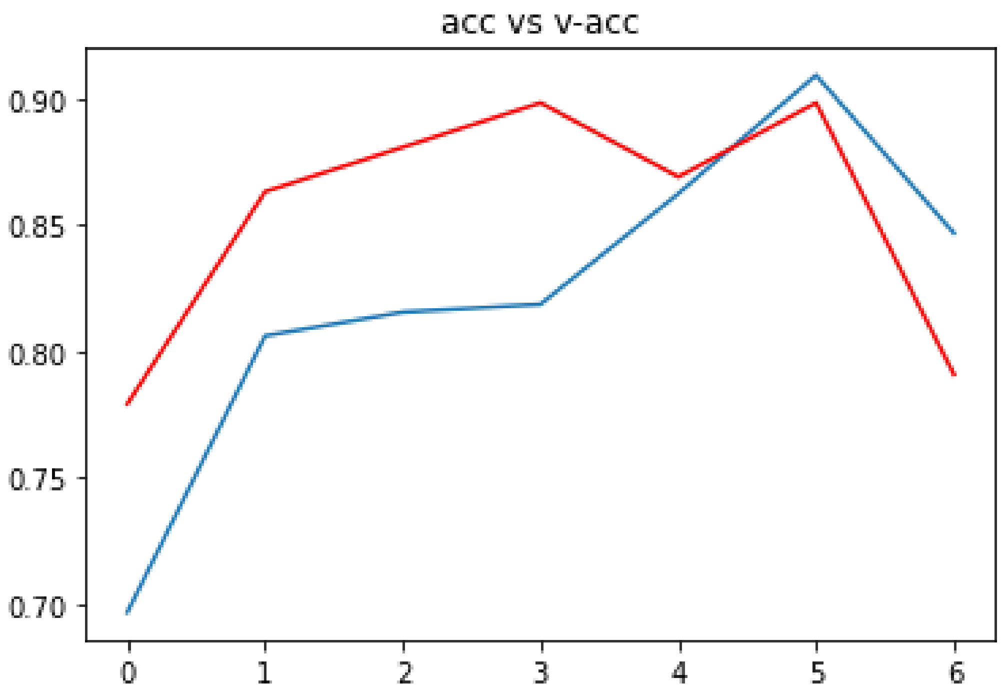


Figure 7. Graph representing accuracy vs validation accuracy

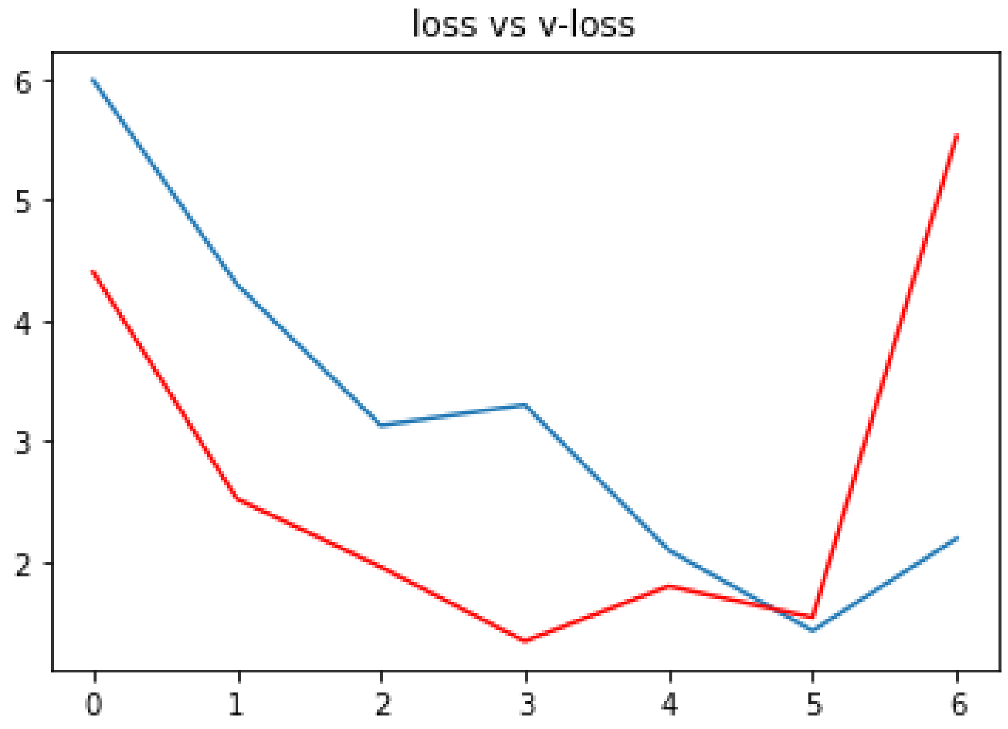


Figure 8. Graph representing loss vs validation loss

```
The given X-Ray image is of type = Covid

The chances of image being Covid is : 99.99998807907104 percent

The chances of image being Normal is : 2.405164067909027e-07 percent
```

