

**COVID VISION: ADVANCED COVID-19 DETECTION FROM LUNGS  
X-RAYS WITH MACHINE LEARNING OR DEEP LEARNINGS**

**PROJECT REPORT**

*Submitted by*

**TEAM ID: NM2023TMID03654**

<b>Team Lead:</b>	<b>SAM NIJIN S (961420104048)</b>
<b>Team Member 1:</b>	<b>BENIHIM S S (961420104017)</b>
<b>Team Member 2:</b>	<b>GOKUL B S (961420104026)</b>
<b>Team Member 3:</b>	<b>HANEESH NOYAL T R S (961420104025)</b>

*In partial fulfilment for the award of the degree of*

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**MAR EPHRAEM COLLEGE OF ENGINEERING AND TECHNOLOGY**

# 1. INTRODUCTION

## 1.1 Project Overview

The main objective of this project is to make the covid testing very quickly with the usage of the X-Ray images. This will reduce the fear on the people for a long time waiting for their result from RT-PCR test, which will take more than a day to get the results. In this project we have used technologies like Deep Learning – training classification model, Django – web application (hospitals), Flutter – mobile application (public users). This project is built with the Convolutional Neural Network Algorithm which is capable of extracting the features of the image. By means of the transfer learning and ensemble technique this project will provide a good metrics and performs well on the real time data. The Django application is deployed in the server and the link is given below <https://covid-detective.gokulsreejith.com/>

## 1.2 Purpose

The major pandemic covid-19, is a deadliest disease which spreads through air. So, the people must get a periodic checkup or need to get checkup when they feel ill. Traditional method of testing is RT-PCR which, takes a lot amount of time to get the result until that the specific patient need to be quarantined. This will cause some impact in their and their family members mental health also will make them panic. So, with the technology advancement this project is built with the help of Artificial Intelligence which is capable of providing the result just with the X-Ray images. So, this process is really a faster one which makes the non-covid person to not be panic. Which is a good practice and also the good performing model with good metrics on evaluation should be trained else it may cause very serious issues. To make that possible the transfer learning technique with the upvoting technique of ensemble technique has been used in this project.

## 2. IDEATION & PROPOSED SOLUTION

### 2.1 Problem Statement Definition

#### Customer Problem Statement Template:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Patient	Checkup Covid-19	It takes long time	PCR & NAATs tests	Tensed and worried
PS-2	Doctor	Diagnose Covid-19	Testing takes huge time	Chemical test	Uncertainty

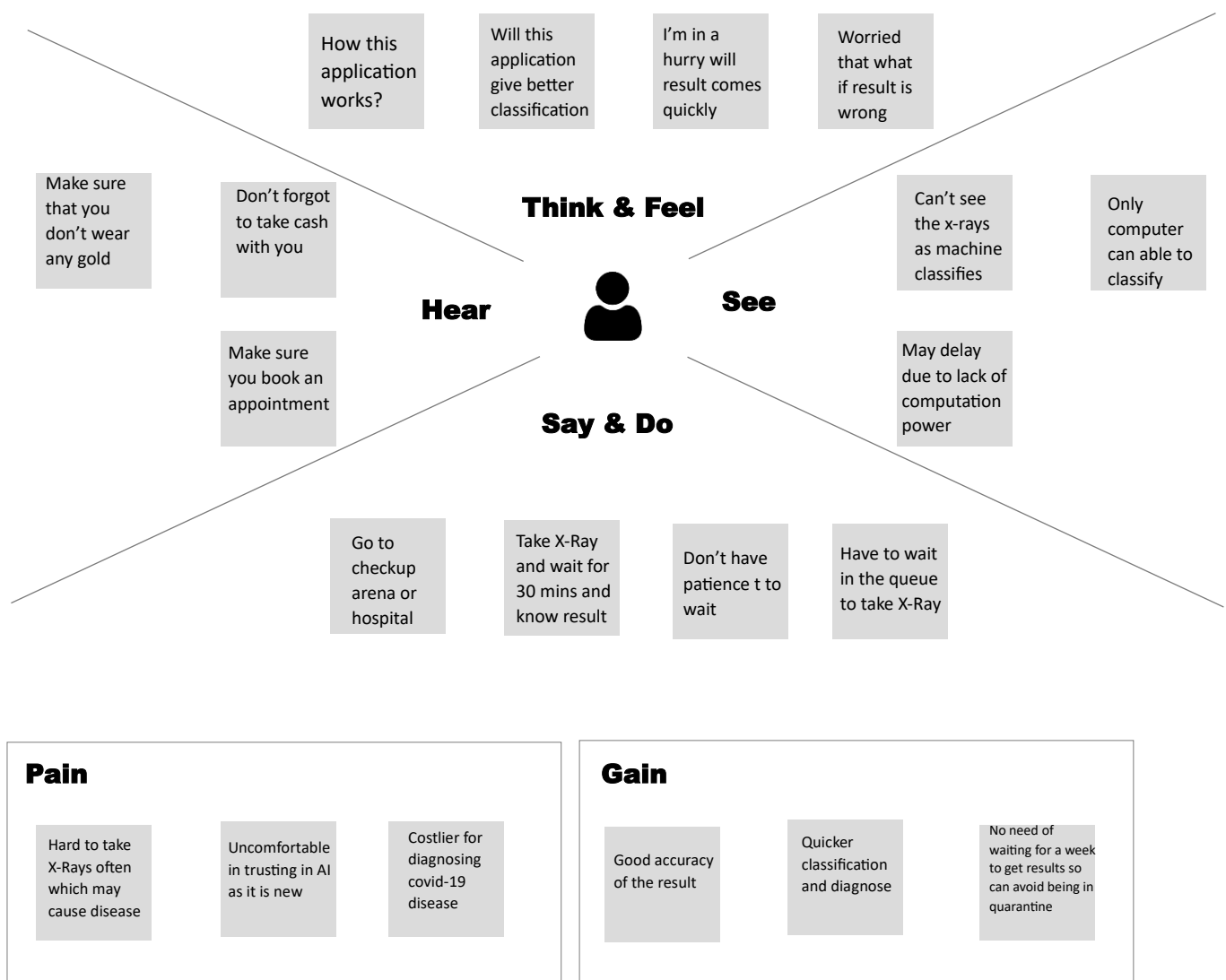
## 2.2 Empathy Map Canvas

### Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

### Covid Vision: Advanced COVID-19 Detection from Lung X-rays with Machine Learning or Deep Learnings




## 2.3 Ideation & Brainstroming

### Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

#### Step-1: Team Gathering, Collaboration and Select the Problem Statement



### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
🕒 1 hour to collaborate  
👤 2-8 people recommended

➔

#### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

---

**A** Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

**B** Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

**C** Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

#### Define your problem statement


What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

---

**PROBLEM**

Covid-19 disease is a pandemic which is more dangerous. The process of diagnosing it takes huge amount of time using PCR techniques, which is not an efficient way of testing the patients.



#### Key rules of brainstorming

To run a smooth and productive session

🗨️ Stay in topic.

🕒 Defer judgment.

🗨️ Go for volume.

💡 Encourage wild ideas.

👂 Listen to others.

👁️ If possible, be visual.

## Step-2: Brainstorm, Idea Listing and Grouping

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

**TIP**  
You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!



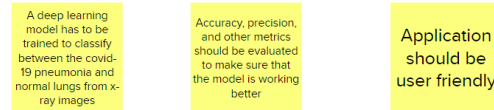
3

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

**TIP**  
Add customisable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.



## Step-3: Idea Prioritization

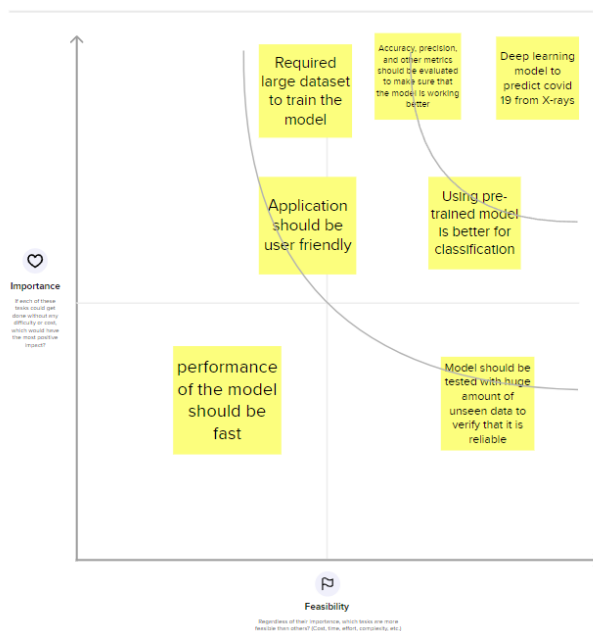
4

### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

**TIP**  
Participants can use their markers to point at where sticky notes should go on the grid. The facilitator can confirm the work by using the laser pointer holding the key on the keyboard.



