

A  
Project Report  
on

# **DEEP LEARNING ANALYSIS OF COVID-19 USING CHEST CT SCAN**

Submitted for partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**MOHD. ATAUR RAHMAN (2451-17-733-026)**  
**SAHITHYA NAMANI (2451-17-733-027)**  
**LALITA SNIGDHA AKKAPEDDI (2451-17-733-054)**

Under the guidance of

**Mrs. B. SARITHA**  
Associate Professor  
Department of CSE



**Maturi Venkata Subba Rao (MVSr) ENGINEERING COLLEGE**

Department of Computer Science and Engineering  
(Affiliated to Osmania University & Recognized by AICTE)  
Nadargul, Saroor Nagar Mandal, Hyderabad – 501 510  
Academic Year: 2020-2021



## CERTIFICATE

*This is to certify that the project work entitled “**Deep Learning Analysis on COVID-19 using Chest CT Scans**” is a bonafide work carried out by **Mr. Mohd. Ataur Rahman (2451-17-733- 026)**, **Ms. Sahithya Namani (2451-17-733-027)**, **Ms. Lalita Snigdha Akkapeddi(2451-17-733-054)** in partial fulfilment of the requirements for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** from **Maturi Venkata Subba Rao (MVSR) Engineering College**, affiliated to **OSMANIA UNIVERSITY, Hyderabad**, during the **Academic Year 2020-21** under our guidance and supervision.*

*The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma to the best of our knowledge and belief..*

### **Internal Guide**

Mrs B Saritha  
Associate Professor  
Department of CSE  
MVSREC.

### **Head of the Department**

Prof Prasanna Kumar  
Professor & Head  
Department of CSE  
MVSREC.

## **DECLARATION**

This is to certify that the work reported in the present project entitled “Deep Learning Analysis on COVID-19 using Chest CT Scans” is a record of bonafide work done by us in the Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University during the Academic Year 2020-21. The reports are based on the project work done entirely by us and not copied from any other source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mohd. Ataur Rehman**  
(2451-17-733-026)

**Sahithya Namani**  
(2451-17-733-027)

**Lalita Snigdha Akkapeddi**  
(2451-17-733-054)

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude and indebtedness to our project external guides **Dr. V. Hanuman Prasad, Professor of Radiology, Government Medical College, Mahabubnagar** and **Dr. P. Srinivas, Director of Govt. Medical College, Mahabubnagar** and our internal guide **Mrs. B. Saritha, Associate Professor** for their valuable suggestions and interest throughout the course of this project.

We are also thankful to our principal **Dr. G. Kanaka Durga** and **Mr. J. Prasanna Kumar**, Professor and Head, Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Hyderabad for providing excellent infrastructure and a nice atmosphere for completing this project successfully as a part of our B.E. Degree (CSE).

We would like to thank our project coordinator **Mrs. K. Padma** for their constant monitoring, guidance and support. We convey our heartfelt thanks to the Faculty Coordinators **Prof. J Prasanna Kumar, G Vijay Kumar, Dr. Akhil Khare, Dr. B Sandhya, Dr. Sesham Anand, Dr. Rajesh Kulkarni, Dr. D Sirisha** for helping us complete our project by giving valuable suggestions.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Mohd. Atuar Rehman (2451-17-733-026)

Sahithya Namani (2451-17-733-027)

Lalita Snigdha Akkapeddi (2451-17-733-054)

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

### VISION

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societal problems.

### MISSION

- To make learning process exciting, stimulating and interesting.
- To impart adequate fundamental knowledge and soft skills to students.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity in students.
- To develop economically feasible and socially acceptable software.

### PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

**PEO-1:** Demonstrate technical competence to successfully execute industry related software projects as a team member, leader or entrepreneur to meet customer business objectives.

**PEO-2:** Engage in life-long learning process by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.

**PEO-3:** Advance in their professional careers by understanding the impact of computing on society or environment to make technical contributions using a multidisciplinary and ethical approach.

### PROGRAM OUTCOMES(POs)

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principle and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinary domains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs through a holistic professional development leading to pioneering careers or entrepreneurship

## **COURSE OBJECTIVES AND OUTCOMES**

### **Course Objectives:**

- Understand the significance of survey in necessary domains for problem identification.
- Understand to map requirements into software specification.
- Learn to apply concepts of software engineering for design of the identified real world problem
- Improve the coding capabilities by implementing the various modules of project
- Comprehend the suitable documentation procedure for a technical project.
- Exhibit effective communication and presentation skills.
- Learn the process of planning the complete lifecycle of a project

### **Course Outcomes:**

- CO1:** Review recent advancements to formulate a precise problem statement with applications towards society.
- CO2:** Identify system requirements, explore design alternatives to design software based solution within the scope of project.
- CO3:** Implement, test and build the solution using contemporary technologies and tools.
- CO4:** Exhibit effective communication to present ideas clearly and produce well-structured technical report.
- CO5:** Demonstrate qualities necessary to work in a team and execute project as per plan.

## ABSTRACT

The COVID-19 pandemic worldwide is caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). It has created havoc all over the world. Various clinical studies and researchers across the globe have established that Chest CT Scans are providing accurate clinical diagnosis on detection of COVID-19. As the traditional gold standard RT-PCR Testing methodology might give more false positive and false negative results than the desired rates.

Automation of detection of covid-19 using CT Scans is crucial for effective containment and management as many countries face an acute shortage of trained radiologists. And CT Scan has an edge over X-ray images as they produce test results nearly immediately. The goal of the project is to detect COVID-19 on chest CT Scan images. The key factors distinguishing the presence of COVID-19 within all the CT Scans (inclusive of both COVID positive and COVID negative images) are Round Ground Glass Opacity, Reverse Halo Sign, Crazy Paving and Tree-in-bud appearance.

AI has proven to be the driving force in developing various COVID-19 management tools. Among various methods available in AI, Deep Learning methodologies will make it possible to build an image classifier to achieve our task. We have developed a Deep Convolutional Neural Network for classifying COVID and Non-COVID CT Scan images. We used transfer learning on the fully connected layers of an existing convolutional neural network named VGG-19 which was pre-trained for image classification on ImageNet data. An overall training accuracy of 93% was achieved. The model was deployed into production using Heroku Cloud and Python Flask Web Framework.

Mohd. Ataur Rahman (2451-17-733-026)

Sahithya Namani (2451-17-733-027)

Lalita Snigdha Akkapeddi (2451-17-733-054)



# TABLE OF CONTENTS

<b>CONTENTS</b>	<b>PAGE NO.s</b>
Certificate	i
Declaration	ii
Acknowledgement	iii
Vision & Mission , PEOs, POs and PSOs	iv
Course Objectives & Outcomes	vi
Abstract	vii
Table of Contents	viii
List of Figures	x
List of Tables	xii
List of Abbreviations	xiii

## **CHAPTERS**

### **1. INTRODUCTION**

1.1 Problem statement	3
1.2 Objective	4
1.3 Motivation	4-5
1.4 Scope	5
1.5 Software and Hardware Requirements	5-7

### **2. LITERATURE SURVEY**

2.1 Survey of Major Area Relevant to Project	8-10
2.2 Techniques and Algorithms Relevant with respect to Project	
2.2.1 Transfer Learning	10-11
2.2.2 Transfer Learning Approach	12
2.2.3 Transfer Learning with Image Data	12-13
2.2.4 VGG-19	
2.2.4.1 Architecture	14
2.2.4.2 Architecture Walkthrough	14-15
2.2.4.3 Layers in VGGNet	16-20
2.3. Applications	20-21

### **3. SYSTEM DESIGN**

3.1 System Architecture	22-23
3.2 Modules	
3.2.1 Module-1 : Train and Test Module	24
3.2.2 Module-2 : Deployment Module	24

### **4. IMPLEMENTATION**

4.1 Environmental Setup	
4.1.1 PyCharm	
4.1.1.1 Features provided by PyCharm	25
4.1.1.2 Steps to Install PyCharm	25-16
4.1.2 Git for PyCharm	
4.1.2.1 Commit and push changes to Git Repository	26
4.1.2.2 Set your Git Username	26
4.1.3 Heroku	
4.1.3.1 Heroku Setup	27
4.1.3.2 Using Heroku with Python	27-28

4.2 Implementation of each Module	29-34
4.3 Integration and Deployment	
4.3.1 Integration	34-38
4.3.2 Heroku setup for deploying Flask app using PyCharm	38-40
<b>5. EVALUATION</b>	
5.1 Dataset	41-42
5.2 Evaluation Procedure	
5.2.1 Classification report	42-43
5.2.2 Confusion Matrix	43-44
5.2.3 ROC Curve	44
5.2.4 Accuracy Loss Plot	45
5.3 Test cases	
5.3.1 COVID Positive Prediction	46-47
5.3.2 COVID Negative Prediction	47
5.4 Observation & Results	
5.4.1 Observations	48-58
5.4.2 Results	58
<b>6. CONCLUSION &amp; FUTURE ENHANCEMENTS</b>	59
<b>REFERENCES</b>	60
<b>APPENDIX</b>	61-66

## LIST OF FIGURES

Figure No.	Figure Name	Page No.
1.1	Transmission of COVID-19	1
1.2	Major diagnostic methods reported for the detection of SARS-CoV-2	3
2.1	COVID-19 lung images (left) and normal lung images (right) with their heatmaps	9
2.2	Transfer Learning Strategy	11
2.3	Comparison of different VGG variants	13
2.4	Details of the 19 layers of VGG19 network 21 used for feature extraction	14
2.5	Details on the VGG19 architecture. For each layer, number of filters, parameters and activations	15
2.6	Activation Function	19
2.7	CO-RADS* Levels	21
3.1	Flow Chart of the system	22
3.2	Detailed System Architecture	23
3.3	Flow Chart for Module-1	24
3.4	Flow Chart for Module-2	24
4.1	Steps in developing deep learning model	29
4.2	Code snippet for Normalization	30
4.3	Code snippet for Train and Test Split	30
4.4	Features to be extracted from input images	31
4.5	Architecture for deploying Flask app into production using Heroku	40
5.1	Dataset description	41
5.2	Positive COVID-19 CT Scan images	41
5.3	Negative COVID-19 CT Scan images	42
5.4	Classification report of our model with Softmax Activation function and training batch size 32	43
5.5	Confusion Matrix of our model with Softmax activation function and batch size 32	44
5.6	ROC curve of our model with Softmax activation function and batch size 32	44
5.7	Accuracy Vs Epoch Plot	45
5.8	Loss Vs Epoch Plot	45
5.9	Home Screen GUI of our application	46
5.10	Selecting an input image	46
5.11	COVID prediction made by the model	47
5.12	Non COVID Prediction made by the model	47
5.13	Classification report for model with Softmax function	48
5.14	Classification report for model with Sigmoid function	49
5.15	Classification report for model with ReLU function	49
5.16	Accuracy and Loss Plots for model with Softmax function	49
5.17	Accuracy and Loss Plots for model with Sigmoid function	49
5.18	Accuracy and Loss Plots for model with ReLU function	50
5.19	Confusion Matrix for model with Softmax function	50
5.20	Confusion Matrix for model with Sigmoid function	50

5.21	Confusion Matrix for model with ReLU function	51
5.22	Classification report for model with categorical cross entropy function	52
5.23	Classification report for model with binary cross entropy function	52
5.24	Accuracy and Loss plots for model with categorical cross entropy function	53
5.25	Accuracy and Loss plots for model with binary cross entropy function	53
5.26	Confusion matrix for model with categorical cross entropy function	53
5.27	Confusion matrix for model with binary cross entropy function	54
5.28	Classification report for model with batch size - 8	55
5.29	Classification report for model with batch size - 16	55
5.30	Classification report for model with batch size - 32	55
5.31	Accuracy and Loss plots for model with batch size - 8	56
5.32	Accuracy and Loss plots for model with batch size - 16	56
5.33	Accuracy and Loss plots for model with batch size - 32	56
5.34	Confusion Matrix for model with batch size - 8	57
5.35	Confusion Matrix for model with batch size - 16	57
5.36	Confusion Matrix for model with batch size - 32	57
5.37	Classification report for VGG-19 model with Softmax activation function, Categorical cross entropy loss function, and training batch size - 32	58

## LIST OF TABLES

Table No.	Table Name	Page No.
2.1	Detailed VGG-19 Architecture	16
5.1	Metrics obtained using different activation functions	48
5.2	Metrics obtained using different loss functions	51
5.3	Metrics obtained using different batch size	54

## LIST OF ABBREVIATIONS

1. COVID-19	-	Coronavirus disease of 2019
2. CORADS	-	Coronavirus disease 2019 Reporting and Data System
3. VGG	-	Visual Geometry Group
4. ReLU	-	Rectified Linear Unit
5. MERS	-	Middle East Respiratory Syndrome
6. ARDS	-	Acute Respiratory Distress Syndrome
7. SARS	-	Severe Acute Respiratory Syndrome
8. AI	-	Artificial Intelligence
9. CNN	-	Convolutional Neural Networks
10. PaaS	-	Platform as a Service
11. IDE	-	Integrated Development Environment
12. ResNet	-	Residual Neural Network
13. DenseNet	-	Densely Connected Convolutional Networks
14. Xception	-	Xtreme Inception
15. ILSVRC	-	ImageNet large scale visual recognition challenge
16. IoT	-	Internet of Things
17. FC	-	Fully Connected

## 1. INTRODUCTION

In late December 2019, previously unidentified coronavirus, currently named as the 2019 novel coronavirus, emerged from Wuhan, China, and resulted in a formidable outbreak in many cities in China and expanded globally. The disease is officially named as Coronavirus Disease-2019 (COVID-19, by WHO). The outbreak of coronavirus disease 2019 (COVID-19) has created a global health crisis that has had a deep impact on the way we perceive our world and our everyday lives. COVID-19 has proliferated to more than 213 countries and territories in the world.

COVID-19 is a potential zoonotic disease with low to moderate (estimated 2%–5%) mortality rate. Person-to-person transmission may occur through airborne droplet inhalation or contact transmission and if there is a lack of stringent infection control or if no proper personal protective equipment is available, it may jeopardize the front-line healthcare workers.

SARS-COV-2 is part of the Coronavirus family that includes the common cold, Severe Acute Respiratory Syndrome (SARS) and Middle East Respiratory Syndrome (MERS). These viruses can cause Acute Respiratory Distress Syndrome (ARDS), bilateral pneumonia and pulmonary failure leading to mortality. Early identifying, isolation of COVID-19 patients are extremely important to prevent the spread of the disease, and medical attention for patients is a key strategy for a better management of this pandemic.

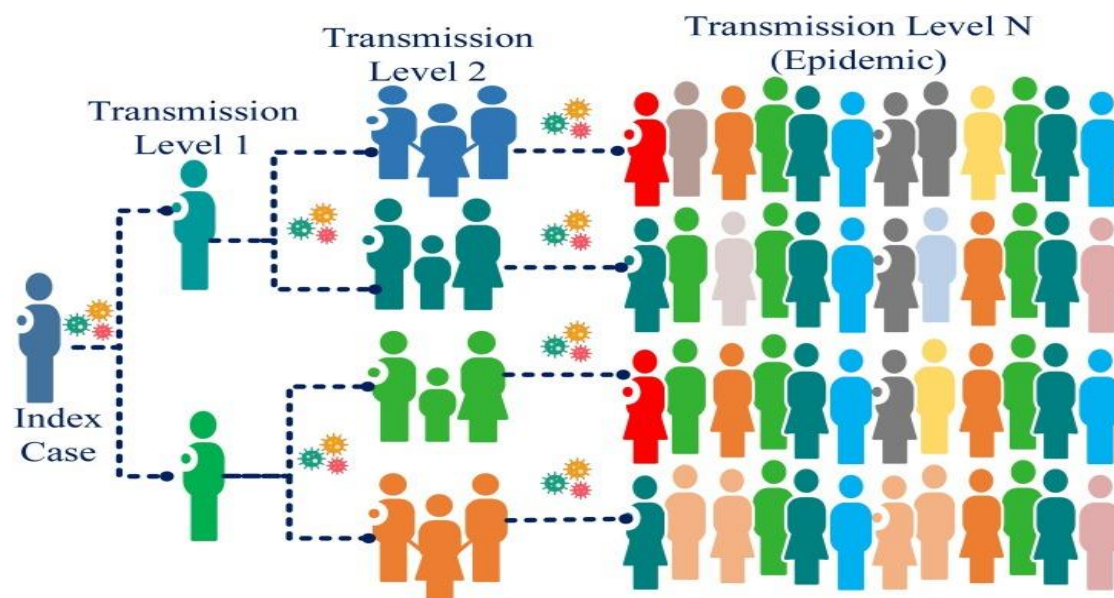


Fig.1.1 Transmission of COVID-19

Reverse-transcription polymerase chain reaction (RT-PCR) tests are currently the gold standard diagnostic tool for COVID-19. RT-PCR assays can be performed on nasopharyngeal, oropharyngeal swabs, sputum, blood samples, body fluids, stool samples and bronchoalveolar lavage fluid. RT-PCR is limited by its low sensitivity and associated relatively high number of false-negative rates, which can lead to the erroneous assumption that an infected person to be categorised as a non-infected person ultimately leading to an undetected transmission risk in either the community or within an institutional setting and also results in the suspected patient to be tested multiple times for achieving convincing diagnosis. To efficiently utilize the scarce RT-PCR resources as well as to achieve better accuracy in COVID-19 diagnosis, doctors are also relying on additional medical imaging technologies.