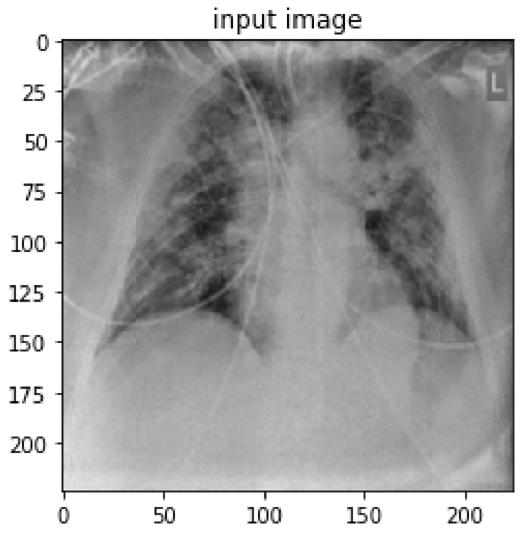


Figure 9. Figure representing input image and it's chances of being Covid and Normal

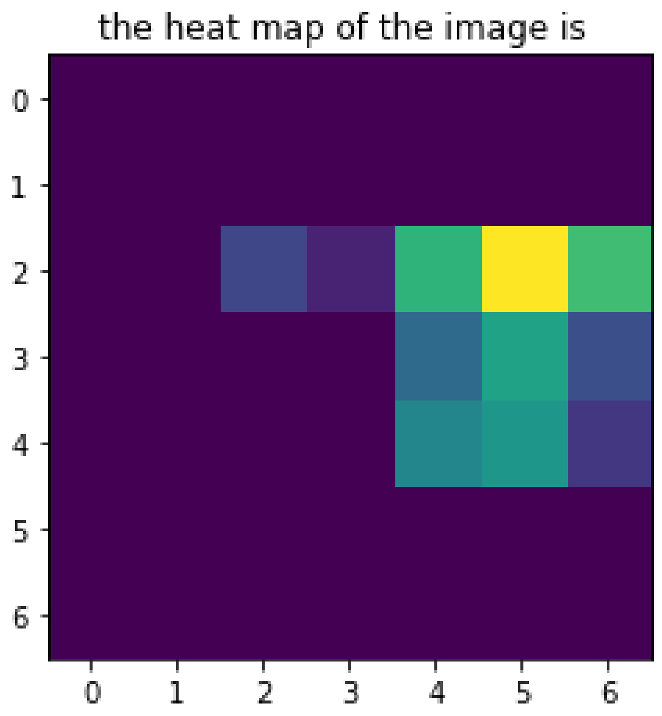
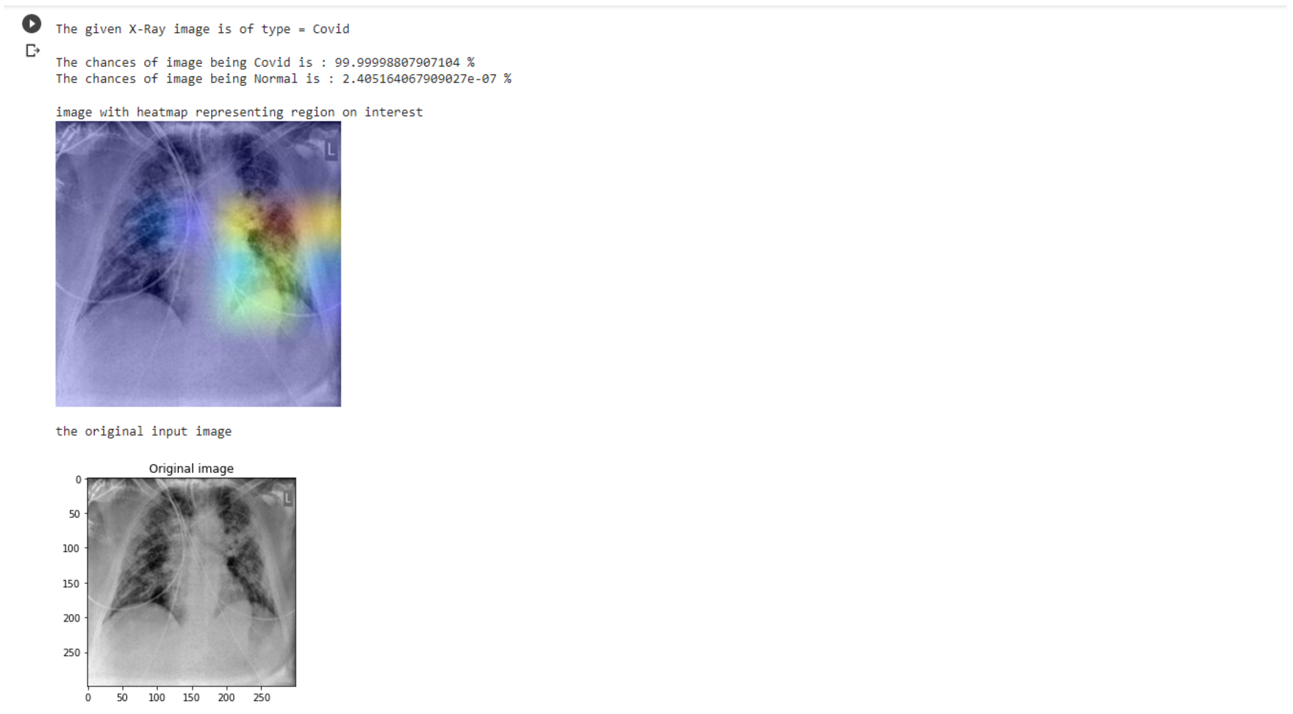