## 2.4 Proposed Solution

### Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Covid-19 is a more dangerous pandemic. The process of diagnosing it takes a huge amount of time using PCR techniques, which is not an efficient way of diagnosing the patient. The time that patients have to wait for the results may result in anxiety and depression, which is also not good for their health.
2.	Idea / Solution description	The idea is to provide a deep learning model which is capable of classifying between the normal lung X-Rays and the covid-19 pneumonia X-Rays. So, this will produce an efficient way to diagnose the covid-19 disease.
3.	Novelty / Uniqueness	Already there exists three models which are as follows COVID-Net, DarkCovidNet, DeepCovid-XR. The model which is about to develop will be more efficient to produce more faultless predict. This is done with the ensemble techniques & uses grid search cv to find better hyperparameters combination to make the model results in predicting accordingly. And there will be a mobile app for public to make use of this application. So that the customer need not to wait in hospitals to take X-Rays and get results. They can get the result just from their finger tips if they have the recent X-Ray with them.
4.	Social Impact / Customer Satisfaction	Patients and their close as well as normal relatives' expectation can be fulfilled as this is a quick process of classifying between the normal X-Rays and the Covid pneumonia X-Rays.

5.	Business Model (Revenue Model)	This application will generate revenue as the new data are to be trained & updated in application which will be more efficient day by day, with the maintenance progress too. The mobile app for the public is allowed for a trial after that it will charges per month accordingly.
6.	Scalability of the Solution	The cost of implementing this application will be affordable to all the hospitals even though the small clinics can afford to it. Then there is a user login where the user can be able to get register and make use of this application. Even for the normal people and poor people too. So, the scalability of this application is really a huge one as it is efficient and low of costs which makes affordable to a wide range of customers.



### 3. REQUIREMENT ANALYSIS

#### Solution Requirements (Functional & Non-functional)

##### 3.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Username and password Registration through E-Mail
FR-2	User Confirmation	Confirmation via OTP Confirmation via Aadhar
FR-3	Image Format	The image should be taken from the app. If the source image is directly uploaded it must be DICOM or PNG format.
FR-4	User Data	The user has to provide details like which doctor they are consulting and from where the Xray is taken and relatedly

### 3.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	This application can be used with any age group people as per their needs. Mostly the person's who needs to get the result very quickly can use this.
NFR-2	<b>Security</b>	This application has the security feature, which allows a specific user to login to their dashboard and perform the test. This never allows other people to enter into some other persons dashboard without credentials. So, this maintains security.
NFR-3	<b>Reliability</b>	This system has the ability to tolerate fault and errors, if the user is trying to upload any wrong format of image it never allows the user to do and alerts him regarding the same.
NFR-4	<b>Performance</b>	The system is capable of performing very good with a speed of 1 minute or 1 minute and 30 seconds for predicting the image. But it is based on the plan the customer had chosen. If they chosen lighter plan the performance will be slower a little bit.
NFR-5	<b>Availability</b>	This application is deployed on the IBM Watson cloud which is a vast clous which is capable of allowing several users to work on the same time. So, this application will be available to each and every person all the time. The trial version is only available for the user for a single time there after the user must get the premium version of the application in order to pursue with it.
NFR-6	<b>Scalability</b>	This application has the tendency to control the traffic density which means that based on the level of the premium a user had taken the performance will be. If the higher-level premium user has requested the results and there isn't enough resources then the system will allocate the resources to the high level premium user.

## 4. PROJECT DESIGN

### 4.1 Data Flow Diagrams

#### Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

#### Example: [\(Simplified\)](#)

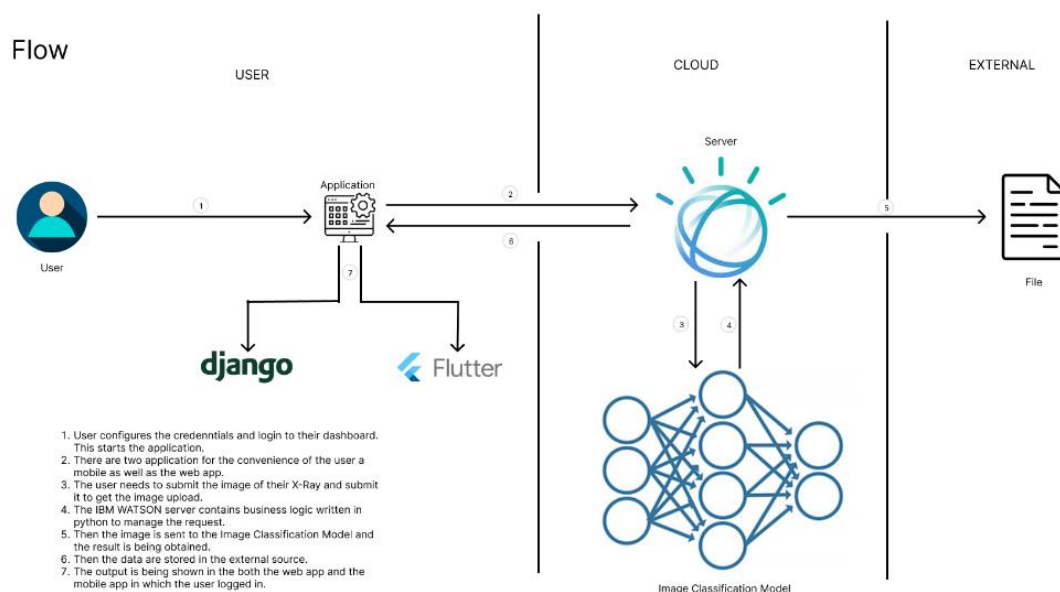


Fig: 4.1

Example: DFD Level 0 (Industry Standard)

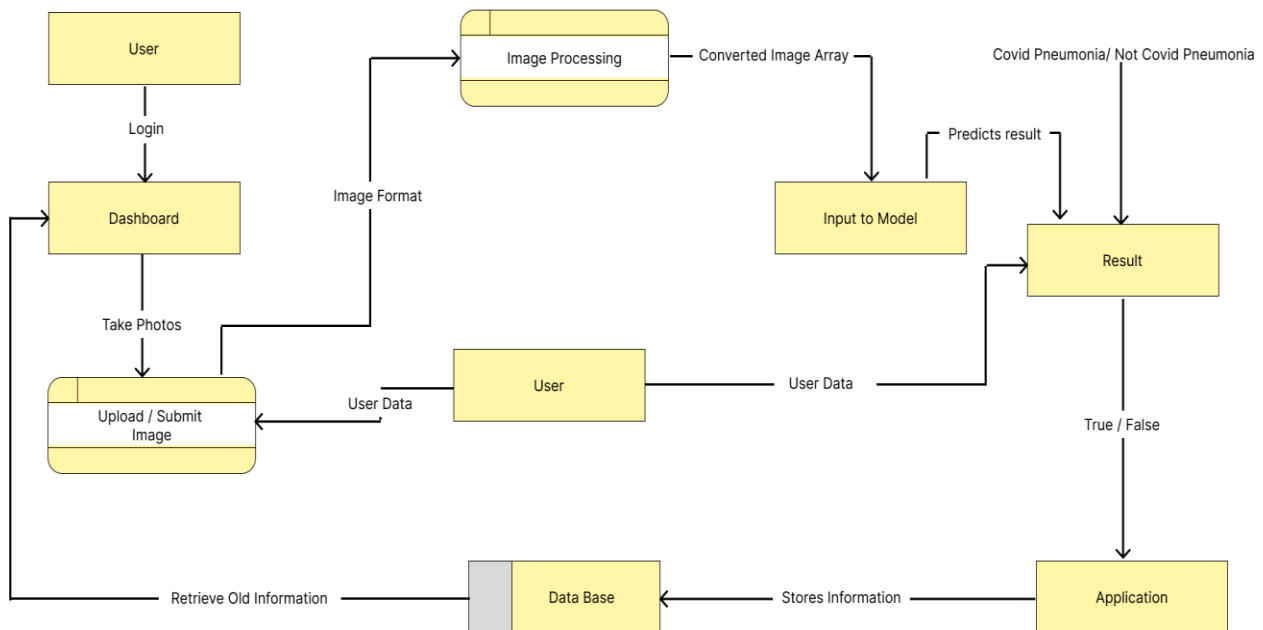


Fig: 4.2

## 4.2 Solution & Technical Architecture

Date	13 May 2023
Team ID	NM2023TMID03654
Project Name	Project - Covid Vision: Advanced COVID-19 Detection from Lung X-Rays with Machine Learning or Deep Learnings

### Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

### Example: Covid Pneumonia and non-Covid classifier

Reference: <https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/>

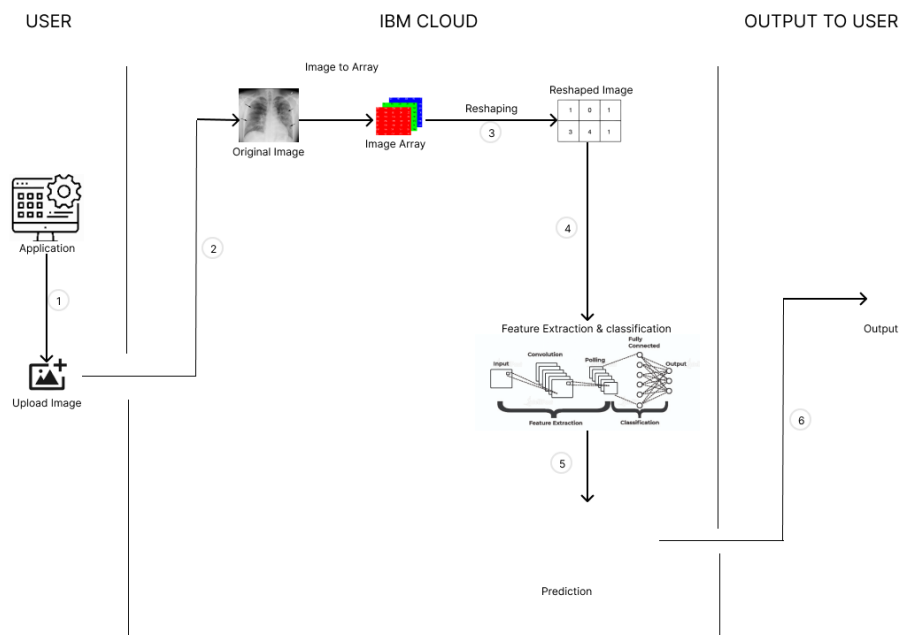


Fig : 4.3

**Table-1 : Components & Technologies:**

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI and Mobile App.	Django and Flutter.
2.	Application Logic-1	How the model is trained and implemented for the application.	Python
3.	Application Logic-2	How the application behaves when user operates the application.	IBM Watson STT service
4.	External API-1	How the data or the image is transferred from the end user to the server for computation purpose.	REST API
5.	Machine Learning Model	The purpose of this machine learning model is to classify the True positives and False negatives of the given image as an input.	Image classification model, pretrained model – Xception, InceptionV2, Resnet and Inception_resnet_v2. Used transfer learning technique, ensemble technique
6.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local / Cloud Server Configuration.	Local, Cloud Foundry, Kubernetes, etc.

**Table-2: Application Characteristics:**

<b>S.No</b>	<b>Characteristics</b>	<b>Description</b>	<b>Technology</b>
1.	Open-Source Frameworks	The open-source frame works are shown here	Python, Django, Flutter, Xception network, InceptionV2 network, Resnet network, Inception_resnet_v2
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Microservices, Kubernetes, Catching
4.	Availability	Justify the availability of application (e.g., use of load balancers, distributed servers etc.)	Distributed servers, Monitoring and alerting, Load Balancing
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	The CNN model needs a big computation power so the model is deployed on a high-performance system, and technologies like catching, redundancy are been used to improve the performance.

### 4.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sam Nijin S
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Haneesh Noyal T R S
	Login	USN-3	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Bei him S S
	Dashboard	USN-4	As a user, I can view my record and details, request for any updating from my dashboard	I can access my dashboard to do all the necessary changes	Low	Haneesh Noyal T R S



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Web User)	Registration	USN-5	As a user I need to have my personal account	I can register for an account and have a personal dashboard	Medium	Gokul B S
		USN-6	As a user I need to upload my X-Rays to get the test result	I can upload the X-Ray images with PNG format and DICOM format	High	Sam Nijin S
		USN-7	Once the image uploaded, what should I do now?	The image will get processed if it is valid else it will show an error message	Medium	Gokul B S
Customer Care Executive	Help Desk	USN-8	The process of helping the customers regarding the issues and support.	The customers can be able to get support and resolve their issues	High	Sam Nijin S

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Administrator	Monitor the application and generate report	USN-9	The admin needs to monitor the application and need to generate the report frequently	The report is being generated very frequently which enables the admin to monitor the application	High	Sam Nijin S
Administrator	Update the Model	USN-10	There may be change of change in the visuals of the covid pneumonia in the chest X-Rays, so the admin needs to update and change the model.	The new model performs well compared to old model and predicts new variant of covid pneumonia	High	Sam Nijin S

## 5. CODING & SOLUTIONING

### 5.1 Feature 1

The use of transfer learning which result in good training of the model. In this project we had used three different pre-trained keras application which includes the following,

- a. Resnet
- b. Xception
- c. Inception

a. Resnet:

The model has been trained with adam optimizer with a learning rate of 1e-6. Which results better the following image shows the model summary.

```
resnet_model = model_creation(base_model=resnet, dropout_value=0.3)
resnet_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50v2 (Functional)	(None, 7, 7, 2048)	23564800
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 256)	25690368
batch_normalization (Batch Normalization)	(None, 256)	1024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
batch_normalization_1 (Batch Normalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
batch_normalization_2 (Batch Normalization)	(None, 64)	256
dense_3 (Dense)	(None, 2)	130

=====  
Total params: 49,298,242  
Trainable params: 25,732,546  
Non-trainable params: 23,565,696  
=====

Fig: 5.1

This image illustrate the architecture of the model that is trained with resnet.

Resnet is a good performing model for the current dataset. Of the X-Ray images of covid pneumonia. Where it performs a great classification of the images and received a fl score of 93%. The accuracy and validation accuracy vs epochs and the loss and validation loss vs epochs plots are illustrated below,

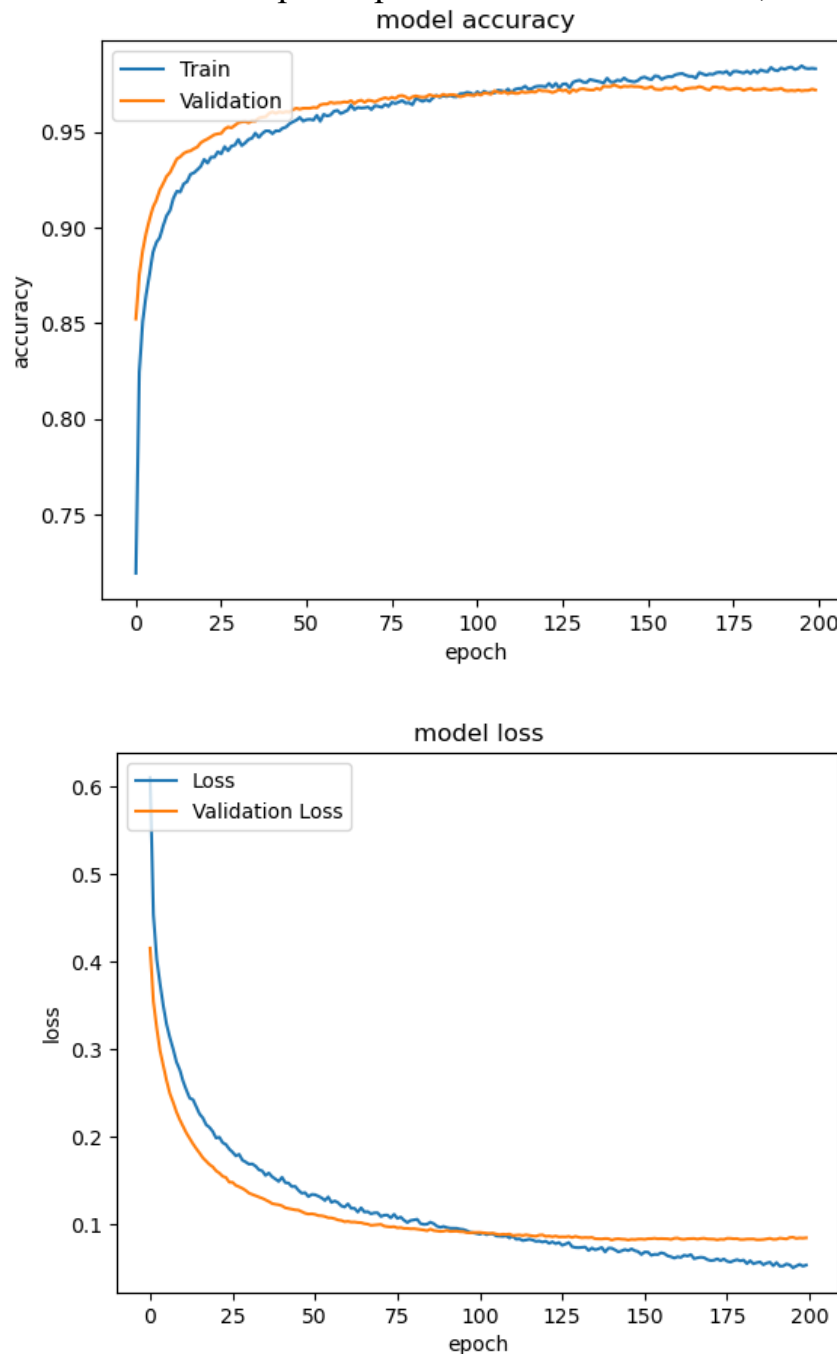


Fig: 5.2

From the above two plots it is clear that the accuracy and loss are inversely proportional to that of the epochs. Which means that when the epoch is increased the accuracy improves and as a result of increased epochs and improved accuracy the loss is been reduced. Which imply the inverse proportion.

## b. Xception:

This model is also trained with the adam optimizer with the same learning rate of 1e-6. This results good in the unseen data but bad in the data used for validation on training. The following image shows the model architecture. This is also one of the good pre-trained model that is used for image classification purposes. And provides an efficient and an optimal accuracy on unseen data. The following image shows the model architecture of the Xception model,

```
xception_model = model_creation(base_model=xception, dropout_value=0.5)
xception_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
xception (Functional)	(None, 7, 7, 2048)	20861480
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 256)	25690368
batch_normalization_4 (Batch Normalization)	(None, 256)	1024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
batch_normalization_5 (Batch Normalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
batch_normalization_6 (Batch Normalization)	(None, 64)	256
dense_3 (Dense)	(None, 2)	130

=====  
Total params: 46,594,922  
Trainable params: 25,732,546  
Non-trainable params: 20,862,376  
=====

Fig: 5.3

This image illustrates the architecture of the model trained with xception.

Xception is a good performing model for this specific dataset of the X-Ray images of covid pneumonia. Where it performs really good on the unseen data and some bad in the validation data that is used for training. It is some what an overfitting condition anyway this model performs well on the unseen data. Even though this should be tuned as sometimes it may misclassify the images which may cause a great bad impact on this project. This model has a f1 score of around 87%. The accuracy and validation accuracy vs epochs and the loss and the validation loss vs epochs plots are illustrated below,

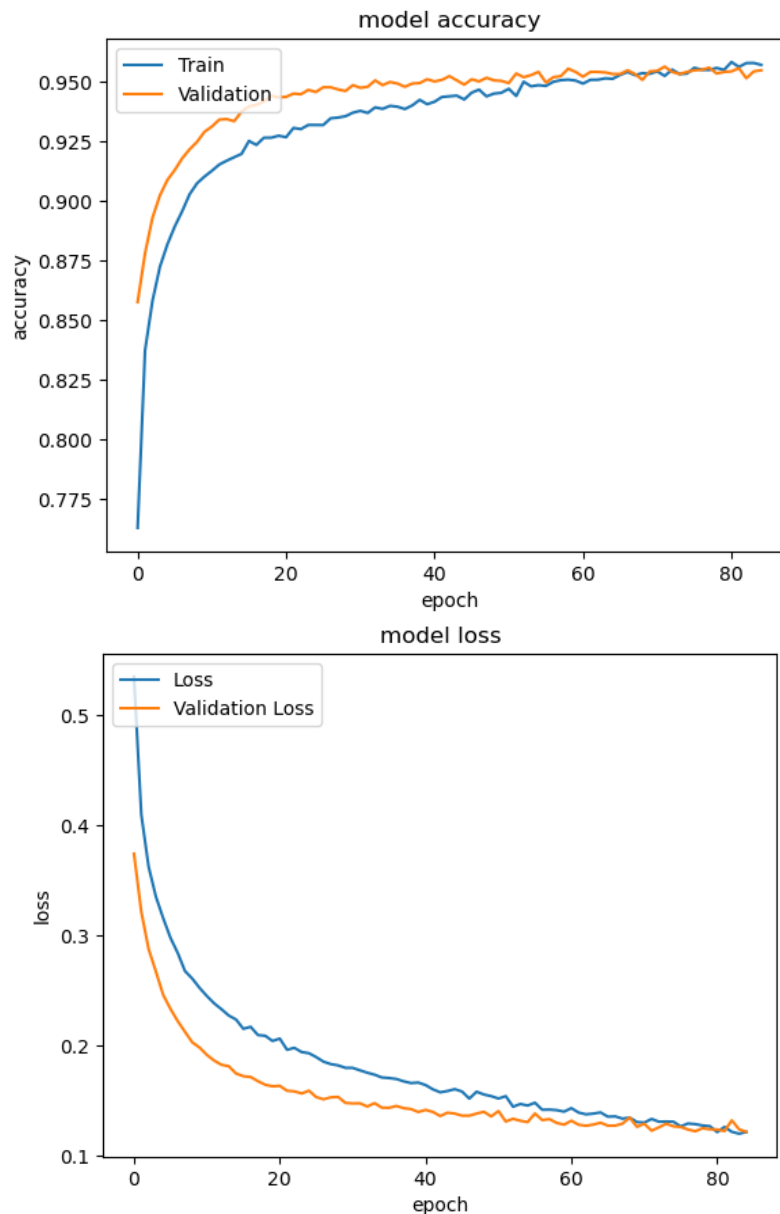


Fig: 5.4

From the above two plots it is clear that the accuracy and loss are inversely proportional to that of the epochs. Which means that when the epoch is increased the accuracy improves and as a result of increased epochs and improved accuracy the loss is been reduced. Which imply the inverse proportion.