Among the medical imaging technologies available, Computed Tomography scan (CT scan) and X-ray images are widely used for the detection of COVID-19. The advantages of CT scan over X-ray includes the large amount of data a CT scan provides, the ability of the physician to manipulate the data into various views without additional imaging of the patient, and the ability to selectively enhance or remove structures from the images and higher sensitivity of CT scan over X-ray. CT scan has become a standard of care in the diagnosis and assessment of a variety of respiratory conditions and optimises the management process. Although CT scans are not routinely used to diagnose ARDS, certain complications relating to mechanical ventilation, including pneumonia, pneumothorax and emphysema, are sometimes identified by CT but not chest radiography. In addition, CT imaging can be used to identify lung atelectasis due to poor positioning of the endotracheal tube as well as potentially directing ventilation to achieve optimal pressures and air recruitment. This is important in relation to COVID-19 since patients in ICU require optimisation of ventilatory settings and it is increasingly recognised that prone ventilation appears favourable. Nevertheless, in the clinical setting, the benefits of routine CT imaging must be weighed against the considerable practicalities and risks, including that of infection transmission associated with transporting a patient from the intensive care unit, or elsewhere in the hospital, to the radiology department.



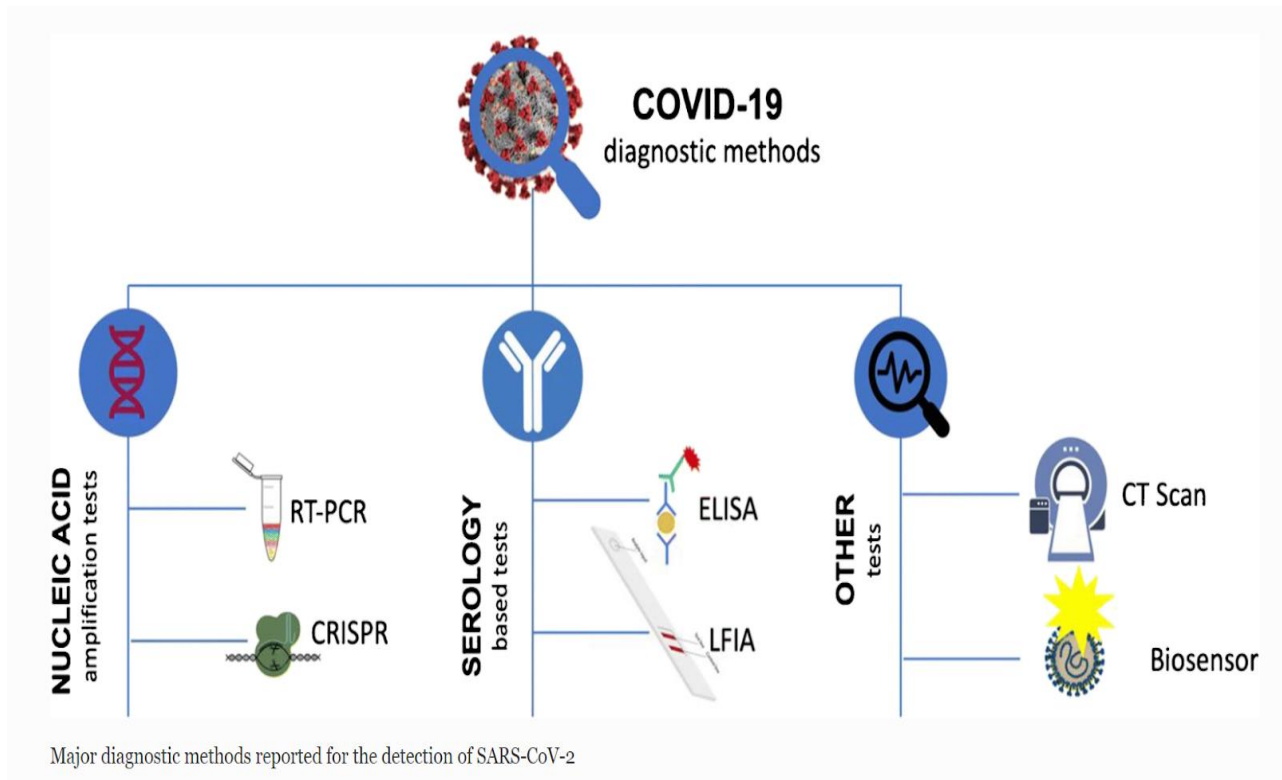


Fig. 1.2 - Major diagnostic methods reported for the detection of SARS-CoV-2

## 1.1 PROBLEM STATEMENT

Major methodologies used for detecting presence of COVID-19 in humans are the Serological tests and RT-PCR. While the Serological tests are more likely to report false-negative, research has found that RT-PCR did not detect 48 (36% [95% CI 28–44]) of 134 infected close contacts. And also the RT-PCR turnaround time depends on the testing situation and order received. The median turnaround times have been approximately 15-24 hours once the sample is received.

Detecting COVID-19 early may help in devising an appropriate treatment plan and disease containment decisions. Large-scale testing and contact tracing are central to efforts to control the pandemic. This heavily relies on diagnostic tests that can rapidly, accurately and reliably detect SARS-CoV-2. Testing plays a key role in our efforts to contain and mitigate the COVID-19 pandemic by identifying infected individuals to help prevent further person-to-person transmission of COVID-19. Under such circumstances, there is an urgent need to develop a smarter, faster and more effective detection model. Based on Artificial Intelligence (AI), the diagnosis of COVID-19 has become a more sensible solution when compared to other existing methods. Such algorithms can analyze the results of CT scan images in a shorter time than other existing methods, which can save more time and avoid direct contact.

## 1.2 OBJECTIVE

The main objective of our project is to aid early detection of COVID-19 by developing an AI tool to screen the presence of COVID-19 in the patient through analysing the Computed Tomography scan (CT scan) images. To address this Deep Learning model VGG-19 is used in this work to develop a classification algorithm which predicts whether the given input HRCT image belong COVID-19 patient or not. Computed tomography has substantially improved diagnostic performance over Chest X-rays in detecting COVID-19 and also gives faster and more accurate results compared to RT-PCR. The results can then be analysed by the medical professional for the treatment of patients which helps in early detection, isolation of infected patients which results in prevention of the further spread of virus and in providing accurate medical assistance required to the patient. This work aims to demonstrate a less hardware-intensive approach for COVID-19 detection with excellent performance that can be combined with medical equipment and help ease the examination procedure.

## 1.3 MOTIVATION

COVID-19's impact on sensitive populations, including the elderly, infants, and peoples with other comorbidities, is noticeable. The scale and the destructive potential of this virus has made it into a multidisciplinary issue. Experts from various fields such as epidemiology, the pharmaceutical industry, and diagnosis system modelling are working to combat the virus. This pandemic has exposed the inadequacy of public health systems worldwide. As the fog of confusion lifts and we begin to understand the rudiments of how the virus behaves, the end of the pandemic is nowhere in sight. The number of cases and the deaths continue to rise. The latter breached the 174 million mark, and it looks likely now that, in terms of severity, this pandemic will surpass the Asian Flu of 1957-58 and the Hong Kong Flu of 1968-69.

Moreover, a parallel problem may well exceed the direct death toll from the virus. We are referring to the growing economic crisis globally, and the prospect that these may hit emerging economies especially hard. The economic fall-out is not entirely the direct outcome of the COVID-19 pandemic but a result of how we have responded to it—what measures governments took and how ordinary people, workers, and firms reacted to the crisis. Therefore it is extremely important for timely detection and containment of the virus inorder to minimize the human fatalities and economic losses .

The increasing number of infected people poses a problem for the current screening procedures as they are overwhelmed by the number of tested people. Evaluation of the CT scan is time-

consuming and puts an extra load on the doctors involved in quarantining and treating the infected patients. To overcome this problem, the process of diagnosis through CT scans can be fastened through state-of-the-art (SOTA) computer vision and deep learning techniques. These algorithms can be used to evaluate COVID-19 as they can identify the lesions and the varied opacities at a faster rate than the existing methods.

## **1.4 SCOPE**

- The proposed work will be able to evaluate varying degrees of COVID-19 infection from CT-scans fuelled by techniques such as transfer learning, segmentation, and a robust preprocessing pipeline, which reduce the overall bias that may occur in datasets that are comparatively smaller in size.
- The results produced can also be extended to include the critical information about the spread of virus in the patient's lungs.
- CT would provide an automated “second reading” to clinicians, assisting in the diagnosis and criticality assessment of COVID-19 patients to assist in better decision making in the global fight against the disease.

## **1.5 SOFTWARE AND HARDWARE REQUIREMENTS**

### **1.5.1 Software Requirements**

#### **1.5.1.1 Programming Language and Coding Tools**

##### **1.5.1.1.1 PyCharm**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

##### **1.5.1.1.2 Python**

Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

### 1.5.1.2 Packages

#### 1.5.1.2.1 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

#### 1.5.1.2.3 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. Developers can build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

#### 1.5.1.2.4 Python Libraries

These are the basic libraries that transform Python from a general-purpose programming language into a powerful and robust tool for data analysis and visualization. They're the foundation that the more specialized tools are built on.

1. **NumPy** is the foundational library for scientific computing in Python, and many of the libraries on this list use NumPy arrays as their basic inputs and outputs. In short, NumPy introduces objects for multidimensional arrays and matrices, as well as routines that allow developers to perform advanced mathematical and statistical functions on those arrays with as little code as possible.
2. **SciKit-Learn** is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.
3. **Matplotlib** is a multi-platform data visualization library built on NumPy arrays. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

#### 1.5.1.2.5 Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot support—all designed to enhance developer productivity.

#### 1.5.1.2.6 Git and Github

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

### 1.5.2 Hardware Requirements

- Processer : Any Updated Processer (Itanium series)
- Ram : Min 4 GB
- Hard Disk : Min 50 GB

## 2. LITERATURE SURVEY

### 2.1 SURVEY OF MAJOR AREA RELEVANT TO PROJECT

Artificial Intelligence is modern and highly sophisticated technological applications became a huge trend in the industry. Artificial Intelligence is Omnipresent and is widely used in various applications. It is playing a vital role in many fields like finance, Medical science and in security. Artificial Intelligence is used to discover patterns from medical data sources and provide excellent capabilities to predict diseases.

Deep learning is contributing to the high level of services to the healthcare sector. As the digital medical data is increasing exponentially with time, early detection and prediction of diseases are becoming more efficient because of the deep learning techniques which reduce the fatality rate to a great extent. The state-of-the-art architectures effectively carry out analysis of 2D and 3D medical images to make the diagnosis of patients faster and more accurate. The use of popular approaches in machine learning such as ensemble and transfer learning with fine-tuning of parameters improve the performance of the deep neural networks in medical image analysis.[1]

Recently, several deep learning methods have been applied on chest x-ray images and CT scans for COVID-19 diagnosis as well as segmentation of lung infections.[2] Artificial Intelligence (AI)-based chest CT analysis is playing an important role in fighting the COVID-19 pandemic.

There were relevant studies on the AI algorithm that was applied to the detection of COVID-19. From multiple sources, Qiu Hu established a comprehensive data set of X-ray and CT scan images, and adopted deep learning and transferred learning algorithms to provide a simple and effective COVID-19 detection technology. A dual sampling attention network was proposed to classify the different manifestations between COVID-19 and community-acquired pneumonia (CAP) infections. In order to focus on the lungs, a lung mask was applied to suppress the image context of non-lung regions in chest CT. At the same time, the online mechanism increased the focus on the deep learning model for the better focus on the lung infectious area. In this way, the model helps explain the evidence for the automatic diagnosis of COVID-19. The experimental results also proved that the proposed online attention refinement can effectively improve the classification performance. A quantitative system based on deep learning can automatically carry out CT segmentation of the chest CT infectious area

and the entire lung, and establish a VNet neural network, which can segment the COVID-19 infectious area in the CT scan image. In order to accelerate the speed of data labeling, human-in-the-loop optimization was adopted to label each situation.[3]

In [4], they have developed a deep learning-based Convolutional Neural Network (CNN) model to classify COVID-19 cases from healthy cases. They have collected 10979 chest CT images of 131 COVID-19 positive patients, with the infection masks segmented by a radiologist, and 150 healthy subjects. For network training, 1936 CT slices involved by infection along with 1735 slices from healthy subjects, with slice locations similar to those of the COVID-19 positive group were used, to eliminate non-infection differences between the groups and make the model more generalized, reliable, and focused on the infected regions. The preprocessing steps consisted of transforming image intensities into the Hounsfield unit, extracting the lung part by image processing techniques, equalizing histograms with a transformation created by the intensities of the infection regions, and stretching the contrast of the images. Then, after training and evaluating several deep learning networks, such as ResNet, Inception, VGG16, DenseNet, and Xception by the images of 80% of the cases in each group, VGG16 was chosen base model, since it is less complicated compared to the other models. The two-class output was obtained using pooling, dropout, and dense layers. Finally, model was tested by the CT images of 20% of the cases in each group. The results showed 89.78% precision, 91.50% sensitivity, 88.66% specificity, 0.9063 F1-Score, and 90.14% accuracy.

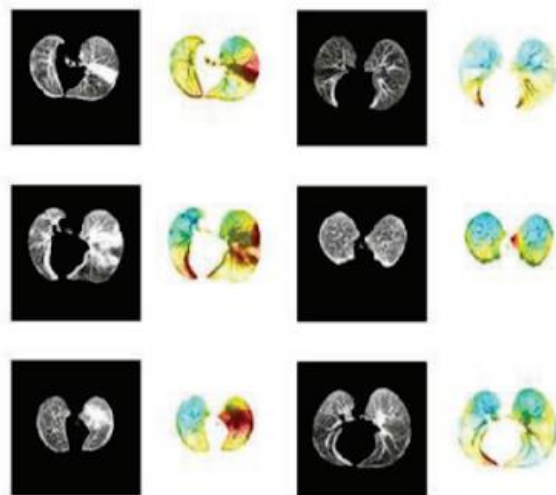


Fig. 2.1 - COVID-19 lung images (left) and normal lung images (right) with their heatmaps

The AI-based analysis of chest CT for probable COVID-19 prevalence involves multiple steps from image acquisition, image pre-processing, segmentation, and final diagnosis. However, the recent deep learning-based approaches require labeled datasets to train models. Since the labeling process of CT scans requires expert knowledge (mainly from a radiologist) and a significant amount of time, most of the supervised learning-based models are trained on a limited amount of data.[5]

An alternative method of training deep learning models is “transfer learning” whereby a deep learning network is pre weighted with the results of a previous training cycle from a different domain. This technique is commonly used as a basis for initializing deep learning models which are then fine tuned using the limited available medical sample data set with results that have been documented to outperform fully trained networks under certain circumstances.[6]

## **2.2 TECHNIQUES AND ALGORITHMS RELEVANT WITH RESPECT TO PROJECT**

### **2.2.1 Transfer Learning**

Transfer learning as a new machine learning paradigm has gained increasing attention lately. In situations where the training data in a target domain are not sufficient to learn predictive models effectively, transfer learning leverages auxiliary source data from other related source domains for learning.

Transfer learning is a method that utilizes the knowledge acquired by a CNN from a specific problem to solve a distinct but similar task. This transferred knowledge is used in a new dataset, whose size is usually smaller than the adequate size to train a CNN from scratch. In deep learning, this method requires an initial training of a CNN for a given task, using large datasets. The availability of a sizable dataset is the main factor to ensure the success of the method since CNN can learn to extract the most significant features of a sample.