Figure 10. Figure representing heat map of the image



```
The given X-Ray image is of type = Covid

The chances of image being Covid is : 99.99998807907104 %
The chances of image being Normal is : 2.405164067909027e-07 %

image with heatmap representing region on interest
```

```
the original input image
```

Figure 11. Image with heatmap representing region on interest and it's original input image for Covid

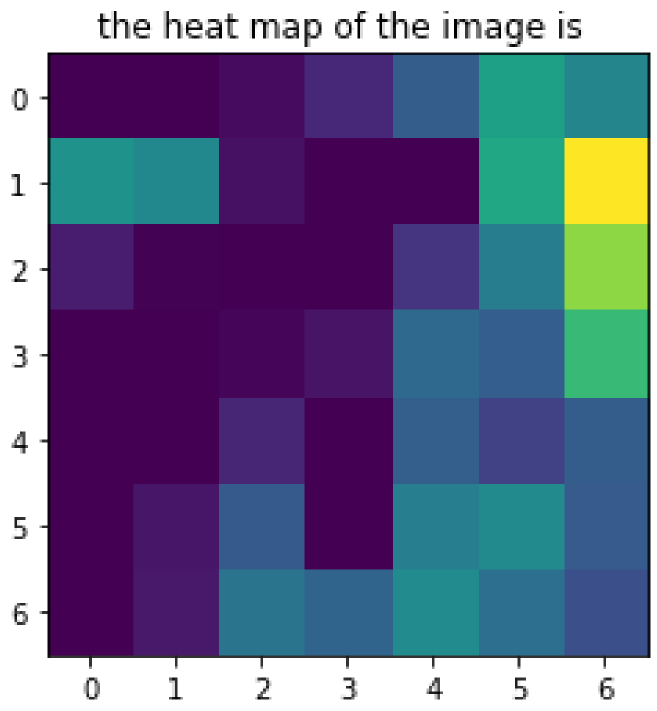
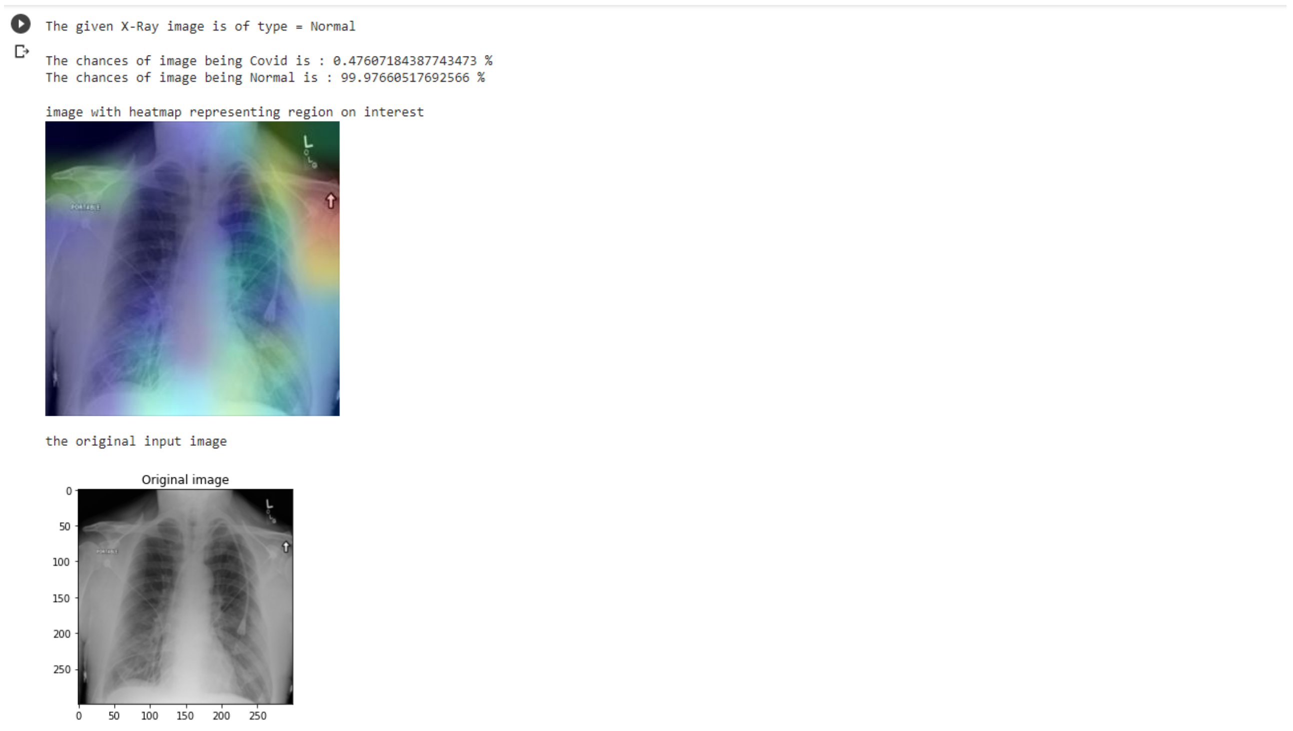