### c. Inception

The inception based model was trained with the adam optimizer with the same learning rate of 1e-6. This also performs good in the unseen data but some what bad on the data used for validation in training purposes. The following image shows the configuration of this model.

```
inception_model = model_creation(base_model=inception, dropout_value=0.3)
inception_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
flatten (Flatten)	(None, 51200)	0
dense (Dense)	(None, 256)	13107456
batch_normalization_94 (Batch Normalization)	(None, 256)	1024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
batch_normalization_95 (Batch Normalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
batch_normalization_96 (Batch Normalization)	(None, 64)	256
dense_3 (Dense)	(None, 2)	130
=====		
Total params: 34,953,314		
Trainable params: 13,149,634		
Non-trainable params: 21,803,680		

Fig: 5.5

The above image shows the internal architecture of the model constructed with the inception model as the base model and with 2 fully connected layer with dropout layers and the batch normalization later to normalize the weights.

Inception is a well known pre-trained model for its perfect layers and their interconnections. This is really a very good pre-trained model for the medical images as it provides a very efficient preprocessing function which even results in the better dataset for the model to train with. The model is trained with the ‘imagenet’ weights which is an efficient weight for building a model that classifies the image based on the features. The following image shows the accuracy and validation accuracy vs epochs and loss and validation loss vs epochs plots as follows,

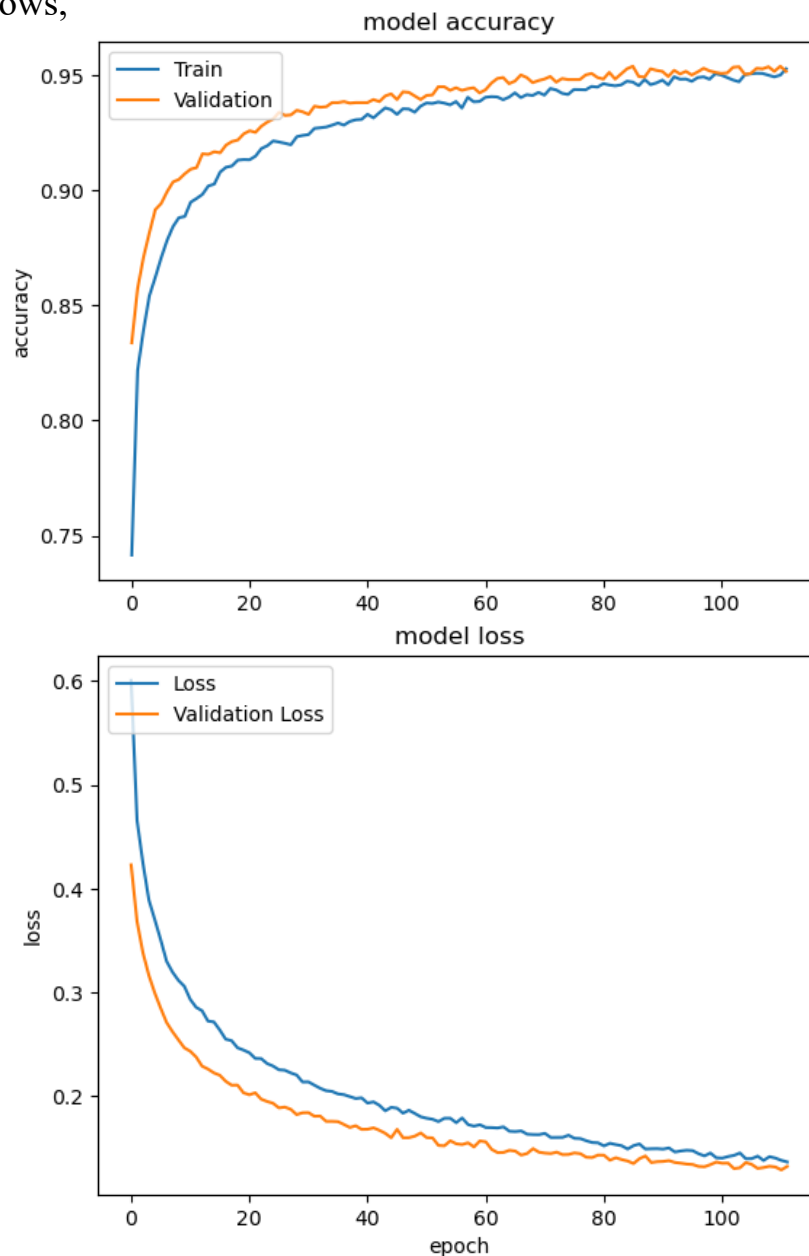


Fig: 5.6

From the above two plots it is clear that the accuracy and loss are inversely proportional to that of the epochs. Which means that when the epoch is increased the accuracy improves and as a result of increased epochs and improved accuracy the loss is been reduced. Which imply the inverse proportion.



## 5.2 Feature 2

This project is made with the ensemble technique which means that the dataset will be trained with multiple models and the result is taken with mode is so that the accuracy and the highest performance will be retrieved. This enables the better performance of the model. So, the model can be used for the real time purposes with high efficiency. The following figure shows the ensembling technique which is carried out in this project for the prediction purpose,

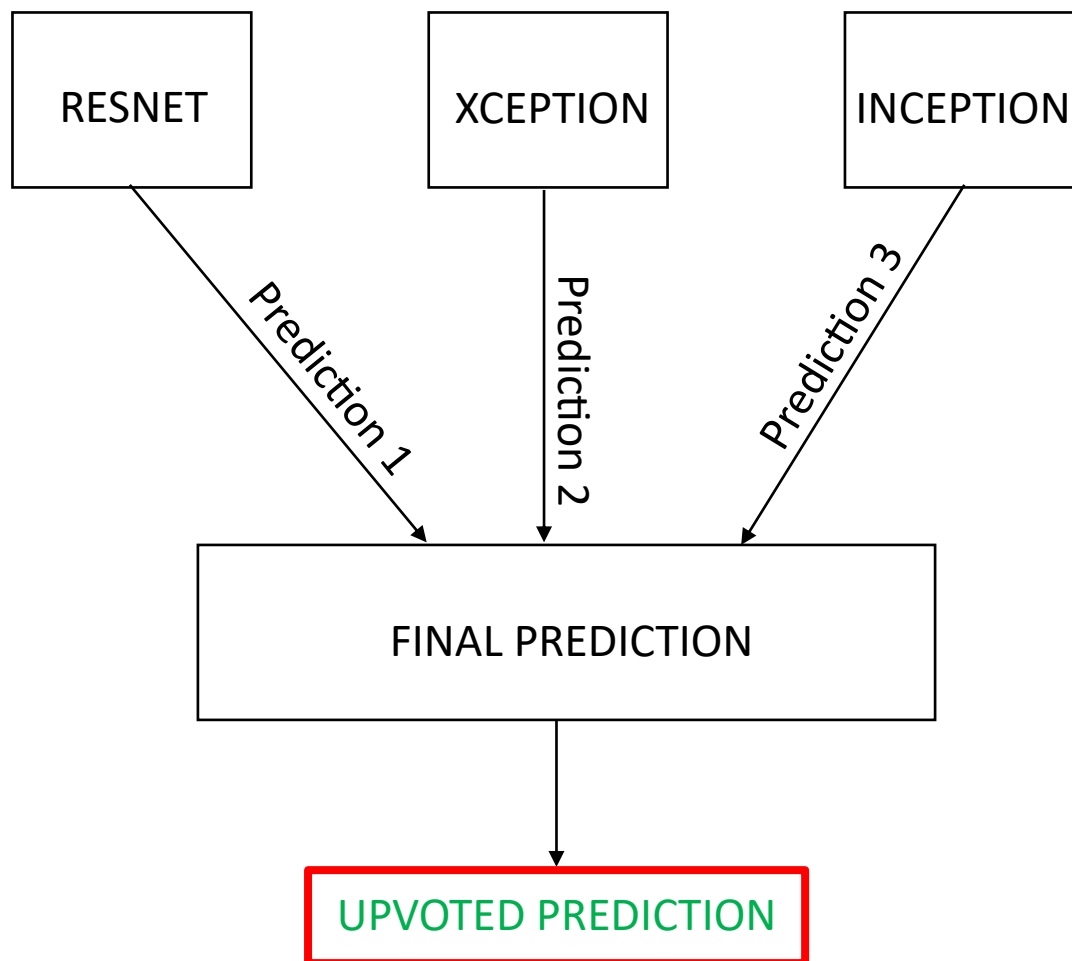


Fig: 5.7

As shown in the above figure the prediction is done from the prediction of the three models and the final prediction is made from the previous three predictions. This improves the accuracy in this. The upvoting technique which is taking the mode of the old prediction is made. Which means that the result of the 2 models will be the final result. This states that eve if a model performs wrongly then, the final prediction won't be affected. This improves the performance of the model and improves the reliability score of the model which result in using the model in real time can be more efficient and much optimal way.

## 6. RESULTS

### 6.1 Model Performance Test

#### Model Performance Testing:

Project team shall fill the following information in the model performance testing template.

S.No	Parameter	Values	Screenshot
1.	Model Summary	-	<a href="https://drive.google.com/drive/folders/1-oP25kRFXFS5pHbN78n7c-s24TzhCK0Z?usp=sharing">https://drive.google.com/drive/folders/1-oP25kRFXFS5pHbN78n7c-s24TzhCK0Z?usp=sharing</a>
2.	Accuracy	Training Accuracy – 98.29  Validation Accuracy - 97.18	<a href="https://drive.google.com/drive/folders/1xx0-6wi2nsqAMPIzhSmb9x7j6-8eSbDa?usp=sharing">https://drive.google.com/drive/folders/1xx0-6wi2nsqAMPIzhSmb9x7j6-8eSbDa?usp=sharing</a>
3.	Confidence Score (Only Yolo Projects)	Class Detected -  Confidence Score -	

## **7. ADVANTAGES & DISADVANTAGES**

### **7.1 Advantages:**

- The major advantage of this project is the time efficiency of the people.
- This enables to test the covid-19 pneumonia very quickly.
- So it boost the mental health of the people.
- No need of waiting just for the result.
- No need to be in quarantine for a long time while waiting for the result.

### **7.2 Disadvantages:**

- More computation power is needed.
- Costlier in implementing as it needs high computational power.
- Very difficult to train ideal model.

## 8. CONCLUSION

This project is very useful to predict between the normal and the covid-19 pneumonia affected lung x-ray. This is deployed in the Django framework as well as in the flutter framework for the app to be dedicatedly make for the mobile phones. This makes the patient to not visit the hospital and hence the crowd in the hospital will get reduced. This ensure that due to this waiting for the result. Because covid-19 is a deadliest pandemic which can spread through the air medium itself. This project got an accuracy for more than 97% for the 3 models and hence the model is finally predicted with the mode of the three previously predicted value and hence the accuracy, precision, recall, fl score, auc and roc metrics will perform better. This seems to be great model while testing with the real time x-ray images.

The models used in this project are the resnet pre-trained model, xception pre-trained model, inception pre-trained model. Both the three models are performing good in the unseen data. And even though a single model fails to predict correctly the rest of the result will be taken as the final report as the mode value of the three predicted value is been used.

Finally, this project will have a great impact on the medical field for detecting or diagnosing the covid-19 pneumonia using the lung x-rays. And this has very good performance in the real time data so that it will predict around 98% correctly.

## **9. FUTURE SCOPE**

The model needs to be tuned some more to result in high accuracy and precision. So that it can have the ability to classify the lung x-ray of covid-19 pneumonia properly. Also, the ensembling technique needed to be improved with 5 models and hence even though the 2 models fail to perform well the other 3 model can be predicted very properly. This improves the reliability of the application. Due to this higher tuning, we need huge computation power. This will cost very high to perform the implementation. Even the prediction need not to have very high computation power as much needed to implement.

In future some more new models will be added and replaced the current model which are used to improve the accuracy and performance of the classification. And the metrics will also be improved.

## 10.APPENDIX

### Source Code