The CNN is deemed suitable for transfer learning if it is found to be able to extract the most important image features. Then, in the transfer learning, the CNN is used to analyze a new dataset of a different nature and extract its features according to the knowledge acquired in the first training. One common strategy to exploit the capabilities of the pre-trained CNN is called feature extraction via “transfer learning”. This approach means that the CNN will retain its



architecture and weights between its layers; therefore, the CNN is used only as a feature extractor. The features are later used in a second network/classifier that will process its classification.

The transfer learning approach is mostly used to work around computational costs of training a network from scratch or to keep the feature extractor trained during the first task. In medical applications, the most accepted practice of transfer learning is to utilize the CNNs that achieved the best results in the ImageNet large scale visual recognition challenge (ILSVRC), which assesses algorithms for object detection and classification in large scales. The use of large datasets for initial training of the network enables high performance in smaller datasets. This performance is linked to various extraction parameters that are typically not allowed as they cause overfitting of the network.

Feature extraction performed with transfer learning allows a large number of features to be extracted by generalizing the problem and avoiding excessive adjustments. The use of transfer learning also allows the use of the internet of things (IoT) systems to classify medical images.

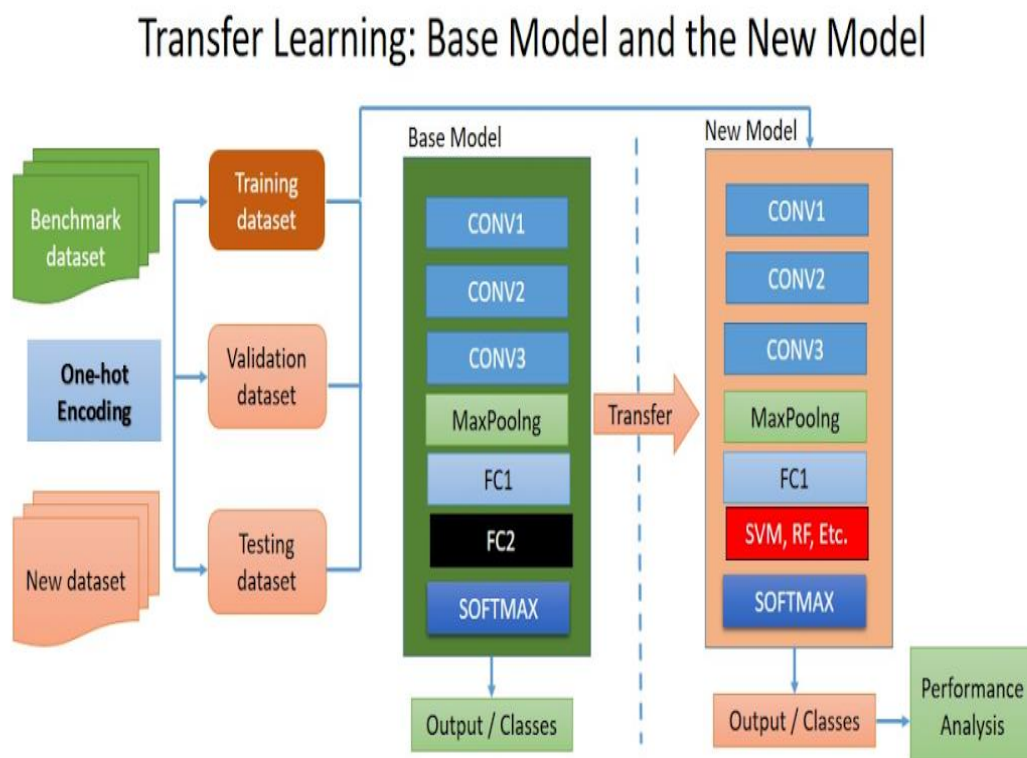


Fig. 2.2 - Transfer Learning Strategy

### 2.2.2 Transfer Learning Approach

One can use transfer learning on your own predictive modelling problems. Two common approaches are as follows:

1. Develop Model Approach
2. Pre-trained Model Approach

Our work uses the second approach which is commonly used in deep learning. It is as follows:

#### 1. Select Source Model:

A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

#### 2. Reuse Model:

The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

#### 3. Tune Model:

Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

### 2.2.3 Transfer Learning with Image Data

It is common to perform transfer learning with predictive modeling problems that use image data as input. This may be a prediction task that takes photographs or video data as input. For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.

The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware. These models can be downloaded and incorporated directly into new models that expect image data as input.

Three main examples of models of this type include:

- Oxford VGG Model
- Google Inception Model
- Microsoft ResNet Model

Our work uses the VGG model which has different variants.

VGG models are “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION” proposed by Karen Simonyan & Andrew Zisserman of Visual Geometry Group (VGG), Oxford University, which brought remarkable results for the ImageNet Challenge.

They experiment with 6 models, with different numbers of trainable layers. Based on the number of models, the two most popular models are VGG16 and VGG19.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 2.3 - Comparison of different VGG variants

### • Selection Of Transfer Learning Model

There are perhaps a dozen or more top-performing models for image recognition that can be downloaded and used as the basis for image recognition and related computer vision tasks. After going through a lot of articles and peer reviewed research papers, we have found even though both VGG-16 and VGG-19 converged well for binary medical image classification, in VGG-16 overfitting was evident from 5 epochs and in VGG-19 overfitting was evident from 20 epochs, which proves that VGG-19 would give us better performance. Hence, we have chosen VGG-19 as our base pre-trained model.

## 2.2.4 VGG-19

### 2.2.4.1 Architecture

The input to VGG based convNet is a  $224 \times 224$  RGB image. Preprocessing layer takes the RGB image with pixel values in the range of 0–255 and subtracts the mean image values which is calculated over the entire ImageNet training data set from the ImageNet competition.

The input images after preprocessing are passed through these weight layers. The training images are passed through a stack of convolution layers.

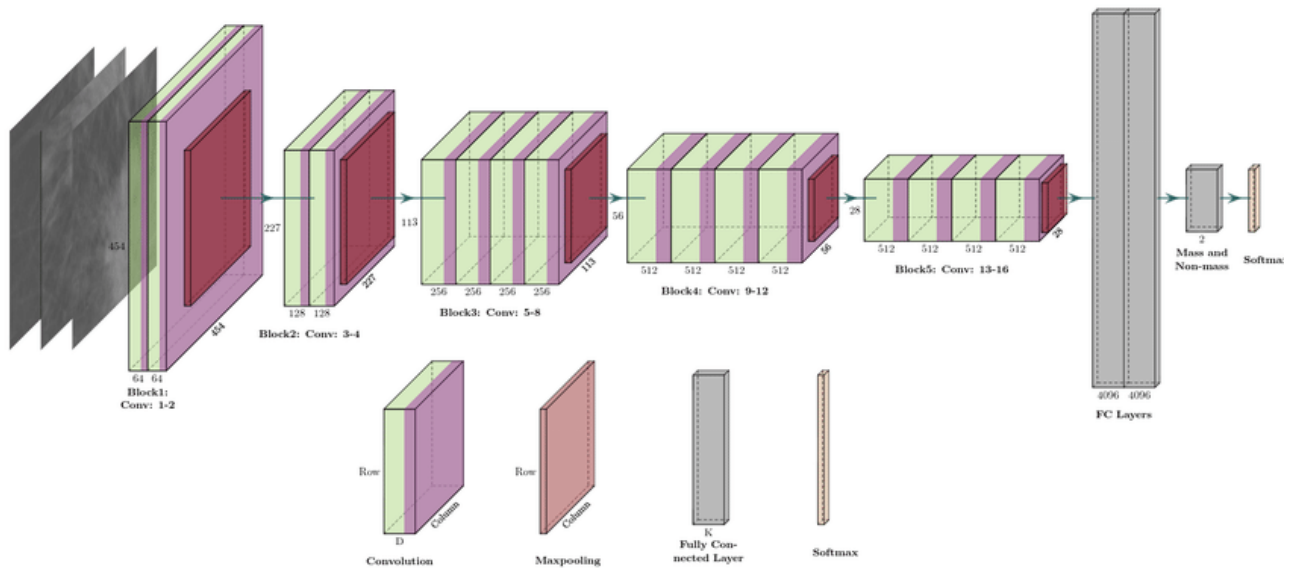


Fig. 2.4 - Details of the 19 layers of VGG19 network 21 used for feature extraction

VGG-19 has 19 weight layers consisting of 16 convolutional layers with 3 fully connected layers and 5 pooling layers. It consists of two Fully Connected layers with 4096 channels each which is followed by another fully connected layer with 1000 channels to predict 1000 labels. Last fully connected layer uses the softmax layer for classification purposes.

### 2.2.4.2 Architecture Walkthrough

Layer name	#Filters	#Parameters	#Activations
input			150K
conv1_1	64	1.7K	3.2M
conv1_2	64	36K	3.2M
max pooling			802K
conv2_1	128	73K	1.6M
conv2_2	128	147K	1.6M
max pooling			401K
conv3_1	256	300K	802K
conv3_2	256	600K	802K
conv3_3	256	600K	802K
conv3_4	256	600K	802K
max pooling			200K
conv4_1	512	1.1M	401K
conv4_2	512	2.3M	401K
conv4_3	512	2.3M	401K
conv4_4	512	2.3M	401K
max pooling			100K
conv5_1	512	2.3M	100K
conv5_2	512	2.3M	100K
conv5_3	512	2.3M	100K
conv5_4	512	2.3M	100K
max pooling			25K
fc6		103M	4K
fc7		17M	4K
output		4M	1K

Fig. 2.5 - Details on the VGG19 architecture. For each layer, number of filters, parameters and activations

- The first two layers are convolutional layers with 3\*3 filters, and first two layers use 64 filters that results in 224\*224\*64 volumes with the same convolutions are used. The filters are always 3\*3 with stride of 1
- After this, pooling layer was used with max-pool of 2\*2 size and stride 2 which reduces height and width of a volume from 224\*224\*64 to 112\*112\*64.
- This is followed by 2 more convolution layers with 128 filters. This results in the new dimension of 112\*112\*128.
- After pooling layer is used, volume is reduced to 56\*56\*128.
- Two more convolution layers are added with 256 filters each followed by down sampling layer that reduces the size to 28\*28\*256.
- Two more stacks each with 3 convolution layers are separated by a max-pool layer.
- After the final pooling layer, 7\*7\*512 volume is flattened into Fully Connected (FC) layer with 4096 channels and softmax output of 1000 classes. The FC layers are used for classification.

- In all layers except the last one, ReLU activation function is used, while in the last one Softmax is used for probability distribution between classes.

### 2.2.4.3 Layers in VGGNet

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

#### Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size  $M \times M$ . By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ( $M \times M$ ). The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

Table-2.1 Detailed VGG-19 Architecture

Layer Type	Feature Map	Size	Kernel Size	Stride	Activation
Image	1	224×224	—	—	—
Convolution	64	224×224	3×3	1	ReLU
Convolution	64	224×224	3×3	1	ReLU
Max Pooling	64	112×112	2×2	2	—
Convolution	128	112×112	3×3	1	ReLU
Convolution	128	112×112	3×3	1	ReLU

Max Pooling	128	56×56	2×2	2	–
Convolution	256	56×56	3×3	1	ReLU
Convolution	256	56×56	3×3	1	ReLU
Convolution	256	56×56	3×3	1	ReLU
Convolution	256	56×56	3×3	1	ReLU
Max Pooling	256	28×28	2×2	2	–
Convolution	512	28×28	3×3	1	ReLU
Convolution	512	28×28	3×3	1	ReLU
Convolution	512	28×28	3×3	1	ReLU
Convolution	512	28×28	3×3	1	ReLU
Max Pooling	512	14×14	2×2	2	–
Convolution	512	14×14	3×3	1	ReLU
Convolution	512	14×14	3×3	1	ReLU
Convolution	512	14×14	3×3	1	ReLU
Convolution	512	14×14	3×3	1	ReLU
Max Pooling	512	7×7	2×2	2	–

Fully Connected	—	4096	—	—	ReLU
Fully Connected	—	4096	—	—	ReLU
Fully Connected	—	1000	—	—	Softmax

### Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon the method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from the feature map. Average Pooling calculates the average of the elements in a predefined size Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

### Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

### 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 a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.



## Activation functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network

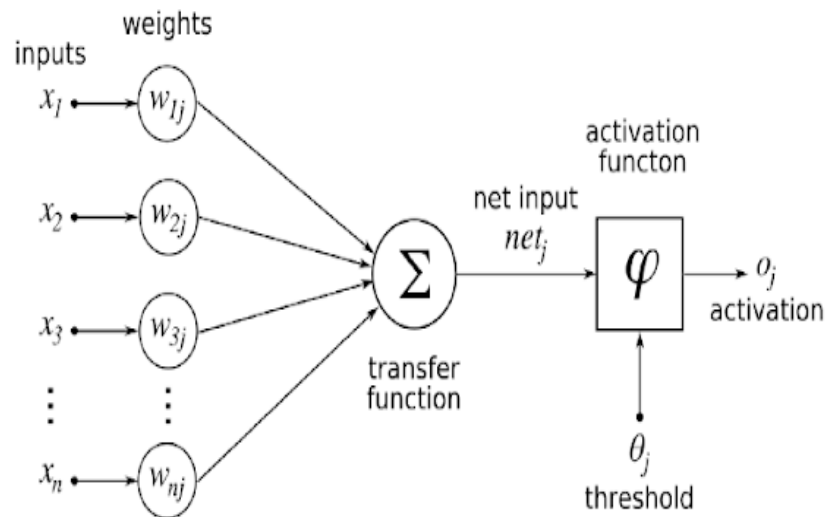


Fig. 2.6 - Activation Function

There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and SoftMax functions are preferred and for a multi-class classification, generally SoftMax is used. VGG-19 uses SoftMax Activation function which is described below.

- **SoftMax**

The ImageNet dataset contains images of fixed size of  $224 \times 224$  and have RGB channels. So, we have a tensor of  $(224, 224, 3)$  as our input. This model processes the input image and outputs a vector of 1000 values.(I)

$$\hat{y} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hat{y}_{999} \end{bmatrix}$$

This vector represents the classification probability for the corresponding class. Suppose we have a model that predicts that image belongs to class 0 with probability .1, class 1 with probability 0.05, class 2 with probability 0.05, class 3 with probability 0.03, class 780 with probability 0.72, class 999 with probability 0.05 and all other class with 0. so, the classification vector for this will be: (II)

$$\hat{y} = \begin{bmatrix} \hat{y}_0 = 0.1 \\ 0.05 \\ 0.05 \\ 0.03 \\ \vdots \\ \vdots \\ \hat{y}_{780} = 0.72 \\ \vdots \\ \vdots \\ \hat{y}_{999} = 0.05 \end{bmatrix}$$

To make sure these probabilities add to 1, we use the softmax function. This softmax function is defined as : (III)

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(j)}}}{\sum_{k=0}^k e^{\Theta_k^{(j)}}}$$

softmax function

where  $\Theta = w_0 x_0 + w_1 x_1 + \dots + w_k x_k = \sum_{i=0}^k w_i x_i = w^T x$

## 2.3 APPLICATIONS

- Enhancing Automation in COVID-19 Screening process:**

**“Better testing makes us better equipped to defeat COVID-19”.[7]**

Rapid patient testing is an essential part of the nation’s strategy to combat the COVID-19 crisis. Automation of the Screening process will minimize human interaction. By automating processes and generation of reports more quickly, it will enable to deliver patient results at a pace so that physicians can more rapidly treat patients which will ultimately help in the management of COVID-19.

- **Reusability of Model:**

The developed model can be reused to build several other models on top of it using segmentation for detecting families of such viral diseases, therefore enabling reusability of code. The project can also be extended to generate critical information about the spread virus in the patient's lungs i.e predicting some important factors like CO-RADS\* level.

<b>CO-RADS*</b>		
	<b>Chance of COVID-19</b>	<b>CT findings</b>
<b>CO-RADS 1</b>	Highly unlikely	normal or non-infectious abnormalities
<b>CO-RADS 2</b>	Unlikely	abnormalities consistent with infections other than COVID-19
<b>CO-RADS 3</b>	Equivocal	unclear whether COVID-19 is present
<b>CO-RADS 4</b>	Probable	abnormalities suspicious for COVID-19
<b>CO-RADS 5</b>	Highly likely	typical COVID-19
<b>CO-RADS 6</b>	PCR proven	

Fig. 2.7- CO-RADS\* Levels

- **AI Radiology:**

Currently, we are on the brink of a new era in radiology artificial intelligence. AI has had a strong focus on image analysis for a long time and has been showing promising results. Therefore, there are great expectations around applying AI to radiological images.