Figure 12. Figure representing heat map of the image



```
The given X-Ray image is of type = Normal

The chances of image being Covid is : 0.47607184387743473 %
The chances of image being Normal is : 99.97660517692566 %

image with heatmap representing region on interest
```

```
the original input image
```

Figure 13. Image with heatmap representing region on interest and it's original input image for Normal

OUTPUT (VGG16)

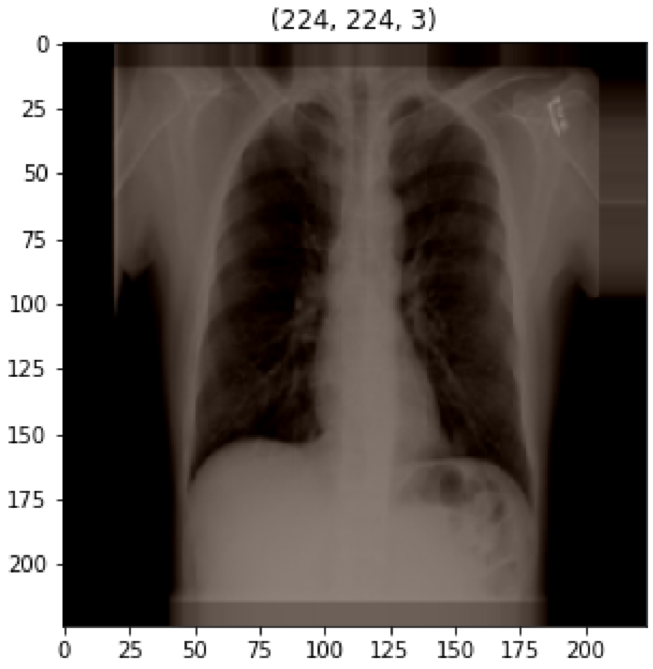


Figure 14. Clipping input data to the valid range for imshow with RGB data

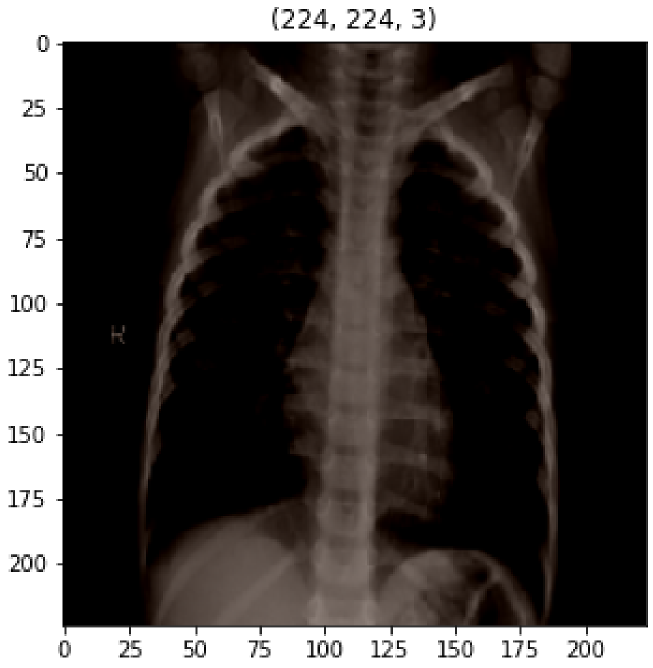