```
# -*- coding: utf-8 -*-
```

```
"""Covid Pneumonia Classifier.ipynb
```

Automatically generated by Colaboratory.

Original file is located at

[https://colab.research.google.com/drive/1rZW6\\_mLUrDSeThMyoQuDcjuOYT  
PFX0kX](https://colab.research.google.com/drive/1rZW6_mLUrDSeThMyoQuDcjuOYT<br/>PFX0kX)

```
"""
```

```
from tensorflow.keras.applications.inception_v3 import preprocess_input as  
incep_preprocess
```

```
from tensorflow.keras.applications.resnet_v2 import preprocess_input as  
res_preprocess
```

```
from tensorflow.keras.applications.xception import preprocess_input as  
xcep_preprocess
```

```
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization,  
Flatten
```

```
from tensorflow.keras.applications import ResNet50V2, Xception, InceptionV3
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import CSVLogger, TerminateOnNaN

from tensorflow.keras.metrics import Precision, Recall

from sklearn.model_selection import train_test_split

from tensorflow.keras.backend import clear_session

from sklearn.metrics import classification_report

from tensorflow.keras.optimizers import Adam

from sklearn.metrics import confusion_matrix

from tensorflow.keras import Sequential

from matplotlib import pyplot as plt

import tensorflow as tf

import seaborn as sns

import pandas as pd

import numpy as np

import warnings

# Suppress the warning

warnings.filterwarnings("ignore")

warnings.resetwarnings()

def show_image_samples(gen):

    class_dict=gen.class_indices
```

```

class_names=list( class_dict.keys())

images,labels=next(gen) # get a sample batch from the generator

plt.figure(figsize=(20, 20))

length=len(labels)

if length<26: #show maximum of 25 images

    r=length

else:

    r=25

for i in range(r):

    plt.subplot(5, 5, i + 1)

    image=(images[i]+1 )/2 # scale images between 0 and 1 becaue pre-
processor set them between -1 and +1

    plt.imshow(image)

    index=np.argmax(labels[i])

    class_name=class_names[index]

    plt.title(class_name, color='blue', fontsize=16)

    plt.axis('off')

plt.show()

def acc_loss_plot(model_history):

    # summarize history for accuracy

    plt.plot(model_history.history['accuracy'])

```



```

plt.plot(model_history.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

# summarize history for loss

plt.plot(model_history.history['loss'])

plt.plot(model_history.history['val_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['Loss', 'Validation Loss'], loc='upper left')

plt.show()

```

```

gpus = tf.config.list_physical_devices('GPU')

if gpus:

    try:

        # Currently, memory growth needs to be the same across GPUs

        for gpu in gpus:

            tf.config.experimental.set_memory_growth(gpu, True)

    logical_gpus = tf.config.list_logical_devices('GPU')

```

```

print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")

except RuntimeError as e:

    # Memory growth must be set before GPUs have been initialized

    print(e)


TRAIN_IMAGE_PATH = 'dataset/train/'

EVALUATE_IMAGE_PATH = 'dataset/eval/'

TRAIN_ANNOT_PATH = 'annotation/train.txt'

EVALUATE_ANNOT_PATH = 'annotation/test.txt'


train_annot = pd.read_csv(TRAIN_ANNOT_PATH, sep=" ", header=None)

train_annot.columns=['patient id', 'file_paths', 'labels', 'data source']

train_annot.drop(['patient id', 'data source'], axis=1, inplace=True)

train_annot.head()


eval_annot = pd.read_csv(EVALUATE_ANNOT_PATH, sep=" ",
header=None)

eval_annot.columns=['patient id', 'file_paths', 'labels', 'data source']

eval_annot.drop(['patient id', 'data source'], axis=1, inplace=True)

eval_annot.head()


train_annot['labels'].value_counts()

```

```
eval_annot['labels'].value_counts()
```

```
train_df, val_df = train_test_split(train_annot, test_size=0.15, random_state=5,  
stratify=train_annot['labels'], shuffle=True)
```

```
print(f'*****\n Train Data
```

```
\n*****\n{train_df["labels"].value_counts()}')
```

```
print(f'\n*****\n Validation Data
```

```
\n*****\n{val_df["labels"].value_counts()}')
```

```
print(f'\n*****\n Evaluating Data
```

```
\n*****\n{eval_annot["labels"].value_counts()}\n')
```

```
IMAGE_SIZE=(224,224)
```

```
BATCH_SIZE=64
```

```
train_df.head(15)
```

```
train_datag_res = ImageDataGenerator(  
    preprocessing_function=res_preprocess,
```

```
    horizontal_flip=True,
```

```
    zoom_range=0.2,
```

```
    shear_range=0.2,
```

)

```
val_datag_res = ImageDataGenerator(  
    preprocessing_function=res_preprocess,  
)
```

```
train_gen_res = train_datag_res.flow_from_dataframe(  
    dataframe=train_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
)
```

```
val_gen_res = val_datag_res.flow_from_dataframe(  
    dataframe=val_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',
```

```
y_col='labels',  
target_size=IMAGE_SIZE,  
batch_size=BATCH_SIZE,  
class_mode='categorical',  
color_mode='rgb',  
shuffle=False,  
)
```