The discovery of X-rays, Positron emission tomography (PET), computed tomography (CT), and MRI, which on its own includes a long list of different imaging options enabling physicians to measure not only different structures but also function of the body.

### 3. SYSTEM DESIGN

#### 3.1 SYSTEM ARCHITECTURE

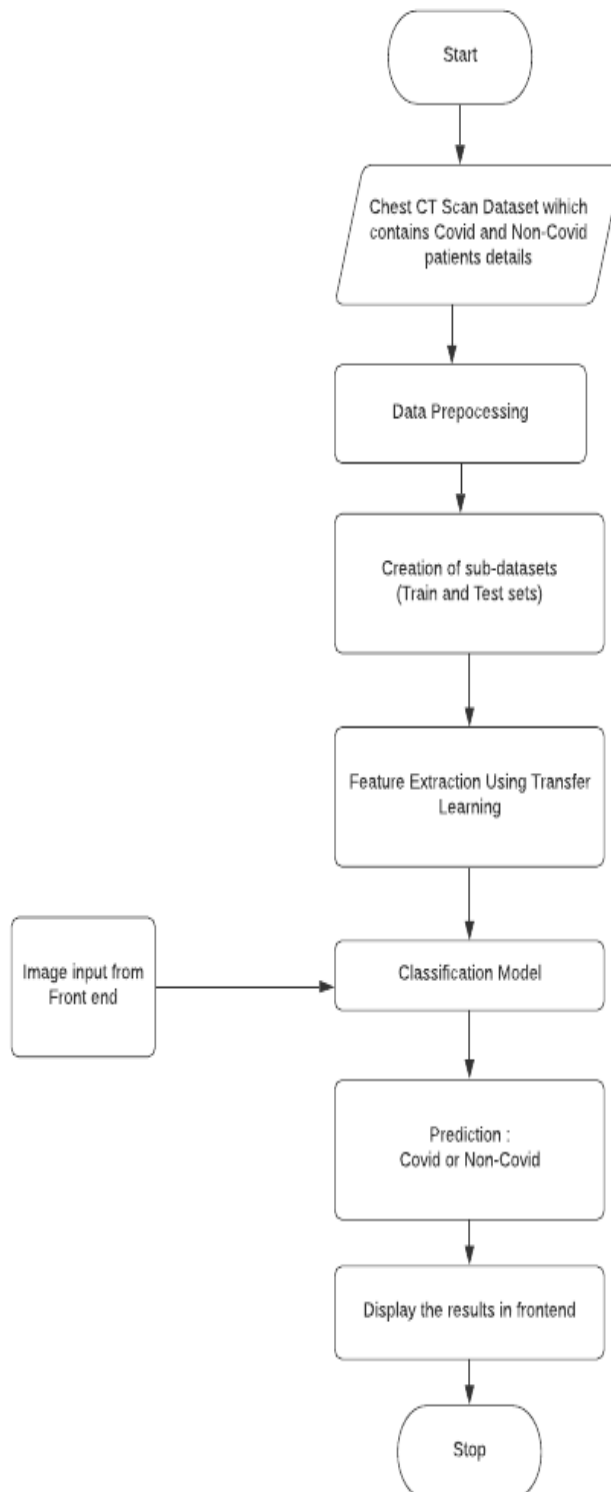


Fig. 3.1 - Flow Chart of the system

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

The above Figure 3.1 represents flow chart of our system where the input CT Scan images of COVID positive and negative patients will undergo various modifications in the Data Preprocessing phase. After essential modifications the images will be in a format that can be accepted by the model. Next comes the Training and Testing phases, where initially the model is split into Train and Test sub datasets. The pre-trained VGG-19 model will be modified and trained on the current input images to get the binary classification outputs. The trained model is then tested against the test dataset and the model performance is determined through various evaluation metrics. Finally the model is deployed in the Python Flask web application to predict outputs in the front end. The detailed system architecture is shown in the figure below:

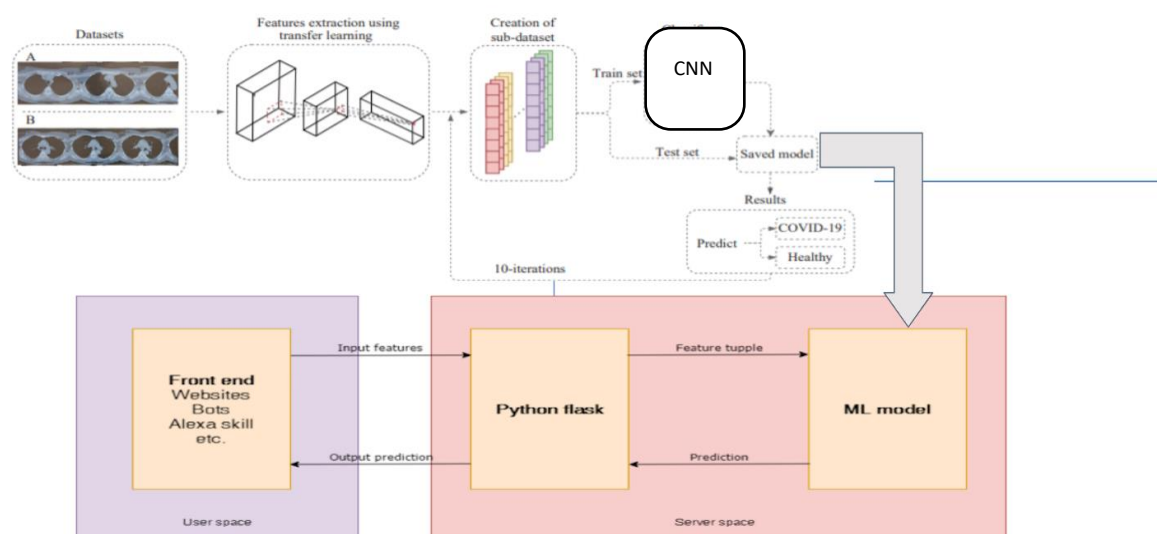


Fig. 3.2 - Detailed System Architecture

## 3.2 MODULES

### 3.2.1 Module-1 : Train and Test Module

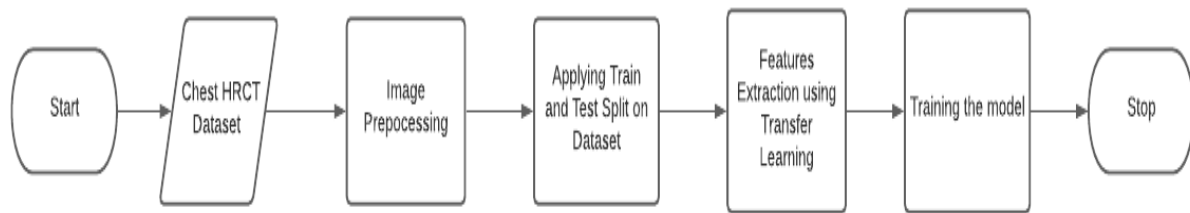


Fig. 3.3 - Flow Chart for Module-1

Our project is divided into two modules and the first module is the Training Module. In this module the Chest HRCT image dataset is modified using various data preprocessing techniques in order to get input according to the model requirements. The output is divided into train and test sets. The train dataset is trained using the pre-trained VGG19 model using transfer learning. Next it is tested against the test dataset and model performance will be evaluated. The final output will be a trained model to predict our outputs.

- Input : Chest HRCT Image
- Output : Presence of COVID-19 Prediction model

### 3.2.2 Module-2 : Deployment Module

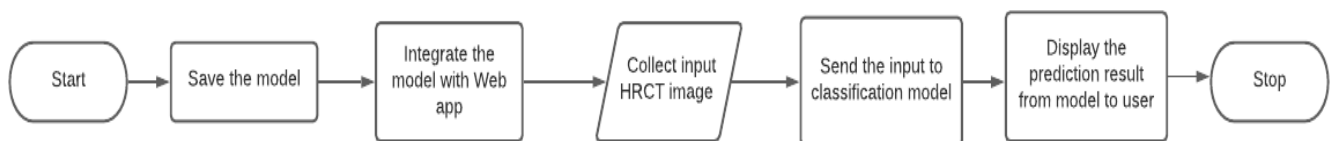


Fig. 3.4 – Flow Chart for Module-2

The output of Module-1, which is the Presence of COVID-19 prediction model, is taken as input in this module. The model is saved to the local system. Later, this model is integrated with the web application using Python Flask framework. Finally, the web app is deployed on the Heroku cloud using GitHub, where we can see the predictions of the model on the web app.

- Input : Presence of COVID-19 prediction model
- Output : Web app

## 4. IMPLEMENTATION

The overall implementation process is condensed into following steps:

- 1.Data Preprocessing and Normalization of input HRCT image dataset [4.2]
- 2.Selection of Transfer Learning Model [2.2.3]
- 3.Building the VGG-19 model to use Transfer Learning and extract the input image features [2.2.4][4.2]
- 4.Training and Testing the model [4.2]
- 5.Deploying the model into Heroku Cloud [4.3]

### 4.1 ENVIRONMENTAL SETUP

#### 4.1.1 PyCharm

To develop our project, we have used Pycharm. PyCharm is an integrated development environment (IDE) used in computer programming specifically for the Python language. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science.

##### 4.1.1.1 Features provided by Pycharm:

- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others.
- Support for web frameworks: Django, web2py and Flask.
- Version Control Integration: unified user Interface for Mercurial, Git, Subversion, Perforce and CVS with
- Support for scientific tools like matplotlib, numpy and scipy.

##### 4.1.1.2 Steps to Install Pycharm

- To download PyCharm visit the website :  
<https://www.jetbrains.com/pycharm/download/>
- Click the "DOWNLOAD" link under the Community Section.
- Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
- On the next screen, Change the installation path if required. Click “Next”.
- On the next screen, you can create a desktop shortcut if you want and click on “Next”.
- Choose the start menu folder. Keep selecting JetBrains and click on “Install”. Wait for the installation to finish.

- Once installation is finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.

#### 4.1.2 Git for PyCharm

When you clone an existing Git repository, or put an existing project under Git version control, PyCharm automatically detects if Git is installed on your computer. If the IDE can't locate a Git executable, it suggests downloading it.

##### 4.1.2.1 Commit and push changes to Git repository:

Configure commit options: Settings/Preferences | Version Control | Commit

Commit tool window Alt+0

Commit Ctrl+K

Commit and Push Ctrl+Alt+K

Push Ctrl+Shift+K

After you've added new files to the Git repository, or modified files that are already under Git version control and you are happy with their current state, you can share the results of your work. This involves committing them locally to record the snapshot of your repository to the project history, and then pushing them to the remote repository so that they become available to others.

##### 4.1.2.2 Set your Git username:

Git needs to know the username to associate commits with an identity. If the username is not set, PyCharm will prompt to specify it when you first attempt to commit changes.

- Open the Terminal and execute one of the following commands:
- To set a name for every Git repository on your machine, use `$ git config --global user.name "John Smith"`.
- To set a name for a single repository, use `$ git config user.name "John Smith"`

#### 4.1.2 Git

Git is a DevOps tool for source code management—an open-source version control system (VCS) used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, supporting non-linear development so that multiple developers can work together in near real-time.



### 4.1.3 Heroku

#### 4.1.3.1 Heroku Setup

Before you can start using [Heroku](#), you need to sign up for a free account. The [signup form](#) is straightforward and takes less than a minute to fill. When you join Heroku, you get free dyno hours that you can use for running your free apps..

Next, you need to install Heroku Command Line Interface (CLI). CLI is the most efficient way to manage your apps. If you are a complete beginner to app development and aren't comfortable with a command line interface, you can use Heroku's dashboard for app management.

In order to install Heroku CLI, you first need to [install Git](#). Once you have Git on your system, you should configure it before you start deploying apps to Heroku. [Configuring Git](#) allows you to set your user identity and set the default text editor, which is important to avoid running into unexpected errors.

To install Heroku CLI on MacOS, type the following in your command line:

```
brew tap heroku/brew && brew install heroku
```

For Windows, download the [32-bit](#) or the [64-bit installer](#), according to the machine you have. Here are [other methods to install Heroku CLI](#) on your system.

Now that you have the prerequisites, you are ready to start deploying your mobile app to Heroku.

#### 4.1.3.2 Using Heroku with Python

##### 1) Prepare The App

First, create a local version of your app in order to set up a Git repository. You will need a Git repository to create a remote Heroku-hosted repository, where your code will go live. To create a local clone of your code, run the following commands:

```
$ git clone https://github.com/heroku/python-getting-started.git  
$ cd python-getting-started
```

In this example, *python-getting-started* is the name of the sample code. Replace it with the exact name of your code that you want to deploy. Additionally, in the first command, *heroku* is the name of the parent directory where the code lives. Replace it with the name of the directory where you have saved your code.

Now that you have cloned your source code, it is time to deploy your app.

##### 2) Create a Heroku Remote

In order to deploy your app to [Heroku](#), you first need to create a Heroku-hosted Git remote repository. This is a version of your local Git repository that lives on other servers. From your app's root directory, run the following CLI command:

```
$ heroku create
```

This creates an empty Heroku Git repository for your app. If you have different versions of your app, such as production and staging, it is a good idea to name the Heroku Git repositories accordingly. To rename a Heroku Git repository, run the following command:

```
$ git remote rename [original name of repository] [new name of repository]
```

For our example, replace [original name of repository] with *heroku* and replace [new name of repository] with something that fits your project.

### **3) Deploy Your App**

Once you have created a remote git, it is time to deploy your code. You do it with the git push command, as follows:

```
$ git push heroku master
```

You need to push the code to the master branch of your remote repository in order to see changes. If you push the changes to any other branch, the changes will not take effect.

Now, you can visit your app on the browser with the following command:

```
$ heroku open
```

### **4) Declare App Dependencies**

To tell Heroku that it is a Python app, you need to install key files. Make a requirements.txt file that lists all the dependencies together. To install the text file, run the following command:

```
$ pip install - r requirements.txt
```

When you run this command, Heroku installs Python dependencies locally. Make sure you have Postgres installed correctly, though, for this step to function properly. Once you have declared the dependencies, you can run your app locally.

### **5) Adding a Database**

Heroku has a large number of data stores in its marketplace. It provides managed data services for Postgres, Redis, and Apache Kafka. By default, Heroku adds a free Postgres Starter Tier database when you deploy your app.

## 4.2 IMPLEMENTATION OF EACH MODULE

The process to build a deep learning model is shown in the picture below:

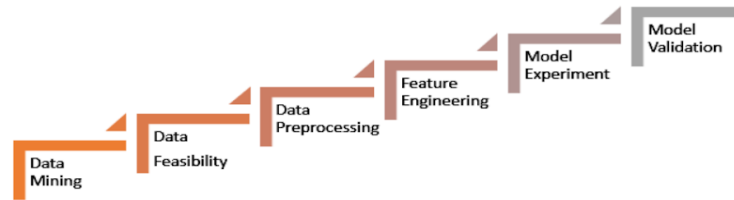


Fig. 4.1 - Steps in developing deep learning model

Due to the pandemic situation, obtaining raw data was not feasible. We have utilised online repositories like Kaggle to collect as much data as possible. Our input dataset consists of 63,849 images of both COVID and Healthy persons. The following are the steps for implementation:

### 1. Data Preprocessing

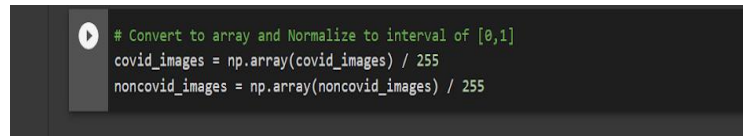
Due to the pandemic situation, we couldn't get raw data from any clinic. So, we have utilised online repositories like Kaggle to collect as much data as possible. The input data set consists of images in 16-bit grey scale TIFF format. But the selected VGG-19 model requires images to be in RGB format with size of 224 X 224 pixels. We have utilized Keras library functions to perform necessary changes to the input data set.

#### • Major Functions Used In This Phase

- *cv2.imread(filepath,flag)* - This method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.
- *cv2.cvtColor()* - This method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV.
- *cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])* - To resize an image, OpenCV provides *cv2.resize()* function.

## 2. Normalization

The VGG-19 model takes images in the form of an array of pixels. Hence we have converted the images into array and normalized them using Numpy library functions.