Figure 15. Clipping input data to the valid range for imshow with RGB data

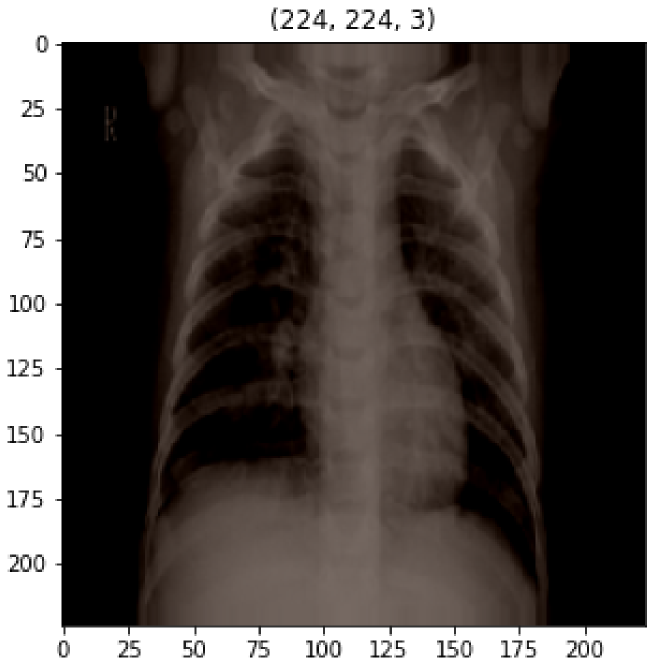


Figure 16. Clipping input data to the valid range for imshow with RGB data

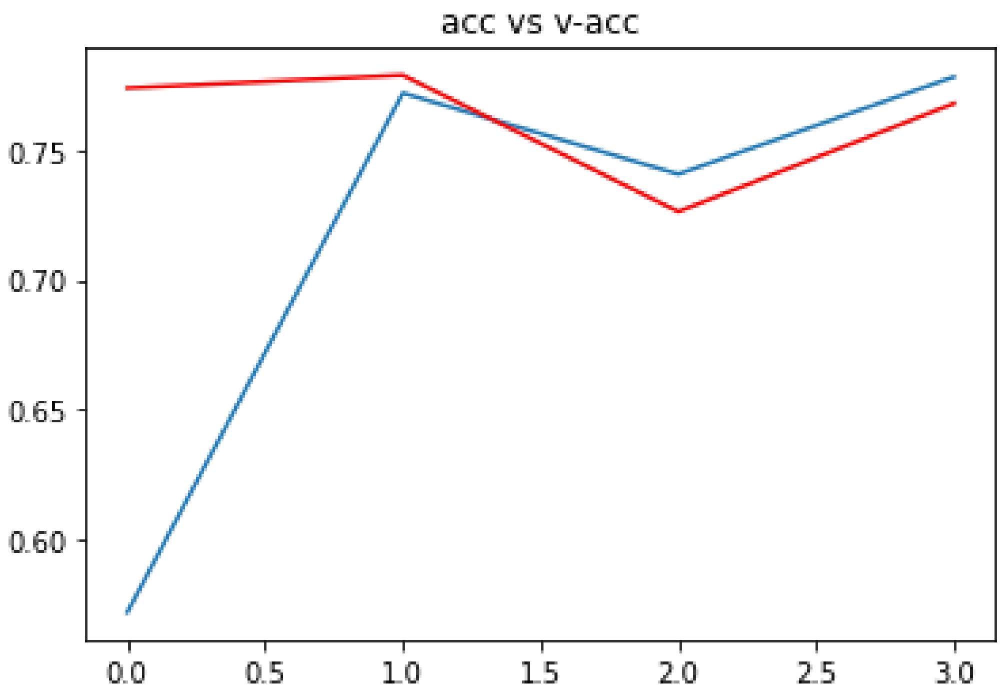


Figure 17. Graph representing accuracy vs validation accuracy

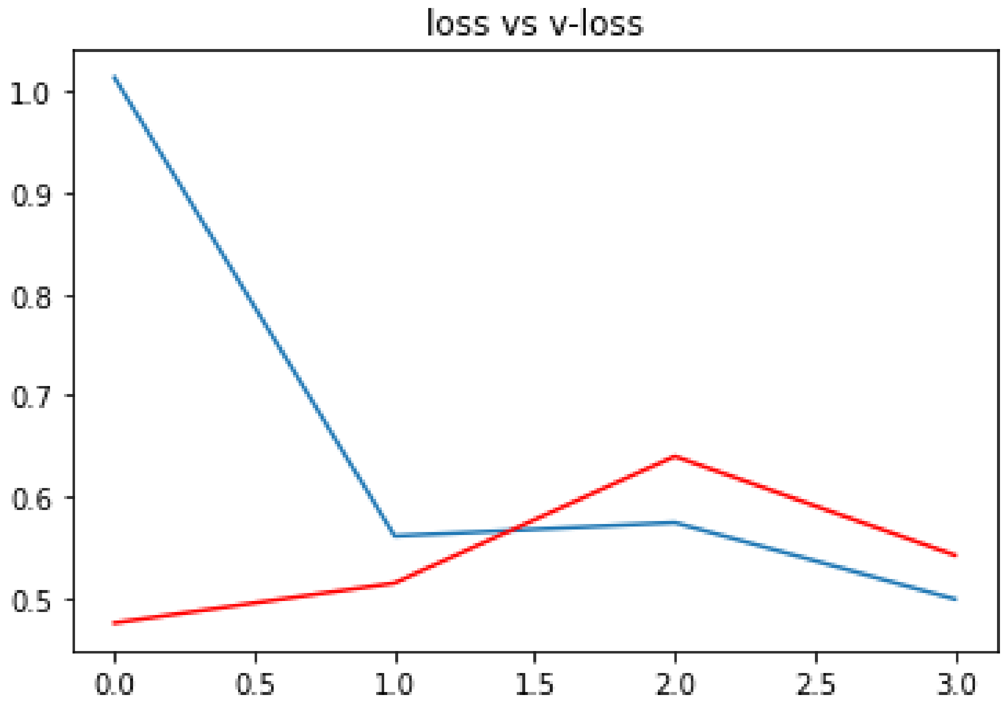


Figure 18. Graph representing loss vs validation loss

```
The given X-Ray image is of type = Covid

The chances of image being Covid is : 84.70245003700256 percent

The chances of image being Normal is : 48.7397164106369 percent
```