```
eval_gen = val_datag_res.flow_from_dataframe(  
    dataframe=eval_annot,  
    directory=EVALUATE_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
    shuffle=False,  
)
```

```
eval_gen.class_indices
```

```
show_image_samples(train_gen_res)
```

```
show_image_samples(val_gen_res)
```

```
show_image_samples(eval_gen)
```

```
def model_creation(base_model, dropout_value):
```

```
    model = Sequential()
```

```
    model.add(layer=base_model)
```

```
    model.add(Flatten())
```

```
    model.add(layer=Dense(units=256, activation='relu'))
```

```
    model.add(BatchNormalization())
```

```
    model.add(Dropout(dropout_value))
```

```
    model.add(layer=Dense(units=128, activation='relu'))
```

```
    model.add(BatchNormalization())
```

```
    model.add(Dropout(dropout_value))
```

```

model.add(layer=Dense(units=64, activation='relu'))

model.add(BatchNormalization())


model.add(layer=Dense(units=2, activation='sigmoid'),)


return model


resnet = ResNet50V2(

    include_top=False,

    input_shape=(224,224,3),

    weights='imagenet',

)

for layer in resnet.layers:

    layer.trainable = False


resnet_model = model_creation(base_model=resnet, dropout_value=0.3)

resnet_model.summary()


# defining the optimizer


adam_optimizer = Adam(learning_rate = 1e-6)

```

```
# compiling the model
```

```
resnet_model.compile(loss="binary_crossentropy",optimizer=adam_optimizer,  
metrics=["accuracy", Precision(), Recall()])
```

```
class StopOnPoint(tf.keras.callbacks.Callback):
```

```
    def __init__(self, acc_threshold, val_acc_threshold, loss_threshold,  
val_loss_threshold):
```

```
        super(StopOnPoint, self).__init__()
```

```
        self.acc_threshold = acc_threshold
```

```
        self.val_acc_threshold = val_acc_threshold
```

```
        self.loss_threshold = loss_threshold
```

```
        self.val_loss_threshold = val_loss_threshold
```

```
    def on_epoch_end(self, epoch, logs=None):
```

```
        accuracy = logs['accuracy']
```

```
        val_accuracy = logs['val_accuracy']
```

```
        loss = logs['loss']
```

```
        val_loss = logs['val_loss']
```

```
        if accuracy >= self.acc_threshold:
```

```
            if val_accuracy >= self.val_acc_threshold:
```

```
                if loss <= self.loss_threshold:
```



```

        if val_loss <= self.val_loss_threshold:

            self.model.stop_training = True

    if accuracy > val_accuracy:

        self.model.stop_training = True

log_filepath = 'log/resnet_training.csv'

csv_logger = CSVLogger(filename=log_filepath, separator=',', append=True)

terminate_on_nan = TerminateOnNaN()

callbacks = [csv_logger, terminate_on_nan, StopOnPoint(acc_threshold=0.97,
val_acc_threshold=0.97, loss_threshold=0.05, val_loss_threshold=0.05)]

# Commented out IPython magic to ensure Python compatibility.

# %%time

# resnet_history = resnet_model.fit(train_gen_res, epochs=200,
validation_data=val_gen_res, callbacks=callbacks,)

acc_loss_plot(model_history=resnet_history)

```

```
train_gen_res.class_indices
```

```
val_pred=resnet_model.predict(val_gen_res)
```

```
val_df["predict_resnet"]=np.argmax(val_pred,axis=1)
```

```
val_df["predict_resnet_class"]=val_df["predict_resnet"].map({0:"negative",1:"positive"})
```

```
val_df.head()
```

```
val_df['predict_resnet_class'].value_counts()
```

```
print(classification_report(y_pred=val_df['predict_resnet_class'],  
y_true=val_df['labels']))
```

```
# confusion matrix for validation data
```

```
sns.heatmap(confusion_matrix(y_pred=val_df['predict_resnet_class'],  
y_true=val_df['labels']), annot=True,
```

```
xticklabels=val_df['labels'].unique(),  
yticklabels=val_df['labels'].unique(), cmap='Greens', fmt="")
```

```
plt.show()
```

```
eval_annot
```

```
eval_pred=resnet_model.predict(eval_gen)
```

```
eval_annot["predict_resnet"]=np.argmax(eval_pred,axis=1)
```

```
eval_annot["predict_resnet_class"]=eval_annot["predict_resnet"].map({0:"negative",1:"positive"})
```

```
print(classification_report(y_pred=eval_annot['predict_resnet_class'],  
y_true=eval_annot['labels']))
```

```
# confusion matrix for evaluation data data
```

```
sns.heatmap(confusion_matrix(y_pred=eval_annot["predict_resnet_class"],  
y_true=eval_annot['labels']), annot=True,
```

```
xticklabels=eval_annot['labels'].unique(),  
yticklabels=eval_annot['labels'].unique(), cmap='Greens', fmt="")  
plt.show()
```

```
resnet_model.save('saved model/resnet_model.h5')
```

```
## model 2
```

```
train_datag_xcep = ImageDataGenerator(  
    preprocessing_function=xcep_preprocess,  
    horizontal_flip=True,  
    zoom_range=0.2,  
    shear_range=0.2,  
)
```

```
val_datag_xcep = ImageDataGenerator(  
    preprocessing_function=xcep_preprocess  
)
```

```
train_gen_xcep = train_datag_xcep.flow_from_dataframe(  
    dataframe=train_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
)
```

```
val_gen_xcep = val_datag_xcep.flow_from_dataframe(  
    dataframe=val_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
)
```

```
show_image_samples(train_gen_xcep)
```

```
show_image_samples(val_gen_xcep)
```

```
xception = Xception(  
    include_top=False,  
    input_shape=(224,224,3),  
    weights='imagenet',  
)
```

```

for layer in xception.layers:

    layer.trainable = False


xception_model = model_creation(base_model=xception, dropout_value=0.3)

xception_model.summary()


log_filepath = 'log/xception_training.csv'

csv_logger = CSVLogger(filename=log_filepath, separator=',', append=True)


terminate_on_nan = TerminateOnNaN()


callbacks = [csv_logger, terminate_on_nan, StopOnPoint(acc_threshold=0.97,
val_acc_threshold=0.97, loss_threshold=0.05, val_loss_threshold=0.05)]


xception_model.compile(loss="binary_crossentropy",optimizer=adam_optimizer, metrics=["accuracy", Precision(), Recall()])


# Commented out IPython magic to ensure Python compatibility.

# %%time

```

```

# xception_history = xception_model.fit(train_gen_xcep, epochs=85,
validation_data=val_gen_xcep, callbacks=callbacks,)

acc_loss_plot(model_history=xception_history)

val_pred=xception_model.predict(val_gen_xcep)

val_df["predict_xcep"]=np.argmax(val_pred,axis=1)

val_df["predict_xcep_class"]=val_df["predict_xcep"].map({0:"negative",1:"positive"})

val_df.head()

val_df['predict_xcep_class'].value_counts()

print(classification_report(y_pred=val_df['predict_xcep_class'],
y_true=val_df['labels']))

# confusion matrix for validation data

sns.heatmap(confusion_matrix(y_pred=val_df['predict_xcep_class'],
y_true=val_df['labels']), annot=True,

```

```

        xticklabels=val_df['labels'].unique(),
        yticklabels=val_df['labels'].unique(), cmap='Greens', fmt=")

plt.show()

eval_annot

eval_pred=xception_model.predict(eval_gen)

eval_annot["predict_xcep"]=np.argmax(eval_pred,axis=1)

eval_annot["predict_xcep_class"]=eval_annot["predict_xcep"].map({0:"negative",1:"positive"})

print(classification_report(y_pred=eval_annot['predict_xcep_class'],
y_true=eval_annot['labels']))

# confusion matrix for evaluation data data

sns.heatmap(confusion_matrix(y_pred=eval_annot["predict_xcep_class"],
y_true=eval_annot['labels']), annot=True,

            xticklabels=eval_annot['labels'].unique(),
            yticklabels=eval_annot['labels'].unique(), cmap='Greens', fmt=")

plt.show()

```



```
xception_model.save('saved model/xception_model.h5')
```

```
## Model 3
```

```
train_datag_incep = ImageDataGenerator(  
    preprocessing_function=incep_preprocess,  
    horizontal_flip=True,  
    zoom_range=0.2,  
    shear_range=0.2,  
)
```