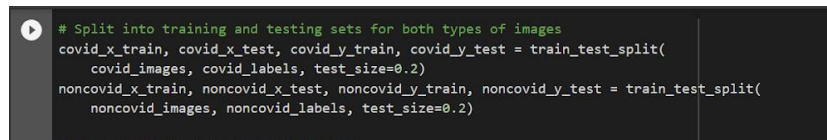


```
# Convert to array and Normalize to interval of [0,1]
covid_images = np.array(covid_images) / 255
noncovid_images = np.array(noncovid_images) / 255
```

Fig. 4.2 – Code snippet for Normalization

## 3. Train And Test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. We have splitted our dataset in 80:20 ratio where training data is of 80% with around 50746 images and test data is of 20% with around 12686 images.



```
# Split into training and testing sets for both types of images
covid_x_train, covid_x_test, covid_y_train, covid_y_test = train_test_split(
    covid_images, covid_labels, test_size=0.2)
noncovid_x_train, noncovid_x_test, noncovid_y_train, noncovid_y_test = train_test_split(
    noncovid_images, noncovid_labels, test_size=0.2)
```

Fig. 4.3 – Code snippet for Train and Test Split

## 4. Building The Model

The VGG-19 model is trained on ImageNet dataset over 14 million images, to classify those images into 1000 classes. It has a total of 138 million parameters. But our requirement is to train 50746 to classify them into two classes, COVID and Healthy. To achieve this we have modified the last Fully Connected layers of the model which are right below the last MaxPooling layer which has 512 Feature maps with images of size 7 X 7 pixels, based on the input received from the Input layer and used the ‘Adam’ optimizer, also added a Dropout layer to boost the learning rate. The following image gives details about the features that are expected by the model to be trained on:

**CT findings** GGO - consolidation - distribution  
 Crazy paving  
 (Reversed) halo - spider web sign  
 Vascular thickening  
 Pleural fluid  
 Enlarged lymph nodes etc

Fig. 4.4 - Features to be extracted from input images

We have added the following layers on the top of our base model

- flatten (Flatten) : This layer converts the 3 dimensional inputs to 1 dimensional.
- dropout (Dropout) : This layer is added to make sure of the overfitting.
- dense (Dense) : A fully connected layer that uses 'softmax' as an activation function to give the output which classifies the input image as Covid positive or Covid Negative.

As part of our project we have also studied the usage of different activation functions by applying Softmax, ReLU and Sigmoid activation functions respectively in the last Fully Connected Layer of VGG-19 model. From our observations, we have concluded that the VGG-19 model with SoftMax activation function will give better outputs and we have proceeded our work using this model. The summary of this model is given below.

- Summary Of The VGG-19 Model With Softmax Activation Function**

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0

dropout (Dropout)	(None, 25088)	0
dense (Dense)	(None, 2)	50178
=====		
=====		
Total params: 20,074,562		
Trainable params: 50,178		
Non-trainable params: 20,024,384		

- **Major Functions Used In This Phase**

- *tf.keras.applications.VGG19(include\_top=True,weights='imagenet',input\_tensor=None,input\_shape=None,pooling=None,classes=1000,classifier\_activation='softmax')* - is VGG19 model for Keras.
- *model.compile()* - Compile function is used here that involves use of loss, optimizers and metrics.

## 5. Image Augmentation

We have used the “ImageDataGenerator” function from keras library to generate augmented images from input dataset so as to train the model on images at different positions, flips, angles,etc.

- **Major Functions Used In This Phase**

- *ImageDataGenerator()* - ImageDataGenerator that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This ImageDataGenerator includes all possible orientations of the image.

## 6. Training The Model

Based on our research on different deep learning image classifiers, we have found that the suitable number of epochs to train this model is 500. Following are our model parameters:

- Input Image size - 224 X 224 pixels
- Color mode - RGB
- Epochs - 500
- Batch-size – 32

- **Saving The Trained Model**

Training a neural network/deep learning model usually takes a lot of time, particularly if the hardware capacity of the system doesn't match up to the requirement. Once the training is done, we save the model to a file. To reuse the model at a later point of time to make predictions, we load the saved model.

Through Keras, models can be saved in three formats

- YAML format
- JSON format
- HDF5 format

YAML and JSON files store only model structure, whereas, HDF5 file stores complete neural network model along with structure and weights. Therefore, if the model structure is saved using YAML or JSON format, weights should be stored in an HDF5 file to store the entire model.

- **Major Functions Used In This Phase**

- `model.save()` - Saves a model as a TensorFlow SavedModel or HDF5 file.
- `model.save_weights()` - Saves the model weights as a HDF5 file.

## 7. Testing The Model

After training the model, we have tested the model against the test dataset and evaluated its performance using various evaluation metrics which will be explained in next sections of this document.

The next steps in implementation are integration and deployment which are explained below.

## 4.3 INTEGRATION AND DEPLOYMENT

After developing our prediction model, the next step is to integrate the model with web application and deploy it in Heroku cloud.

### 4.3.1 Integration

As mentioned in the above section, the “save” function in Keras helps us to save the trained prediction model into our local system. Similarly, “load\_model” function in Keras helps to load the model from the local system to any work file.



The deep learning model is initially saved into the local system and then loaded into a web application by creating a REST API using Python Flask Web Framework. The following are the steps to create Keras REST API using Flask:

## 1. Import Libraries

The following are the three important libraries required for every Keras REST API:

- *load\_model*: Used to load our trained Keras model and prepare it for inference.
- *prepare\_image*: This function preprocesses an input image prior to passing it through our network for prediction. If you are not working with image data you may want to consider changing the name to a more generic *prepare\_datapoint* and applying any scaling/normalization you may need.
- *predict*: The actual endpoint of our API that will classify the incoming data from the request and return the results to the client.

Along with the above libraries, we have to import necessary modules for data preprocessing in order to satisfy the model requirements.

## 2. Load The Model

First we call *load\_model* which loads our Keras model from disk.

The call to *load\_model* is a blocking operation and prevents the web service from starting until the model is fully loaded. Had we not ensured the model is fully loaded into memory and ready for inference prior to starting the web service we could run into a situation where:

- A. A request is POST'ed to the server.
- B. The server accepts the request, preprocesses the data, and then attempts to pass it into the model
- C. But since the model isn't fully loaded yet, our script will error out!

## 3. Data preprocessing

Before we can perform prediction on any data coming from our client we first need to prepare and preprocess the data.

Sample code:

```
def prepare_image(image, target):
    # if the image mode is not RGB, convert it
    if image.mode != "RGB":
        image = image.convert("RGB")

    # resize the input image and preprocess it
    image = image.resize(target)
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)
    image = imagenet_utils.preprocess_input(image)

    # return the processed image
    return image
```

The above function:

- Accepts an input image
- Converts the mode to RGB (if necessary)
- Resizes it to 224x224 pixels (the input spatial dimensions for ResNet/VGGNet)
- Preprocesses the array via mean subtraction and scaling

Again, you should modify this function based on any preprocessing, scaling, and/or normalization you need prior to passing the input data through the model.

#### 4. Predictions

We are now ready to define the predict function — this method processes any requests to the /predict endpoint:

Sample Code:

```
@app.route("/predict", methods=["POST"])
def predict():
    # initialize the data dictionary that will be returned from the
    # view
    data = {"success": False}
```

```

# ensure an image was properly uploaded to our endpoint
if flask.request.method == "POST":
    if flask.request.files.get("image"):
        # read the image in PIL format
        image = flask.request.files["image"].read()
        image = Image.open(io.BytesIO(image))

        # preprocess the image and prepare it for classification
        image = prepare_image(image, target=(224, 224))

        # classify the input image and then initialize the list
        # of predictions to return to the client
        preds = model.predict(image)
        results = imagenet_utils.decode_predictions(preds)
        data["predictions"] = []

        # loop over the results and add them to the list of
        # returned predictions
        for (imagenetID, label, prob) in results[0]:
            r = {"label": label, "probability": float(prob)}
            data["predictions"].append(r)

        # indicate that the request was a success
        data["success"] = True

    # return the data dictionary as a JSON response
    return flask.jsonify(data)

```

The data dictionary is used to store any data that we want to return to the client. The above code includes a boolean used to indicate if prediction was successful or not — we can also use this dictionary to store the results of any predictions we make on the incoming data.

## 5. Launch the service

Sample code:

```
# if this is the main thread of execution first load the model and
# then start the server
if __name__ == "__main__":
    print(("* Loading Keras model and Flask starting server..."
          "please wait until server has fully started"))
    load_model()
    app.run()
```

- Summary
  1. Read it in PIL format
  2. Preprocess it
  3. Pass it through our network
  4. Loop over the results and add them individually to the data["predictions"] list
  5. Return the response to the client in JSON format

### 4.3.2 Heroku setup for deploying Flask app using Pycharm

#### 1. Initial setup

First, register a new account on [www.heroku.com](http://www.heroku.com) .

Download & install the command line tools at : <https://devcenter.heroku.com/articles/getting-started-with-python#set-up> .

Enter terminal and enter the command:

```
heroku login
```

Your credentials will be asked. Enter them, and you'll be in contact with Heroku.

#### 2. New project

Define a new Flask project in PyCharm. Example- Helloku2 .

When you run the app locally, you should be seeing a web page.

To prepare your app for live usage, we need a real web server. Add gunicorn to your project.

In the terminal, ensure you can start the app by typing:

```
gunicorn app:app
```

The application should be accessible in your browser. You can terminate the application with CTRL+C.

### 3. Prepare project for Heroku

Create a new file called Procfile (no extension) in the root of your application, which contains one line:

```
web gunicorn app:app
```

This will tell Heroku that this is a web application, and start it with gunicorn (like in the terminal).

Go to the terminal and type:

```
pip freeze > requirements.txt
```

This command will put all of your dependencies into a file called requirements.txt . Heroku will use this file to build your virtual environment on the server side.

Click the menu VCS -> Enable and select Git. Select “Commit” from the toolbar, and make it ignore .idea and venv folders. Click “Commit” to complete the operation.

### 4. Submit to Heroku

First, we need to create a new app. Go to terminal and type:

```
heroku create helloku2
```

This will create a new application called helloku2. If this name is taken on Heroku, you might need to pick a new name. To enable it for git, type:

```
heroku git:remote -a helloku2
```

Finally, type the following command to submit your app:

```
git push heroku master
```

### 5. Run on Heroku

You start by assigning your app some server resources by typing:

```
heroku ps:scale web=1
```

When you enter <https://helloku2.herokuapp.com/> , you see your application running on the web.

To stop your application, you simply scale it down to zero by typing:

```
heroku ps:scale web=0
```

You can also see your applications on [www.heroku.com](http://www.heroku.com) , under your account dashboard.

## 6. Environment Variables

It is a common task to set & use environment variables in PyCharm.

You can set an environment variable in Heroku by typing:

```
heroku config:set GITHUB_USERNAME=username
```

You can remove an environment variable from Heroku by typing:

```
heroku config:unset GITHUB_USERNAME
```

You can list your Heroku environment variables by typing:

```
heroku config
```

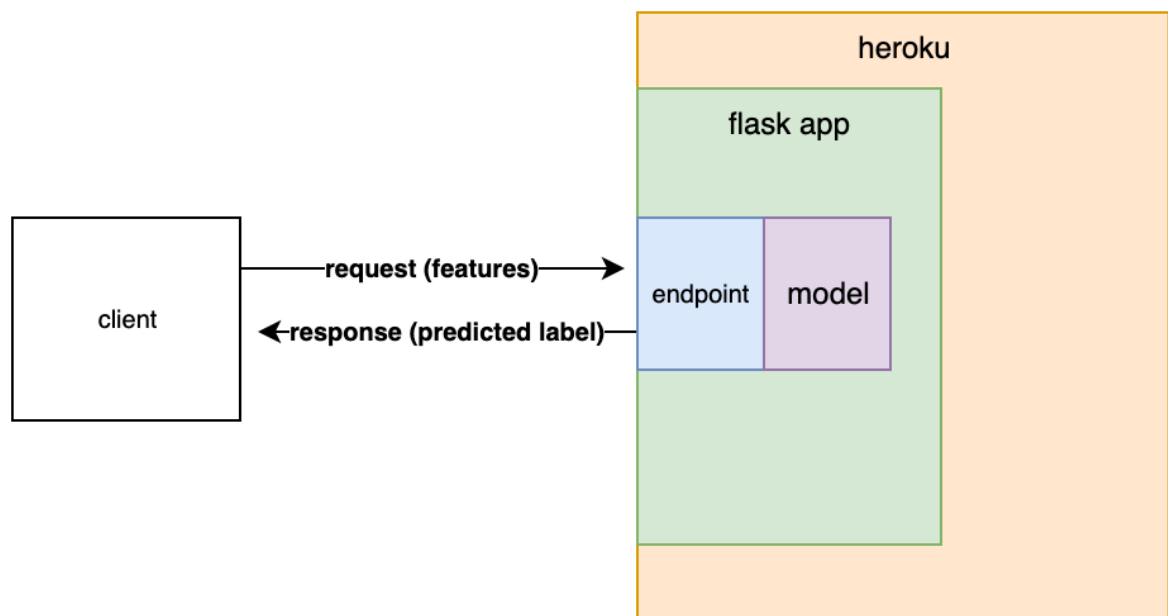


Fig. 4.5 - Architecture for deploying Flask app into production using Heroku

## 5. EVALUATION

### 5.1 DATASET

The dataset used contains the full original CT scans of 377 persons. There are 15589 and 48260 CT scan images belonging to 95 Covid-19 and 282 normal persons, respectively. The images of this dataset are 16-bit uint grayscale in TIFF format, so they cannot be visualized with normal monitors, they would simply appear as black images. To make these images visible with regular monitors, we converted them to float by dividing each image's pixel value by the maximum pixel value of that image. This way, the output images has a 32bit float type pixel values that could be visualized by regular monitors, and the quality of the images is good enough for analysis.

COVID-19 Patients	Normal People	COVID-19 Images	Normal Images
95	282	15589	48260

Fig. 5.1 – Dataset description

Some sample images from the dataset are displayed below

#### Positive COVID-19 CT Scan



Fig. 5.2 - Positive COVID-19 CT Scan images

Negative COVID-19 CT Scan

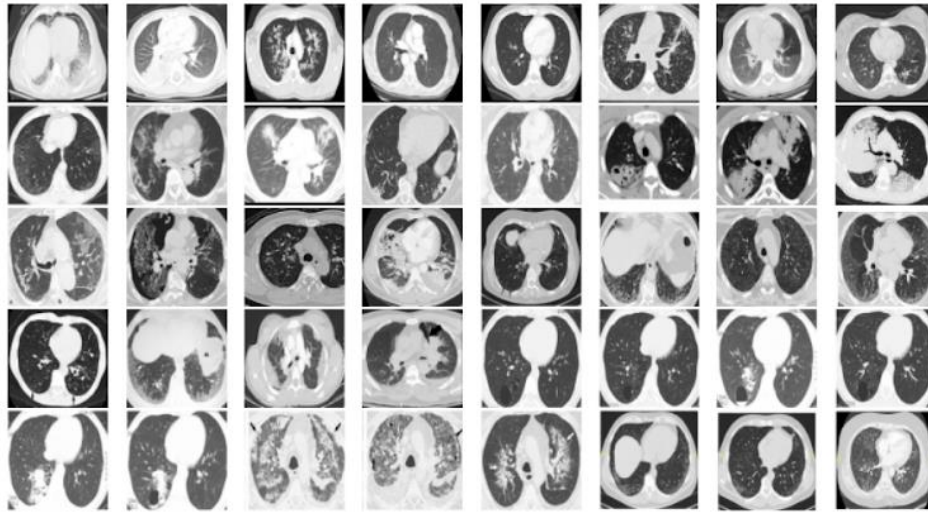


Fig. 5.3 - Negative COVID-19 CT Scan images

## 5.2 EVALUATION PROCEDURE

### 5.2.1 Classification Report

The classification report visualizer displays the precision, recall, F1, and support scores for the model.

There are four ways to check if the predictions are right or wrong:

- TN / True Negative: the case was negative and predicted negative
- TP / True Positive: the case was positive and predicted positive
- FN / False Negative: the case was positive but predicted negative
- FP / False Positive: the case was negative but predicted positive

1. **Precision:** Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. (Accuracy of positive predictions)

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

2. **Recall:** Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. (Fraction of positives that were correctly identified)



$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

3. **F1 score:** The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

4. **Support:** Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

**Classification Report**

```
from sklearn.metrics import classification_report
print(classification_report(y_test_bin, y_pred_bin))
```

	precision	recall	f1-score	support
0	0.93	0.93	0.93	70
1	0.94	0.94	0.94	80
accuracy			0.93	150
macro avg	0.93	0.93	0.93	150
weighted avg	0.93	0.93	0.93	150

Fig. 5.4 - Classification report of our model with Softmax Activation function and training batch size 32

### 5.2.2 Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix. Here are a few definitions, you need to remember for a confusion matrix :

- Accuracy : the proportion of the total number of predictions that were correct.
- Positive Predictive Value or Precision : the proportion of positive cases that were correctly identified.
- Negative Predictive Value : the proportion of negative cases that were correctly identified.