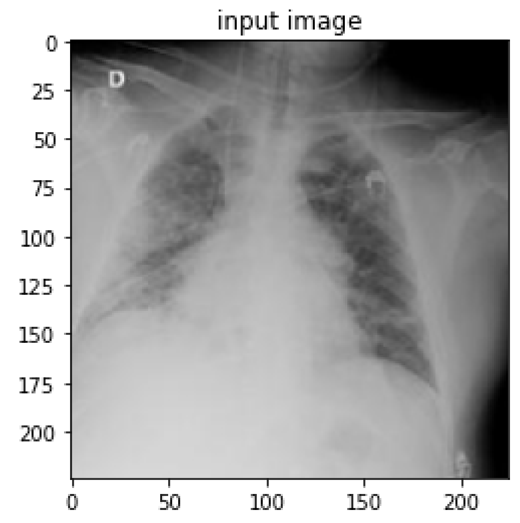


Figure 19. Figure representing input image and it’s chances of being Covid and Normal

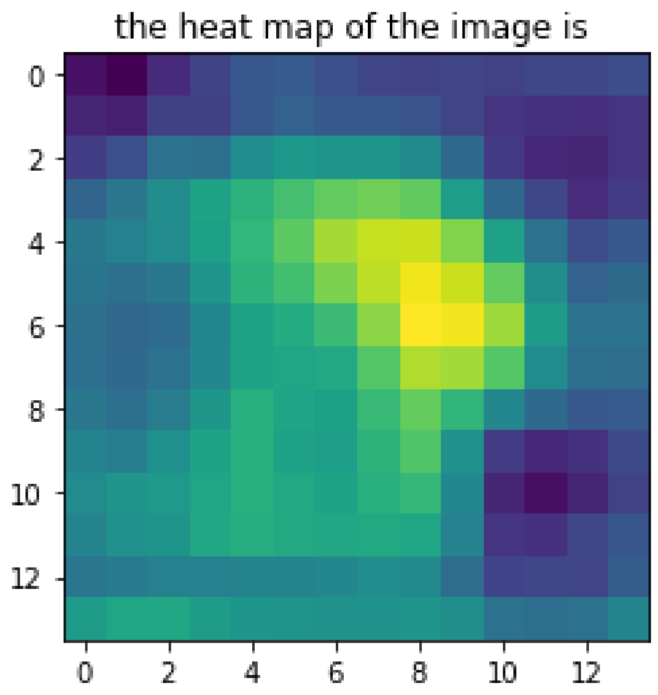
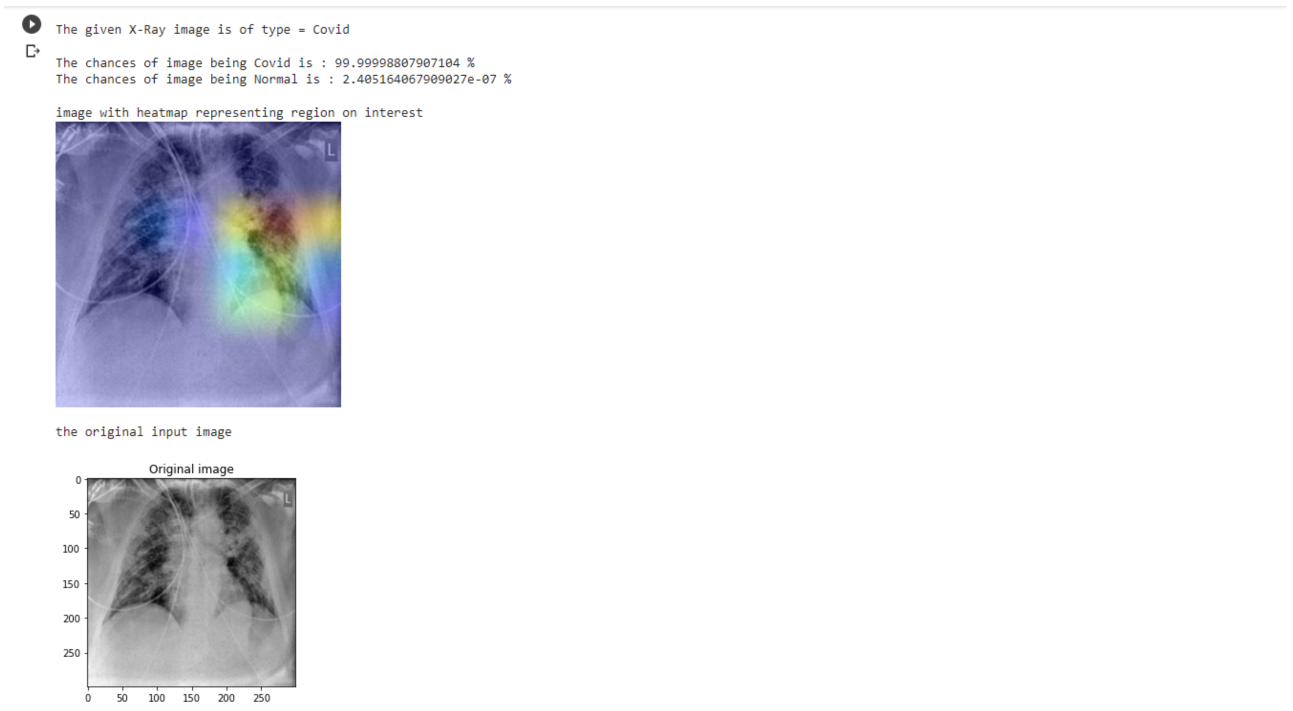