```
val_datag_incep = ImageDataGenerator(  
    preprocessing_function=incep_preprocess  
)
```

```
train_gen_incep = train_datag_incep.flow_from_dataframe(  
    dataframe=train_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,
```

```
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
)
```

```
val_gen_incep = val_datag_incep.flow_from_dataframe(  
    dataframe=val_df,  
    directory=TRAIN_IMAGE_PATH,  
    x_col='file_paths',  
    y_col='labels',  
    target_size=IMAGE_SIZE,  
    batch_size=BATCH_SIZE,  
    class_mode='categorical',  
    color_mode='rgb',  
)
```

```
show_image_samples(train_gen_incep)
```

```
show_image_samples(val_gen_incep)
```

```
inception = InceptionV3(  

```

```
include_top=False,  
input_shape=(224,224,3),  
weights='imagenet',  
)
```

```
for layer in inception.layers:
```

```
    layer.trainable = False
```

```
inception_model = model_creation(base_model=inception, dropout_value=0.3)
```

```
inception_model.summary()
```

```
log_filepath = 'log/inception_training.csv'
```

```
csv_logger = CSVLogger(filename=log_filepath, separator=',', append=True)
```

```
terminate_on_nan = TerminateOnNaN()
```

```
callbacks = [csv_logger, terminate_on_nan, StopOnPoint(acc_threshold=0.97,  
val_acc_threshold=0.97, loss_threshold=0.05, val_loss_threshold=0.05)]
```

```
inception_model.compile(loss="binary_crossentropy",optimizer=adam_optimizer,  
metrics=["accuracy", Precision(), Recall()])
```

```

# Commented out IPython magic to ensure Python compatibility.

# %%time

# inception_history = inception_model.fit(train_gen_incep, epochs=200,
validation_data=val_gen_incep, callbacks=callbacks,)

acc_loss_plot(model_history=inception_history)

val_pred=inception_model.predict(val_gen_incep)

val_df["predict_incep"]=np.argmax(val_pred,axis=1)

val_df["predict_incep_class"]=val_df["predict_incep"].map({0:"negative",1:"po
sitive"})

val_df.head()

val_df['predict_incep_class'].value_counts()

print(classification_report(y_pred=val_df['predict_incep_class'],
y_true=val_df['labels']))

# confusion matrix for validation data

```

```
sns.heatmap(confusion_matrix(y_pred=val_df['predict_incep_class'],
y_true=val_df['labels']), annot=True,

            xticklabels=val_df['labels'].unique(),
            yticklabels=val_df['labels'].unique(), cmap='Greens', fmt=")

plt.show()
```

```
eval_annot
```

```
eval_pred=inception_model.predict(eval_gen)
```

```
eval_annot["predict_incep"]=np.argmax(eval_pred,axis=1)
```

```
eval_annot["predict_incep_class"]=eval_annot["predict_incep"].map({0:"negative",1:"positive"})
```

```
print(classification_report(y_pred=eval_annot['predict_incep_class'],
y_true=eval_annot['labels']))
```

```
# confusion matrix for evaluation data data
```

```
sns.heatmap(confusion_matrix(y_pred=eval_annot["predict_incep_class"],
y_true=eval_annot['labels']), annot=True,
```

```
xticklabels=eval_annot['labels'].unique(),
yticklabels=eval_annot['labels'].unique(), cmap='Greens', fmt=")

plt.show()
```

```
inception_model.save('saved model/inception_model.h5')
```

```
RESNET_MODEL_PATH = 'saved model/resnet_model.h5'
```

```
XCEPTION_MODEL_PATH = 'saved model/xception_model.h5'
```

```
INCEPTION_MODEL_PATH = 'saved model/inception_model.h5'
```

```
import tensorflow as tf
```

```
# Load the h5 model
```

```
resnet = tf.keras.models.load_model(RESNET_MODEL_PATH)
```

```
xception = tf.keras.models.load_model(XCEPTION_MODEL_PATH)
```

```
inception = tf.keras.models.load_model(INCEPTION_MODEL_PATH)
```

```
# Convert the model to TFLite
```

```
converter = tf.lite.TFLiteConverter.from_keras_model(resnet)
```

```
tflite_model_resnet = converter.convert()
```

```
tflite_model_xception = converter.convert()

tflite_model_inception = converter.convert()
```

```
# Save the TFLite model to a file
```

```
with open('./converted model/resnet.tflite', 'wb') as f:
```

```
    f.write(tflite_model_resnet)
```

```
# Save the TFLite model to a file
```

```
with open('./converted model/xception.tflite', 'wb') as f:
```

```
    f.write(tflite_model_xception)
```

```
# Save the TFLite model to a file
```

```
with open('./converted model/inception.tflite', 'wb') as f:
```

```
    f.write(tflite_model_inception)
```

## TESTING

```
sample = eval_annot.sample()
```

```
sample
```

```
import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

from tensorflow.keras.models import load_model

path = EVALUATE_IMAGE_PATH + str(sample['file_paths']).split()[1]

image = cv.imread(path)

img = cv.resize(image, (224, 224))

img = np.reshape(img, [1, 224, 224, 3])

res_image = res_preprocess(img)

xcep_image = xcep_preprocess(img)

incep_image = incep_preprocess(img)


plt.title(str(sample['labels']).split()[1])


plt.imshow(image)

plt.show()


resnet = load_model(RESNET_MODEL_PATH, compile=False)

xception = load_model(XCEPTION_MODEL_PATH, compile=False)

inception = load_model(INCEPTION_MODEL_PATH, compile=False)
```



```
resnet_pred = resnet.predict(res_image)

xception_pred = xception.predict(xcep_image)

inception_pred = inception.predict(incep_image)


classes = []


classes.append(int(np.argmax(resnet_pred, axis=1)))

classes.append(int(np.argmax(xception_pred, axis=1)))

classes.append(int(np.argmax(inception_pred, axis=1)))


labels = {0 : 'negative', 1 : 'positive'}

print(classes)


print(labels[max(set(classes), key=classes.count)])
```

**GITHUB LINK:**

<https://github.com/naanmudhalvan-SI/IBM--13630-1682660810>

**DEMONSTRATION VIDEO LINK:**

<https://youtu.be/i3TO5DQwzvK>

**DATASET LINK:**

<https://www.kaggle.com/datasets/andyczao/covidx-cxr2>

**MODEL LINK:**

<https://drive.google.com/drive/folders/1gMNpZ3ljV9gEBIGW2pIm27K2zgR4kyPc?usp=sharing>

**MODEL SUMMARY LINK:**

<https://drive.google.com/drive/folders/1-oP25kRFXFS5pHbN78n7c-s24TzhCK0Z?usp=sharing>

**MODEL ACCURACY LINK:**

<https://drive.google.com/drive/folders/1xx0-6wi2nsqAMPlzhSmb9x7j6-8eSbDa?usp=sharing>

**CLASSIFICATION REPORT LINK:**

<https://drive.google.com/drive/folders/1Y9N5zd5m5CACI5mKjSczrBp7s3tTcEQz?usp=sharing>

**CONFUSION MATRIX HEATMAP:**

[https://drive.google.com/drive/folders/1FOh12kqlbkib\\_CesxQ\\_ygLnTi-92HF5f?usp=sharing](https://drive.google.com/drive/folders/1FOh12kqlbkib_CesxQ_ygLnTi-92HF5f?usp=sharing)

### **TRAINING LOG:**

[https://drive.google.com/drive/folders/1-RVplqiKm-EapIkJyZopkin1\\_UQjRGRI?usp=sharing](https://drive.google.com/drive/folders/1-RVplqiKm-EapIkJyZopkin1_UQjRGRI?usp=sharing)

### **ACCURACY LOSS PLOT:**

<https://drive.google.com/drive/folders/1FCUKqR64ADu70cmFO5zCIBrmPpA9S6xz?usp=sharing>

### **FLUTTER APK LINK:**

<https://drive.google.com/drive/folders/1JjdAnf8djOJ-aA2pEYi5CsfP6FjKL57h?usp=sharing>