- Sensitivity or Recall : the proportion of actual positive cases which are correctly identified.
- Specificity : the proportion of actual negative cases which are correctly identified.

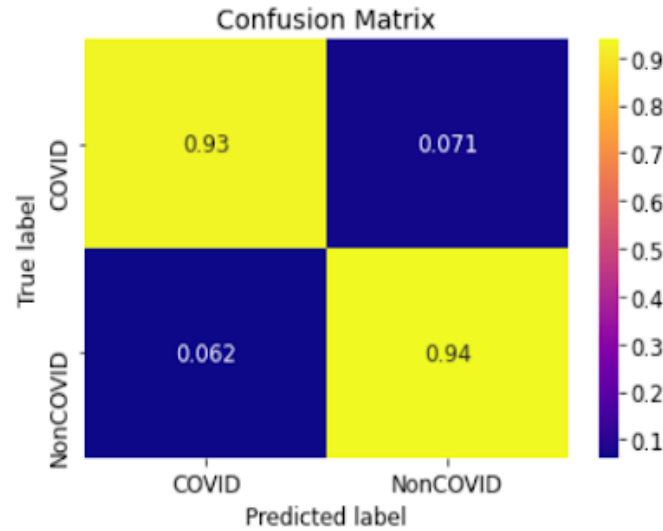


Fig. 5.5 – Confusion Matrix of our model with Softmax activation function and batch size 32

### 5.2.3 ROC Curve

The ROC curve is the plot between sensitivity and (1- specificity). (1- specificity) is also known as false positive rate and sensitivity is also known as True Positive rate.

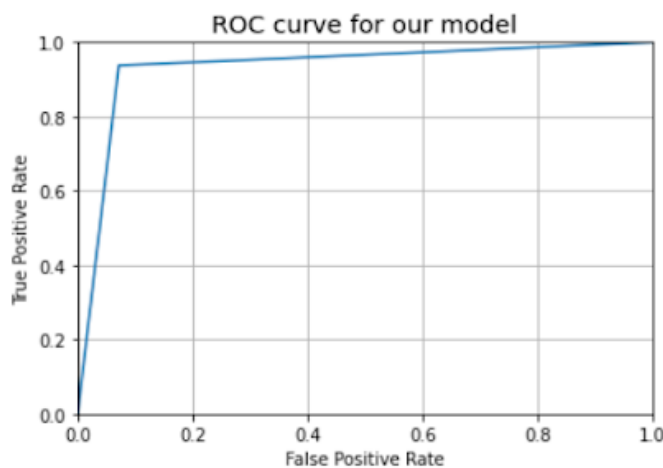


Fig. 5.6 - ROC curve of our model with Softmax activation function and batch size 32

### 5.2.4 Accuracy Loss Plot

Accuracy is a method for measuring a classification model's performance. It is typically expressed as a percentage. Accuracy is the count of predictions where the predicted value is equal to the true value.

A loss function, also known as a cost function, takes into account the probabilities or uncertainty of a prediction based on how much the prediction varies from the true value.

- **Relationship Between Accuracy and Loss:**

Most of the time we would observe that accuracy increases with the decrease in loss -- but this is not always the case. Accuracy and loss have different definitions and measure different things. They often appear to be inversely proportional but there is no mathematical relationship between these two metrics.

Visualizing the training loss vs. validation loss or training accuracy vs. validation accuracy over a number of epochs is a good way to determine if the model has been sufficiently trained. This is important so that the model is not undertrained and not overtrained such that it starts memorizing the training data which will, in turn, reduce its ability to predict accurately.

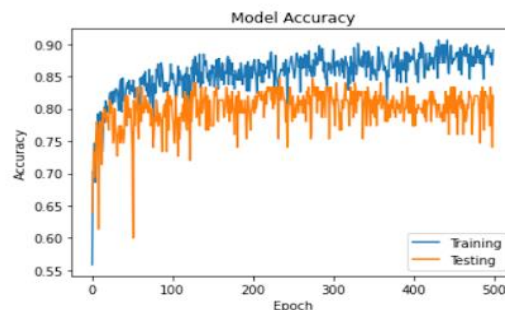


Fig. 5.7 - Accuracy Vs Epoch Plot

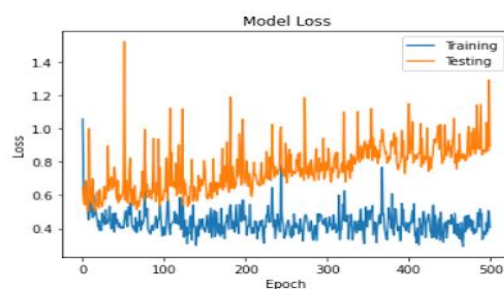


Fig.5.8 - Loss Vs Epoch Plot

## 5.3 TEST CASES

### 5.3.1 COVID Positive Prediction

- **Home Screen**

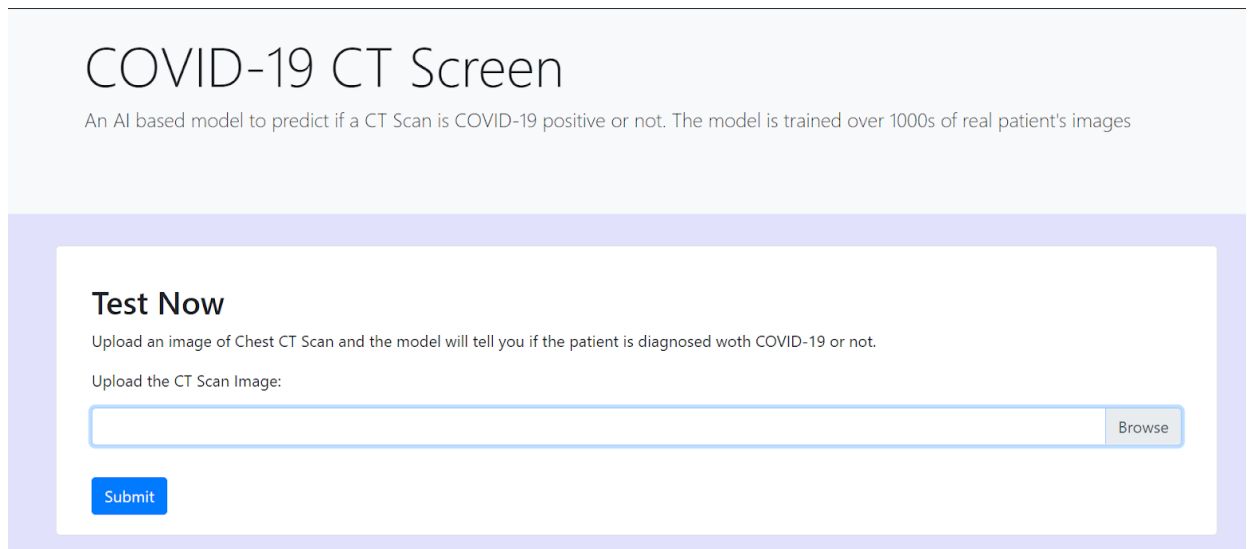


Fig. 5.9 – Home Screen GUI of our application

- **Selecting input image**

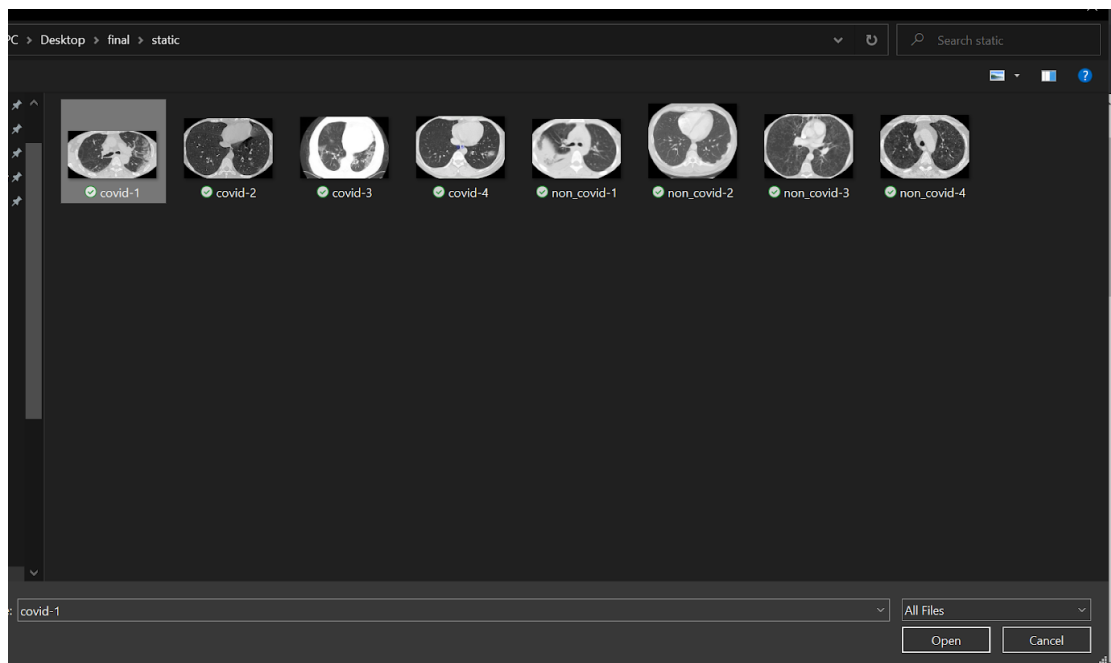


Fig. 5.10 – Selecting an input image

- **Viewing Predictions**

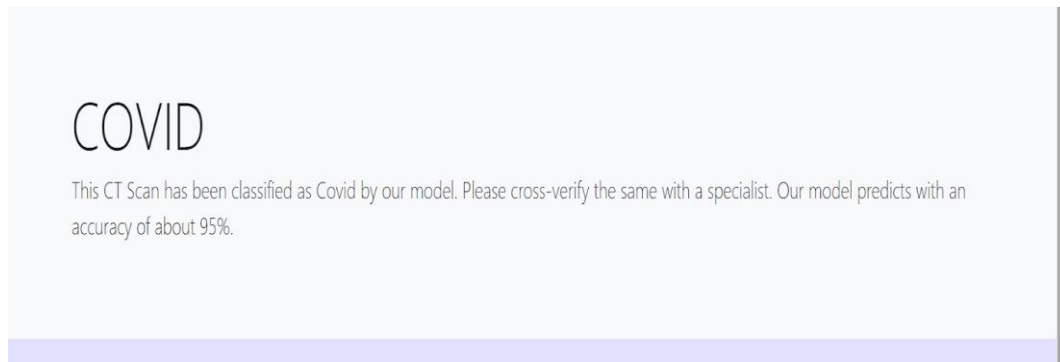


Fig. 5.11 – COVID prediction made by the model

### 5.3.2 COVID Negative Prediction

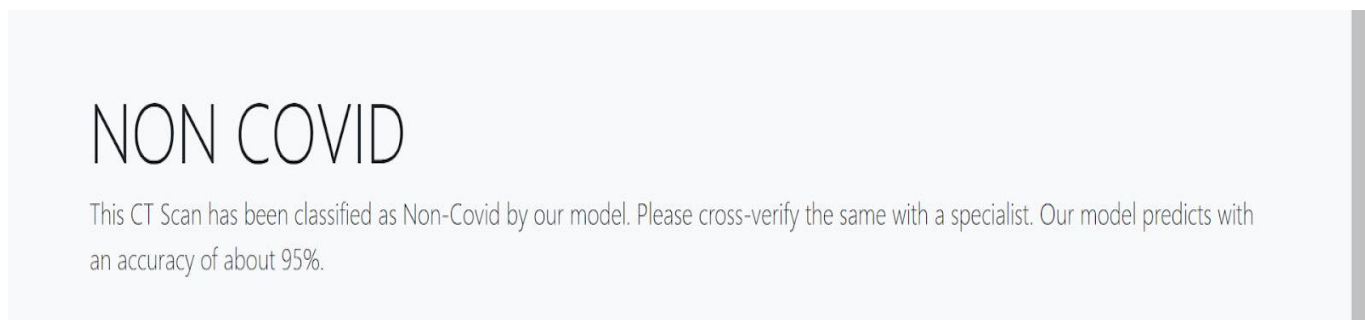


Fig. 5.12- Non COVID Prediction made by the model

## 5.4 OBSERVATIONS & RESULTS

### 5.4.1 Observations

Screening of COVID-19 using Chest CT images is a Classification problem with two class labels COVID and Non-COVID. To build a model which gives best results with better performance we have experimented by applying different parameters to the selected VGG-19 model. The observations are as follows:

#### 1. Model With Different Activation Functions In The Last Fully Connected Layer

We have trained the model using three different activation functions, which are Softmax, ReLU and Sigmoid, to check which function will give us better results. Following are our observations:

Table 5.1 - Metrics obtained using different activation functions

Activation Function	Accuracy	Sensitivity	Specificity
SoftMax	93%	93.75%	92.98%
Sigmoid	82%	76.19%	94.21%
ReLU	80%	78.3%	82.11%

- **Evaluation Criteria:**

#### I. Classification Report:

Following are the Classification reports for the models.

##### 1. Classification report for VGG-19 model with Softmax activation function

	precision	recall	f1-score	support
0	0.93	0.93	0.93	70
1	0.94	0.94	0.94	80
accuracy			0.93	150
macro avg	0.93	0.93	0.93	150
weighted avg	0.93	0.93	0.93	150

Fig. 5.13 - Classification report for model with Softmax function

## 2. Classification report for VGG-19 model with Sigmoid activation function

	precision	recall	f1-score	support
0	0.74	0.96	0.83	70
1	0.95	0.70	0.81	80
accuracy			0.82	150
macro avg	0.84	0.83	0.82	150
weighted avg	0.85	0.82	0.82	150

Fig. 5.14 – Classification report for model with Sigmoid function

## 3. Classification report for VGG-19 model with ReLU activation function

	precision	recall	f1-score	support
0	0.76	0.83	0.79	70
1	0.84	0.78	0.81	80
accuracy			0.80	150
macro avg	0.80	0.80	0.80	150
weighted avg	0.80	0.80	0.80	150

Fig. 5.15 – Classification report for model with ReLU function

## II. Accuracy and Loss Plots

## 1. For Softmax Activation function

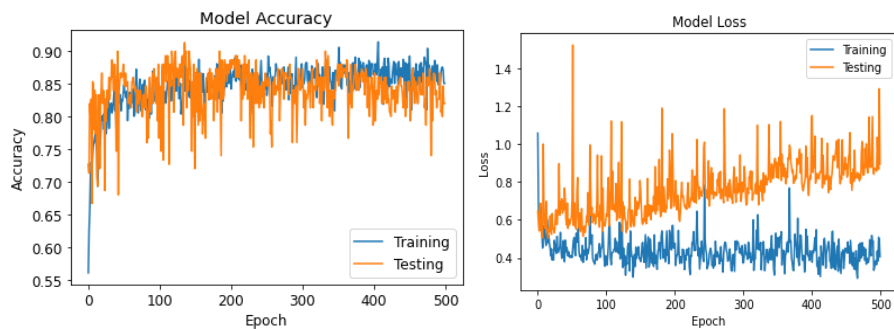


Fig. 5.16 – Accuracy and Loss Plots for model with Softmax function

## 2. For Sigmoid Activation Function

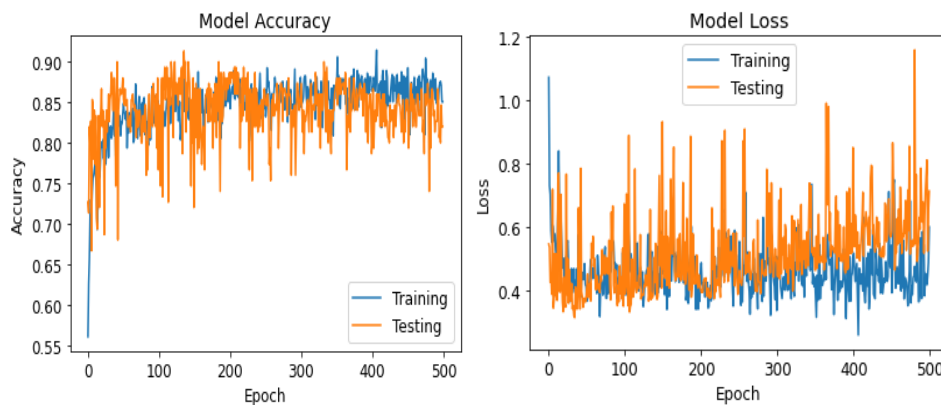


Fig. 5.17 – Accuracy and Loss Plots for model with Sigmoid function

### 3. For ReLU Activation function

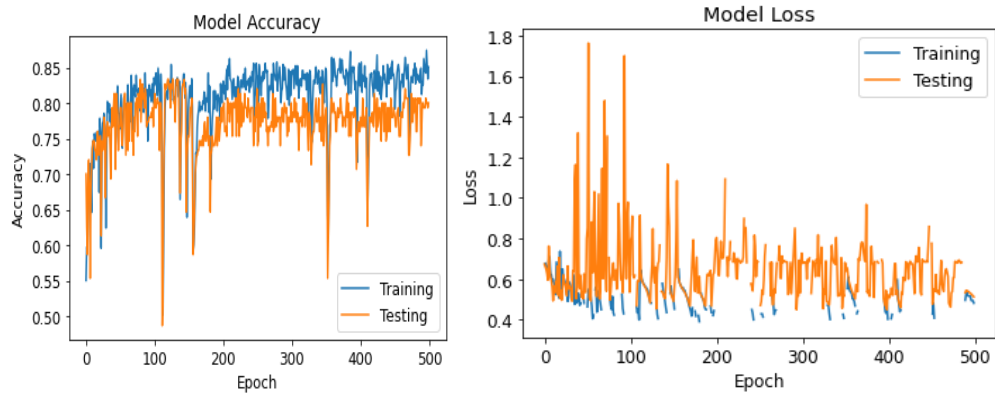


Fig. 5.18 – Accuracy and Loss Plots for model with ReLU function

### III. Confusion Matrix

#### 1. For Softmax Activation Function

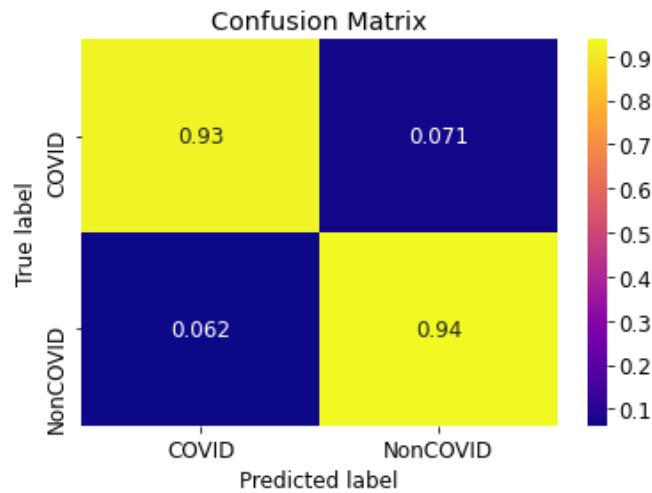


Fig. 5.19 – Confusion Matrix for model with Softmax function

#### 2. For Sigmoid Function

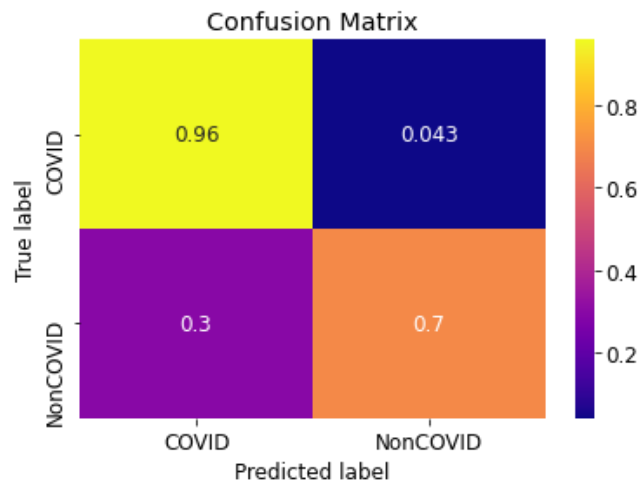


Fig. 5.20 – Confusion Matrix for model with Sigmoid function



### 3. For ReLU Activation Function

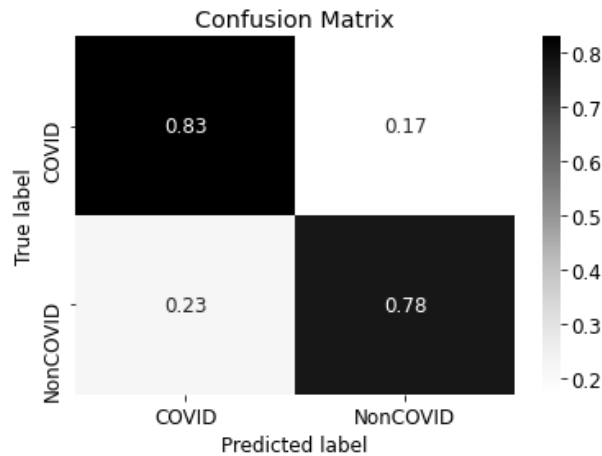


Fig. 5.21 – Confusion Matrix for model with ReLU function

Based on the above results it is evident that the VGG-19 model with SoftMax activation function has the highest accuracy of 93% with better True Positive and True Negative rates. ReLU has the least accuracy of 80% which is nearer to the average performing standard Sigmoid activation function with 82% accuracy. This shows that SoftMax gives better performance for binary classification, whereas the Non-Linear ReLU activation function works well for classifying images into more than two classes.

## 2. Model With Different Loss Functions

After obtaining the suitable activation function, we have experimented with various loss functions that can improve our classifier's performance. Following are our observations:

Table 5.2 - Metrics obtained using different loss functions

Loss Function	Accuracy	Specificity	Sensitivity
Categorical cross-entropy	93%	93.75%	92.98%
Binary cross-entropy	81%	81%	81%

- Binary cross entropy is a loss function that is used in binary classification tasks. These are tasks that answer a question with only two choices (yes or no, A or B, 0 or 1, left or right). Several independent such questions can be answered at the same time, as in multi-label classification or in binary image segmentation.
- Categorical cross entropy also called Softmax Loss, is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification.

- **Evaluation Criteria:**

- I. Classification Report:

Following are the Classification reports for the models.

1. Classification report for VGG-19 model with Categorical Cross Entropy loss function

	precision	recall	f1-score	support
0	0.93	0.93	0.93	70
1	0.94	0.94	0.94	80
accuracy			0.93	150
macro avg	0.93	0.93	0.93	150
weighted avg	0.93	0.93	0.93	150

Fig. 5.22 – Classification report for model with categorical cross entropy function

2. Classification report for VGG-19 model with Binary Cross Entropy loss function

	precision	recall	f1-score	support
0	0.76	0.83	0.79	70
1	0.84	0.78	0.81	80
accuracy			0.80	150
macro avg	0.80	0.80	0.80	150
weighted avg	0.80	0.80	0.80	150

Fig. 5.23 – Classification report for model with binary cross entropy function

## II. Accuracy and Loss Plots

### 1. For Categorical Cross Entropy Loss Function

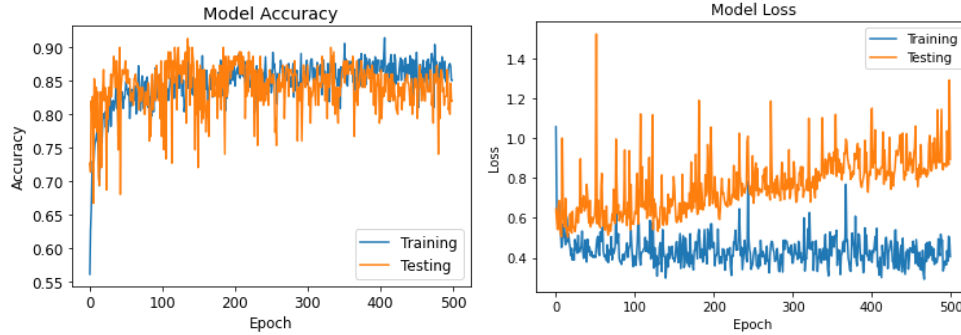


Fig. 5.24 – Accuracy and Loss plots for model with categorical cross entropy function

### 2. For Binary Cross Entropy Loss Function

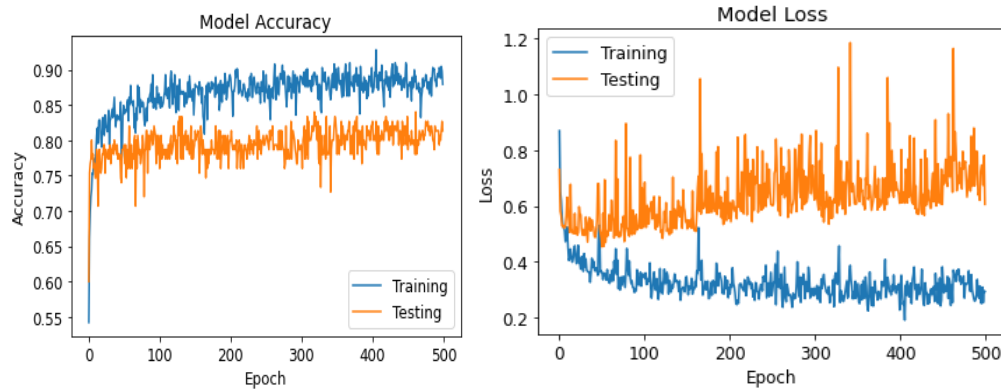


Fig. 5.25 – Accuracy and Loss plots for model with binary cross entropy function

## III. Confusion Matrix

### 1. For Categorical Cross Entropy Loss Function

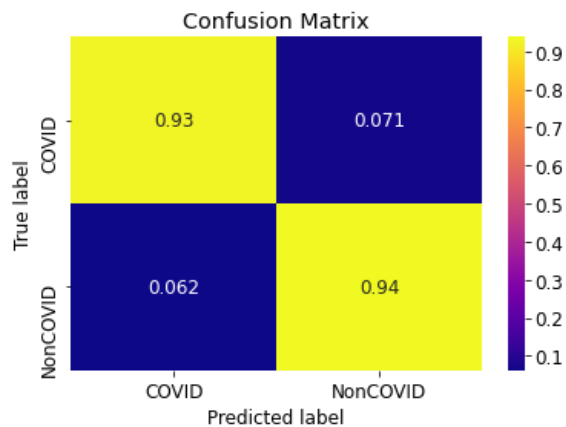


Fig. 5.26 – Confusion matrix for model with categorical cross entropy function

## 2. For Binary Cross Entropy Loss Function

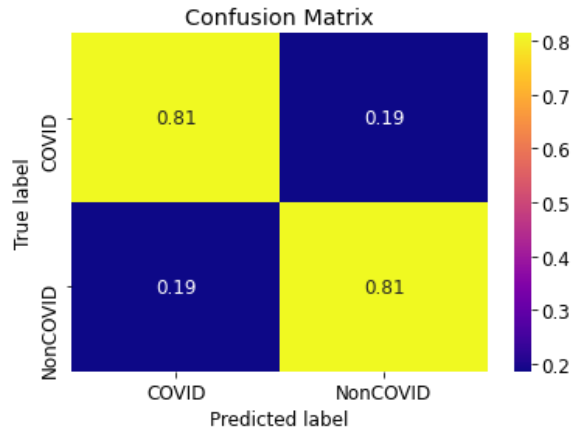


Fig. 5.27 – Confusion matrix for model with binary cross entropy function

The Categorical cross entropy loss function clearly gives the highest accuracy of 93% with better True Positive and True Negative Rates. As it is evident from the Accuracy and Loss plots the model is undertrained in the case of Binary cross entropy function where the losses reaches the maximum scale during initial epochs in training phase, which explains the under performance of the model with this loss function, where the accuracy is dropped to 81%.

## 3. Model With Different Batch Sizes For Training

After building the model with suitable activation and loss function, we have experimented around various training batch sizes, during the training phase. Following are our observations:

Table 5.3 - Metrics obtained using different batch size

Batch Size	Accuracy	Sensitivity	Specificity
8	75%	68.75%	97.75%
16	81%	78.38%	85.4%
32	93%	93.75%	92.98%

- **Evaluation Criteria:**

### I. Classification Report:

Following are the Classification reports for the models.

#### 1. Classification report for VGG-19 model with training batch size 8

	precision	recall	f1-score	support
0	0.66	0.99	0.79	70
1	0.98	0.55	0.70	80
accuracy			0.75	150
macro avg	0.82	0.77	0.75	150
weighted avg	0.83	0.75	0.74	150

Fig. 5.28 – Classification report for model with batch size - 8

#### 2. Classification report for VGG-19 model with training batch size 16

	precision	recall	f1-score	support
0	0.76	0.87	0.81	70
1	0.87	0.76	0.81	80
accuracy			0.81	150
macro avg	0.82	0.82	0.81	150
weighted avg	0.82	0.81	0.81	150

Fig. 5.29 – Classification report for model with batch size - 16

#### 3. Classification report for VGG-19 model with training batch size 32

	precision	recall	f1-score	support
0	0.93	0.93	0.93	70
1	0.94	0.94	0.94	80
accuracy			0.93	150
macro avg	0.93	0.93	0.93	150
weighted avg	0.93	0.93	0.93	150

Fig. 5.30 – Classification report for model with batch size - 32

## II. Accuracy and Loss Plots

### 1. For Training Batch Size 8

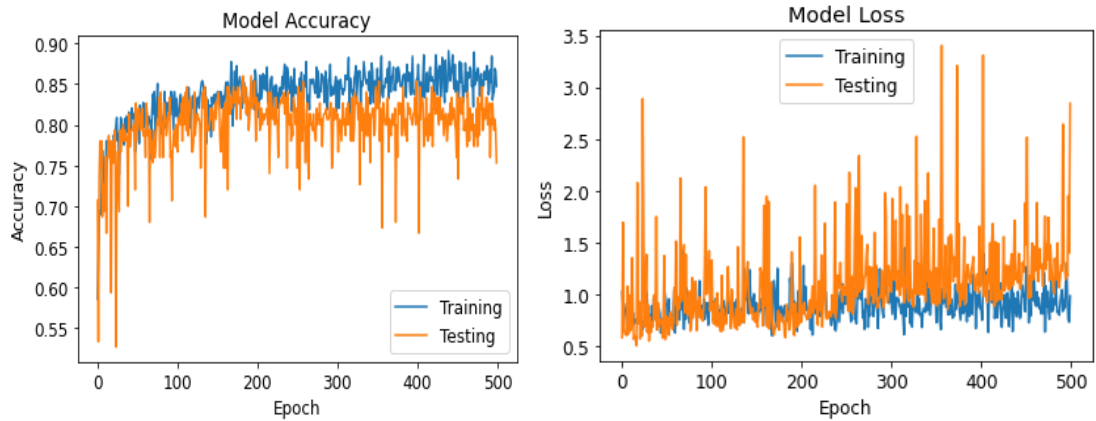


Fig. 5.31 – Accuracy and Loss plots for model with batch size – 8

### 2. For Training Batch Size 16

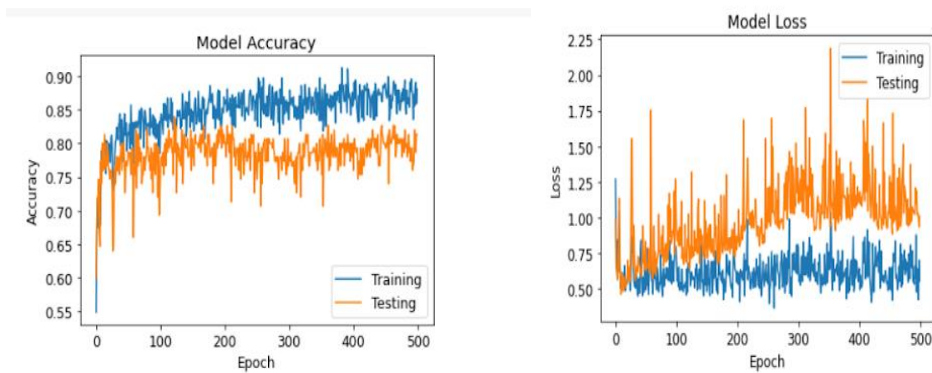


Fig. 5.32 – Accuracy and Loss plots for model with batch size - 16

### 3. For Training Batch Size 32

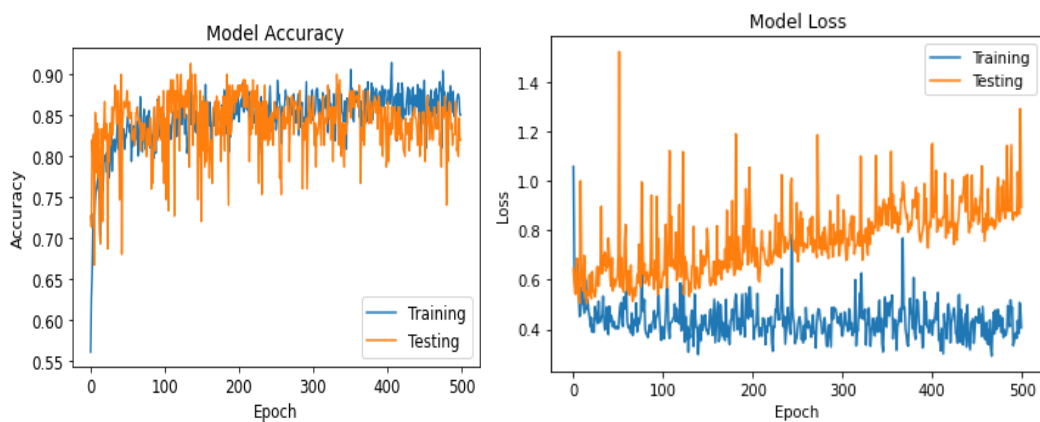


Fig. 5.33 – Accuracy and Loss plots for model with batch size - 32

### III. Confusion Matrix

#### 1. For Training Batch Size 8

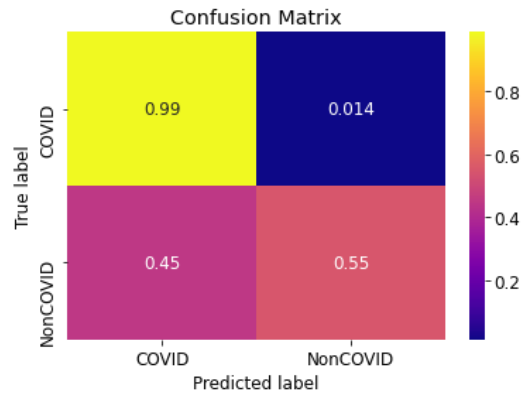


Fig. 5.34 – Confusion Matrix for model with batch size - 8

#### 2. For Training Batch Size 16

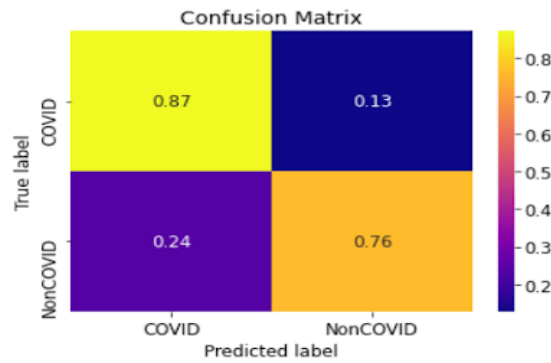


Fig. 5.35 – Confusion Matrix for model with batch size - 16

#### 3. For Training Batch Size 32

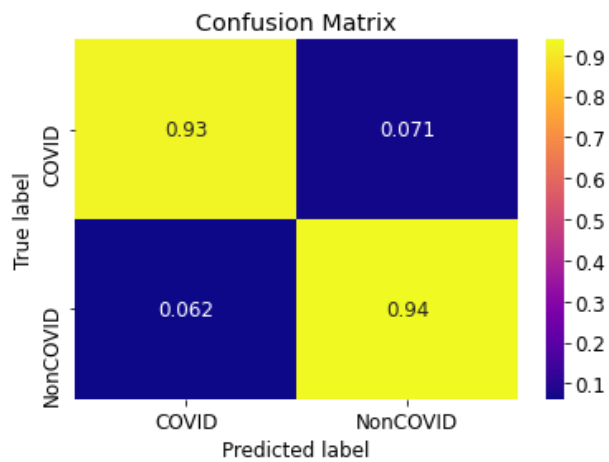


Fig. 5.36 – Confusion Matrix for model with batch size - 32

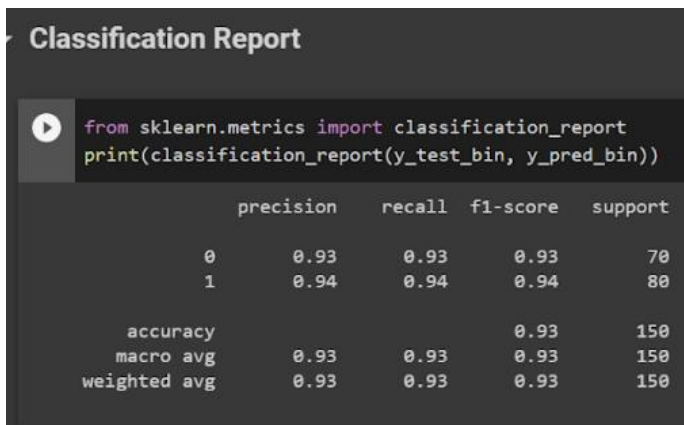
From the above observations it is clear that as the training batch size increased there was an improvement in the performance of the model. The number of epochs were 500. The model with training batch size of 32 and number of epochs as 500 gave us the better performance

## 5.2 RESULTS

After the analysis from the above observations, we have selected the model with following parameters.

- Activation function : Softmax
- Loss function : Categorical-cross entropy
- Training batch size : 32
- Accuracy achieved : 93%

The results of the final model are given below:



The image shows a Jupyter Notebook cell titled 'Classification Report'. It contains a code snippet that imports the classification\_report function from sklearn.metrics and prints the report for y\_test\_bin and y\_pred\_bin. Below the code, the output of the report is displayed as a table.

	precision	recall	f1-score	support
0	0.93	0.93	0.93	70
1	0.94	0.94	0.94	80
accuracy			0.93	150
macro avg	0.93	0.93	0.93	150
weighted avg	0.93	0.93	0.93	150

Fig. 5.37 – Classification report for VGG-19 model with Softmax activation function, Categorical cross entropy loss function, and training batch size - 32



## 6. CONCLUSION & FUTURE ENHANCEMENTS

Our project demonstrates the determination of COVID-19 in the used dataset. Using the current limited and challenging COVID-19 datasets, VGG19 model could be used to develop suitable deep learning-based tools for COVID-19 detection. Initially while selecting the model, we found that both VGG16 and VGG19 classifiers provided good results within the experimental constraints of the small number of currently available COVID-19 medical images. But deeper networks generally struggle, while they will perform better when larger datasets are available which will reduce the impact of data quality variation. And we have chosen to utilise the VGG-19 model because even though both VGG-16 and VGG-19 Converged well, in VGG-16 overfitting was evident from 5 epochs and in VGG-19 overfitting was evident from 20 epochs[1].

We have worked on our project using the VGG-19 model with three different activation functions, which are SoftMax, ReLU and Sigmoid. Among these three variants we found that the VGG-19 model with SoftMax activation function has the highest accuracy of 93%, the model with standard sigmoid activation function has accuracy of 80% and ReLU has the least with 80% accuracy. From these observations, the VGG-19 model with SoftMax activation function gave better output i.e., whether the given input HRCT image belongs to a COVID-19 positive patient or a Healthy patient.

Our future work will focus on improving the performance of the system and we intend to increase the size of the dataset by adding new CT Scan images of patients with COVID-19, and also use the results produced by our model to extended the project to include the critical information about the spread of virus in the patient's lungs i.e predicting some important factors like CO-RADS\* level. And since the data is limited and not proportionate we intend to implement imbalance classification. Several other models can also be built on top of this model which in turn will be useful in detecting families of such viral diseases. The tool has potential to be utilized by radiologists when encompassed in a seamless workflow.

## REFERENCES

- [1] U. Niyaz, A. S. Sambyal and Devanand, "Advances in Deep Learning Techniques for Medical Image Analysis," 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2018, pp. 271-277, doi: 10.1109/PDGC.2018.8745790.
- [2] M. J. Horry et al., "COVID-19 Detection Through Transfer Learning Using Multimodal Imaging Data," in IEEE Access, vol. 8, pp. 149808-149824, 2020, doi: 10.1109/ACCESS.2020.3016780.
- [3] J. Liu, Z. Zhang, L. Zu, H. Wang and Y. Zhong, "Intelligent Detection for CT Image of COVID-19 using Deep Learning," 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2020, pp. 76-81, doi: 10.1109/CISP-BMEI51763.2020.9263690.
- [4] A. K. Abdar, S. M. Sadjadi, H. Soltanian-Zadeh, A. Bashirgonbadi and M. Naghibi, "Automatic Detection of Coronavirus (COVID-19) from Chest CT Images using VGG16-Based Deep-Learning," 2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME), 2020, pp. 212-216, doi: 10.1109/ICBME51989.2020.9319326.
- [5] A. Mohammed et al., "Weakly-Supervised Network for Detection of COVID-19 in Chest CT Scans," in IEEE Access, vol. 8, pp. 155987-156000, 2020, doi: 10.1109/ACCESS.2020.3018498.
- [6] M. J. Horry et al., "COVID-19 Detection Through Transfer Learning Using Multimodal Imaging Data," in IEEE Access, vol. 8, pp. 149808-149824, 2020, doi: 10.1109/ACCESS.2020.3016780.
- [7] <https://www.brookings.edu/research/the-complexity-of-managing-covid-19-how-important-is-good-governance/>
- [8] <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
- [9] <https://mosmed.ai/en/>
- [10] <https://medium.com/analytics-vidhya/image-classification-a-comparison-of-dnn-cnn-and-transfer-learning-approach-704535beca25>
- [11] Tayarani N MH. Applications of artificial intelligence in battling against covid-19: A literature review. Chaos Solitons Fractals. 2021 Jan;142:110338. doi: 10.1016/j.chaos.2020.110338. Epub 2020 Oct 3. PMID: 33041533; PMCID: PMC7532790.
- [12] Singh M, Bansal S, Ahuja S, et al. Transfer learning based ensemble support vector machine model for automated COVID-19 detection using lung computerized tomography scan data. Research Square; 2020. DOI: 10.21203/rs.3.rs-32493/v1.
- [13] <https://www.sciencedirect.com/science/article/pii/S1746809421001853?via%3Dihub>
- [14] A Fully Automated Deep Learning-based Network For Detecting COVID-19 from a New And Large Lung CT Scan Dataset Mohammad Rahimzadeh, Abolfazl Attar, Seyed Mohammad Sakhaei  
medRxiv 2020.06.08.20121541; doi: <https://doi.org/10.1101/2020.06.08.20121541>

## APPENDIX

### Source Code:

#### Model.py

```

from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/My Drive/data'
from builtins import range, input
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, GlobalAveragePooling2D,
Dropout
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.applications import VGG19
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, roc_curve
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from glob import glob
import pandas as pd
import cv2
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
#define size to which images are to be resized
IMAGE_SIZE = [224, 224]
# training config
epochs = 500
batch_size = 32
#define paths
covid_path = '/content/drive/MyDrive/data/CT_COVID'
noncovid_path = '/content/drive/MyDrive/data/CT_NonCOVID'
# Use glob to grab images from path
covid_files = glob(covid_path + '/*')

```

```

noncovid_files = glob(noncovid_path + '/*')
# Visualize file variable contents
print("First 5 Covid Files: ",covid_files[0:5])
print("Total Count: ",len(covid_files))
print("First 5 NonCovid Files: ",noncovid_files[0:5])
print("Total Count: ",len(noncovid_files))
# Fetch Images and Class Labels from Files
covid_labels = []
noncovid_labels = []
covid_images=[]
noncovid_images=[]
for i in range(len(covid_files)):
    image = cv2.imread(covid_files[i]) # read file
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # arrange format as per keras
    image = cv2.resize(image,(224,224)) # resize as per model
    covid_images.append(image) # append image
    covid_labels.append('CT_COVID') #append class label
for i in range(len(noncovid_files)):
    image = cv2.imread(noncovid_files[i])
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image,(224,224))
    noncovid_images.append(image)
    noncovid_labels.append('CT_NonCOVID')
def plot_images(images, title):
    nrows, ncols = 5, 8
    figsize = [10, 6]
    fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=figsize, facecolor=(1, 1, 1))
    for i, axi in enumerate(ax.flat):
        axi.imshow(images[i])
        axi.set_axis_off()
    plt.suptitle(title, fontsize=24)
    plt.tight_layout(pad=0.2, rect=[0, 0, 1, 0.9])
    plt.show()
plot_images(covid_images, 'Positive COVID-19 CT Scan')
plot_images(noncovid_images, 'Negative COVID-19 CT Scan')

```

```

# Convert to array and Normalize to interval of [0,1]
covid_images = np.array(covid_images) / 255
noncovid_images = np.array(noncovid_images) / 255
# Split into training and testing sets for both types of images
covid_x_train, covid_x_test, covid_y_train, covid_y_test = train_test_split(
    covid_images, covid_labels, test_size=0.2)
noncovid_x_train, noncovid_x_test, noncovid_y_train, noncovid_y_test = train_test_split(
    noncovid_images, noncovid_labels, test_size=0.2)
# Merge sets for both types of images
X_train = np.concatenate((noncovid_x_train, covid_x_train), axis=0)
X_test = np.concatenate((noncovid_x_test, covid_x_test), axis=0)
y_train = np.concatenate((noncovid_y_train, covid_y_train), axis=0)
y_test = np.concatenate((noncovid_y_test, covid_y_test), axis=0)
# Make labels into categories - either 0 or 1, for our model
y_train = LabelBinarizer().fit_transform(y_train)
y_train = to_categorical(y_train)
y_test = LabelBinarizer().fit_transform(y_test)
y_test = to_categorical(y_test)
plot_images(covid_x_train, 'X_train')
plot_images(covid_x_test, 'X_test')
# y_train and y_test contain class labels 0 and 1 representing COVID and NonCOVID for
X_train and X_test
# Building Model
vggModel = VGG19(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
outputs = vggModel.output
outputs = Flatten(name="flatten")(outputs)
outputs = Dropout(0.5)(outputs)
outputs = Dense(2, activation="softmax")(outputs)
model = Model(inputs=vggModel.input, outputs=outputs)
for layer in vggModel.layers:
    layer.trainable = False
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',

```

```

        metrics=['accuracy']
    )
train_aug = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)
# Visualize Model
model.summary()
train_aug = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)
history = model.fit(train_aug.flow(X_train, y_train, batch_size=batch_size),
                    validation_data=(X_test, y_test),
                    validation_steps=len(X_test) / batch_size,
                    steps_per_epoch=len(X_train) / batch_size,
                    epochs=epochs)
# Save Model and Weights
model.save('vgg_ct.h5')
model.save_weights('vgg_weights_ct.hdf5')
# Load saved model
model = load_model('vgg_ct.h5')
y_pred = model.predict(X_test, batch_size=batch_size)
prediction=y_pred[0:10]
for index, probability in enumerate(prediction):
    if probability[1] > 0.5:
        plt.title("%.2f" % (probability[1]*100) + '% COVID')
    else:
        plt.title("%.2f" % ((1-probability[1])*100) + '% NonCOVID')
    plt.imshow(X_test[index])
    plt.show()
# Convert to Binary classes

```

```

y_pred_bin = np.argmax(y_pred, axis=1)
y_test_bin = np.argmax(y_test, axis=1)
fpr, tpr, thresholds = roc_curve(y_test_bin, y_pred_bin)
plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve for our model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.grid(True)
def plot_confusion_matrix(normalize):
    classes = ['COVID','NonCOVID']
    tick_marks = [0.5,1.5]
    cn = confusion_matrix(y_test_bin, y_pred_bin,normalize=normalize)
    sns.heatmap(cn,cmap='plasma',annot=True)
    plt.xticks(tick_marks, classes)
    plt.yticks(tick_marks, classes)
    plt.title('Confusion Matrix')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
print('Confusion Matrix without Normalization')
plot_confusion_matrix(normalize=None)
print('Confusion Matrix with Normalized Values')
plot_confusion_matrix(normalize='true')
from sklearn.metrics import classification_report
print(classification_report(y_test_bin, y_pred_bin))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Training', 'Testing'])
plt.savefig('vgg_ct_accuracy.png')

```

```
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Testing'])
plt.savefig('vgg_ct_loss.png')
plt.show()
```