Figure 20. Figure representing heat map of the image



```
The given X-Ray image is of type = Covid

The chances of image being Covid is : 99.99998807907104 %
The chances of image being Normal is : 2.405164067909027e-07 %

image with heatmap representing region on interest
```

```
the original input image
```

Figure 21. Image with heatmap representing region on interest and it’s original input image for Covid

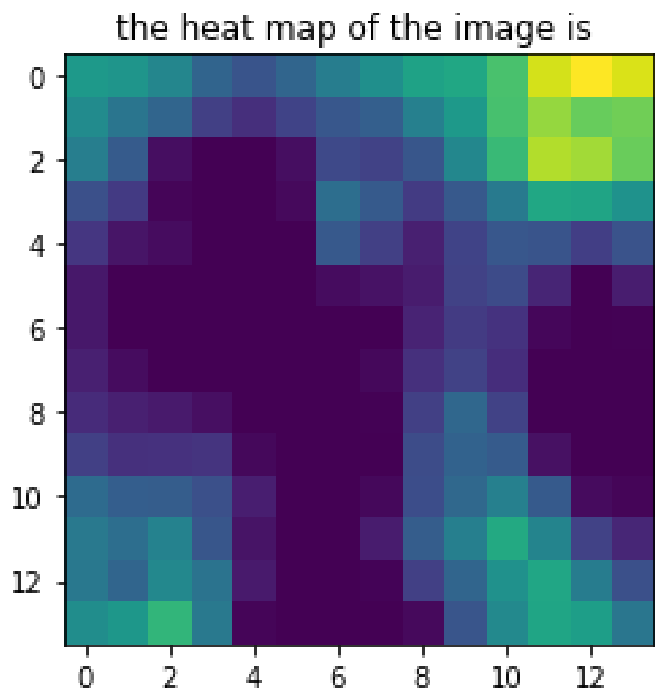
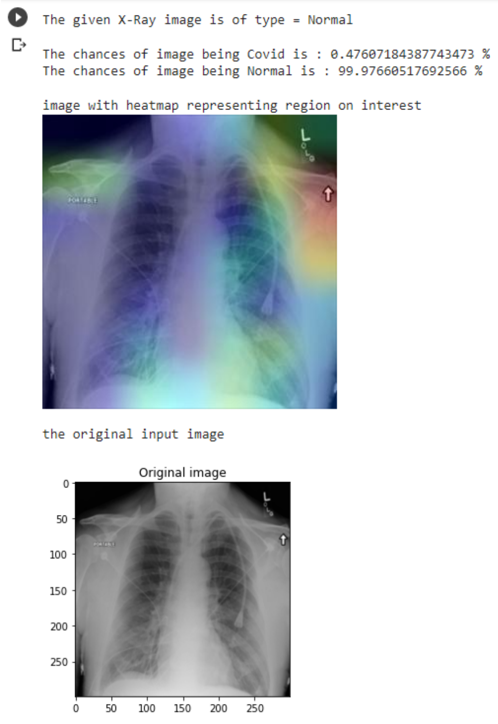


Figure 22. Figure representing heat map of the image



```
The given X-Ray image is of type = Normal

The chances of image being Covid is : 0.47607184387743473 %
The chances of image being Normal is : 99.97660517692566 %

image with heatmap representing region on interest
```

```
the original input image
```

Figure 23. Image with heatmap representing region on interest and it's original input image for Normal

8. CONCLUSION

In this report, the features of both ResNet50 and VGG16 models were thoroughly researched. Mainly accuracy was compared between ResNet50 and VGG16 models which resulted a higher accuracy for ResNet50 model.

As a result, it can be stated that the deep learning model given above accurately classifies chest X-rays for Covid-19 diagnosis. The model's loss is minimized during training, but accuracy grows at the same time through each epoch stage, yielding distinct outcomes for distinguishing Covid and Normal individuals. The data augmentation and preprocessing processes ensure that convolutional neural networks and deep neural networks' performance is not vulnerable to overfitting, ensuring that the results obtained are always consistent. The suggested model accurately predicts whether a given sample of Chest X-ray has Covid or Normal using a lesser number of convolutional layers. This is extremely useful in the medical field for detecting Covid in patients early and accurately. Early diagnosis is critical for saving a person's life by ensuring that the patient receives effective and timely treatment.

9. REFERENCES

[1] E. Irmak, "A Novel Deep Convolutional Neural Network Model for COVID-19 Disease Detection," 2020 Medical Technologies Congress (TIPTEKNO), 2020, pp. 1-4, doi: 10.1109/TIPTEKNO50054.2020.9299286.

[2] S. D. Thepade and K. Jadhav, "Covid19 Identification from Chest X-Ray Images using Local Binary Patterns with assorted Machine Learning Classifiers," 2020 IEEE Bombay Section Signature Conference (IBSSC), 2020, pp. 46-51, doi: 10.1109/IBSSC51096.2020.9332158.

[3] S. Singh, P. Sapra, A. Garg and D. K. Vishwakarma, "CNN based Covid-aid: Covid 19 Detection using Chest X-ray," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1791-1797, doi: 10.1109/ICCMC51019.2021.9418407.

[4] T. Chakravorti, V. K. Addala and J. S. Verma, "Detection and Classification of COVID 19 using Convolutional Neural Network from Chest X-ray Images," 2021 6th International Conference for Convergence in Technology (I2CT), 2021, pp. 1-6, doi: 10.1109/I2CT51068.2021.9418221.

[5] A. K. Siddhu, A. Kumar and S. Kundu, "Review Paper for Detection of COVID-19 from Medical Images and/ or Symptoms of Patient using Machine Learning Approaches," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), 2020, pp. 39-44, doi:10.1109/SMART50582.2020.9336799.

[6] Y. Yari, T. V. Nguyen and H. Nguyen, "Accuracy Improvement in Detection of COVID-19 in Chest Radiography," 2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS), 2020, pp. 1-6, doi: 10.1109/ICSPCS50536.2020.9310066.

[7] S. Thabasum Aara, A. Pandian K, T. S. Sai Kumar and A. Prabalakshmi, "A Novel Convolutional Neural Network Architecture to Diagnose COVID-19," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 2021, pp. 595-599, doi: 10.1109/ICSPC51351.2021.9451701.

[8] Y. Khan, P. Khan, S. Kumar, J. Singh and R. M. Hegde, "Detection and Spread Prediction of COVID-19 from Chest X-ray Images using Convolutional Neural Network-Gaussian Mixture Model," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-6, doi: 10.1109/INDICON49873.2020.9342159.

[9] M. Berrimi, S. Hamdi, R. Y. Cherif, A. Moussaoui, M. Oussalah and M. Chabane, "COVID-19 detection from Xray and CT scans using transfer learning," 2021 International Conference of Women in Data Science at Taif University (WiDSTaif), 2021, pp. 1-6, doi: 10.1109/WiDSTaif52235.2021.9430229.

[10] M. Sevi and İ. AYDIN, "COVID-19 Detection Using Deep Learning Methods," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1-6, doi: 10.1109/ICDABI51230.2020.9325626.

[11] E. Gambhir, R. Jain, A. Gupta and U. Tomer, "Regression Analysis of COVID-19 using Machine Learning Algorithms," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 65-71, doi: 10.1109/ICOSEC49089.2020.9215356.

[12] K. Foysal Haque, F. Farhan Haque, L. Gandy and A. Abdelgawad, "Automatic Detection of COVID-19 from Chest X-ray Images with Convolutional Neural Networks," 2020 International Conference on Computing, Electronics Communications Engineering (iCCECE), 2020, pp. 125-130, doi: 10.1109/iCCECE49321.2020.9231235.

[13] R. Punia, L. Kumar, M. Mujahid and R. Rohilla, "Computer Vision and Radiology for COVID-19 Detection," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154088.

[14] J. C. Sangidong, H. D. Purnomo and F. Y. Santoso, "Application of Deep Learning for Early Detection of COVID-19 Using CT-Scan Images," 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2021, pp. 61-65, doi: 10.1109/EIConCIT50028.2021.9431887.

[15] A. Hassan, I. Shahin and M. B. Alsabek, "COVID-19 Detection System using Recurrent Neural Networks," 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), 2020, pp. 1-5, doi: 10.1109/CCCI49893.2020.9256562.

[16] Dataset : https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

GLOSSARY

Term	**Definition**
CNN	A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.
VGG16	VGG16 is also known as Visual Geometry Group .It is a simple and widely used CNN Architecture used for ImageNet, a large visual database project used in visual object recognition software research.The number 16 in the name VGG refers to the fact that it is 16 layers deep neural network. The VGG16 is a extensive network and has a total of around 138 million parameters.
ResNet50	ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations.