GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

PROJECT REPORT

ON

‘A Machine Learning Approach to detect COVID19 using X-RAY Detection’

SUBMITTED BY-
NAME – RAHUL KUMAR
COURSE - MCA (3rd Semester)
ROLL NO. - 1102286

SUBMITTED TO-
Prof. Anuj Singh Rawat

ACKNOWLEDGEMENT

In completing my project on ‘A Machine Learning Approach to understanding COVID19 detection’, I would like to convey special gratitude to my Project mentor Anuj Singh Rawat for his valuable suggestion, guidance, and support in completing this project.

Secondly, I would like to thank my teachers and friends for being supportive and helping in completion of the project.
It helped me increase my knowledge and skills.

Rahul kumar
1102286

CONTENT

INTRODUCTION

Within the last decade, deep learning techniques have become much more readily available for the masses to use for research, enterprise and personal use. In particular, applications to the medical field are becoming much more widespread as this technology can greatly assist with radiology.

The unfortunate outbreak of Covid-19 has also let to a radical shift towards improving medical technologies and processes. The aim of this project, ableit educational, is to build a simple model which can differentiate between a healthy (normal) lung x-ray and an x-ray of a lung infected with Covid-19.

To re-iterate, this project is intended for educational purposes only, and not to be taken as a vetted model for Covid-19 assessments.

Python is a widely-used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

There are two major Python versions: Python 2 and Python 3. Both are quite different.

Programming Language: -

Python

Data Source

One of the main benefits of the open-source community is the ability to find readily available datasets for which to work with.

Two main datasets were used for this project:

- Covid-19 infected lung images were taken from: https://github.com/ieee8023/covid-chestxray-dataset
- Healthly lung images were taken from: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

The images were downloaded and preprocessed prior to training the model. For example, the dataset contains xray images of patients diagnosed with illnesses other that Covid-19, and we filtered them out. In addition, only PA (Postero-Anterior) images, or frontal images were used for model training.

The standard 80/20 split for training and testing data was applied. After filtering throught the data, we were left with 180 Covid xray images to use. The same number of healthy images were used.

The final breakdown of images used for the model:

- Training: 144 images for each category (Covid & Normal)
- Testing: 36 images for each category (Covid & Normal

Python 3.7.3 is the latest version.

The two of the most used versions has to Python 2.x & 3.x. There is a lot of competition between the two and both of them seem to have quite a number of different fanbase.

For various purposes such as developing, scripting, generation and software testing, this language is utilized. Due to its elegance and simplicity, top technology organizations like Dropbox, Google, Quora, Mozilla, Hewlett-Packard, Qualcomm, IBM, and Cisco have implemented Python.

Python has come a long way to become the most popular coding language in the world. Python has just turned 30, but it still has that unknown charm & X factor which can be clearly seen from the fact that Google users have consistently searched for Python much more than they have searched for Kim Kardashian, Donald Trump, Tom Cruise etc.

Python has been an inspiration for many other coding languages such as Ruby, Cobra, Boo, CoffeeScript ECMAScript, Groovy, Swift Go, OCaml, Julia etc.

LIBRARIES: -

Pandas

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Features of Pandas

>Time series analysis

Time Series / Date functionality (Official Pandas Documentation)
Times series analysis with pandas (EarthPy)
Timeseries with pandas (Jupyter)
Complete guide to create a Time Series Forecast (with Codes in Python) (Analytics Vidhya)

>split-apply-combine

Split-apply-combine is a common strategy used during analysis to summarize data—you split data into logical subgroups, apply some function to each subgroup, and stick the results back together again. In pandas, this is accomplished using the groupby() function and whatever functions you want to apply to the subgroups.

Group By: split-apply-combine (Official Pandas Documentation)
Summarizing Data in Python with Pandas (Brian Connelly)
Using Pandas: Split-Apply-Combine (Duke University)

>Data visualization

Visualization (Official Pandas Documentation)
Simple Graphing with IPython and Pandas (Chris Moffitt)
Beautiful Plots With Pandas and Matplotlib (The Data Science Lab)

>Pivot tables

Reshaping and Pivot Tables (Official Pandas Documentation)
Pandas Pivot Table Explained (Chris Moffitt)
Pivot Tables in Python (O'Reilly)

>Working with missing data

Working with missing data (Official Pandas Documentation)
Handling missing data (O'Reilly)

Uses of Pandas

Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including:

Data cleansing
Data fill
Data normalization
Merges and joins
Data visualization
Statistical analysis
Data inspection
Loading and saving data
And much more

In fact, with Pandas, you can do everything that makes world-leading data scientists vote Pandas as the best data analysis and manipulation tool available.

INSTALLATION: -

pip install pandas

pip install Sklearn

SKLEARN.py: -

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Installation: -
pip install -U scikit-learn

MODELS

Confusion Matrix: -

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Classification Report: -

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True Negatives, and False Negatives are used to predict the metrics of a classification report as shown below.

Support Vector Classification: -

SVM is a supervised machine learning algorithm that can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs

KNN: -

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on the Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most like the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a good suite category by using K- NN algorithm.

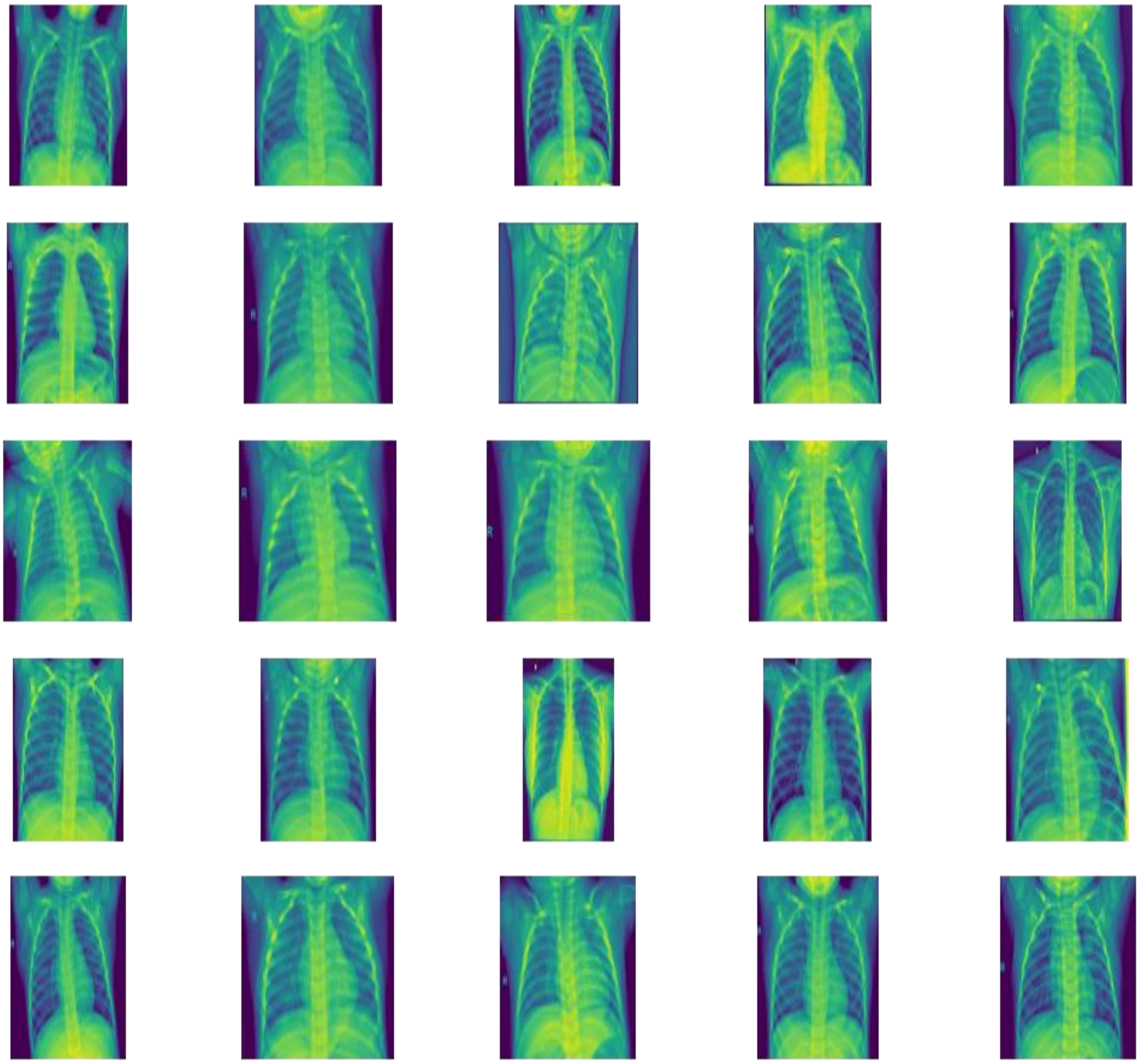
K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumptions on underlying data.

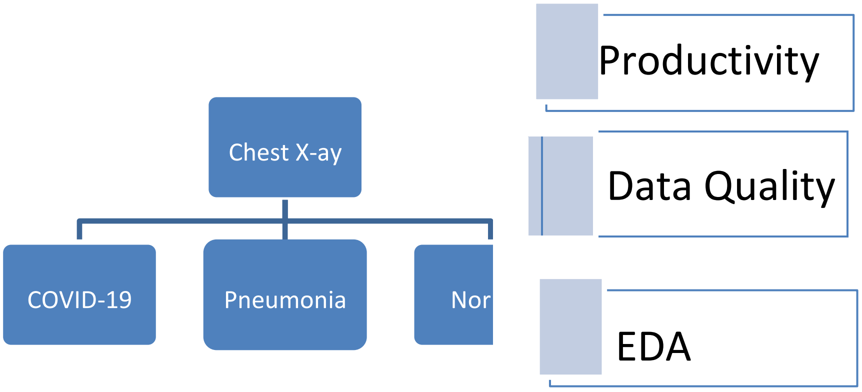
It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much like the new data.

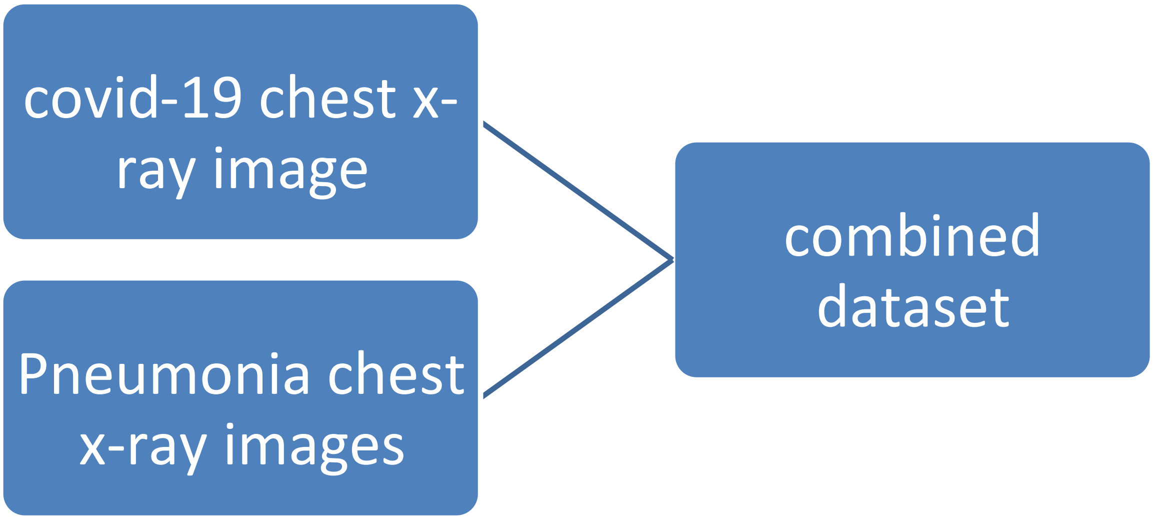
Normal X-Ray Images

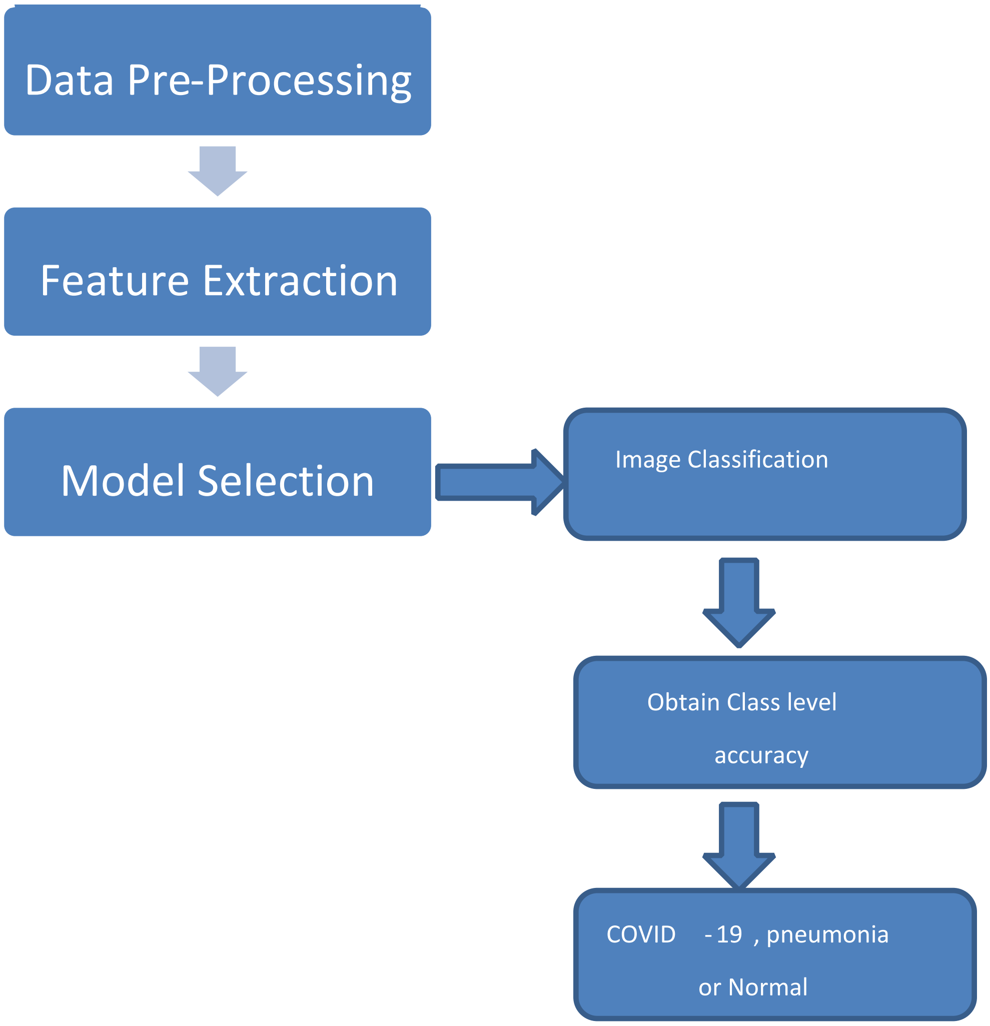


Classifying the chest x-rays



Chest x-ray Classification – High Level Process Map





Data Preparation

Step1: Chest x-ray image of COVID-19,Pneumonia and Normal.

Step2: Normalization the data (image).

Step3: Data Augmentain

Step4: Resize of image.

Step5: Accuracy check after classifying the image

Production

Step1: Collect Chest X-Ray image from patient.

Step2: Normalize the data (image) Step3:

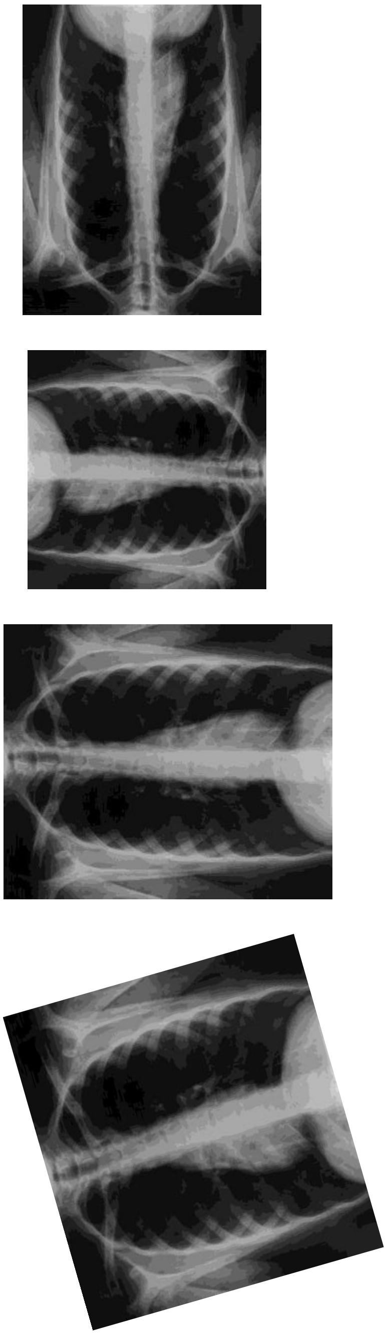
Resizing of images.

Step4: Classifying the images.

Exploratory Data Analysis

Chest x-ray

Augmented Chest x-ray Image

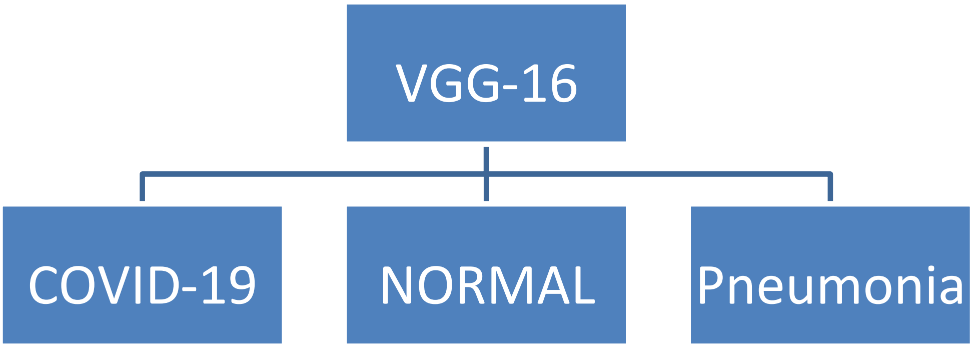


- Image rotations rotation_image
- Image Shifts using width_shift_range
- Height Shift Range
- Image Flips using horizontal_flip.

Architecture – Transfer Learning using the pre- trained models:

VGG 16 and Inception V3

VGG-16 :-



VGG-16

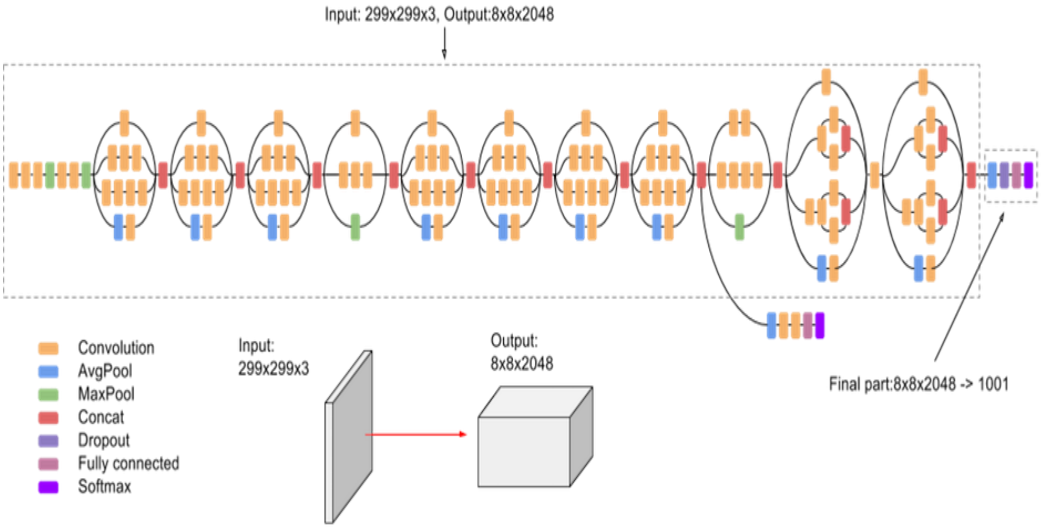
COVID-19

NORMAL

Pneumonia

- VGG-16 is a convolution network architecture.
- It consists of 13 convolution layers , but only 16 weight layers.
- Pre-processed our data using image Data Generator from keras preprocessing library.
- Image size a VGG-16 network.

Inception V3 :-



Inception-v3 ,a 42 layers deep convolution neural net on the same data set.

VGG 16 VS Inception V3

VGG-16

Confusion Matrix

	Confusion Matrix	Actual		
		Covid-19	Normal	Pneumonia
Predicted	COVID -19	21	1	2
	Normal	1	853	47
	Pneumonia	0	31	545

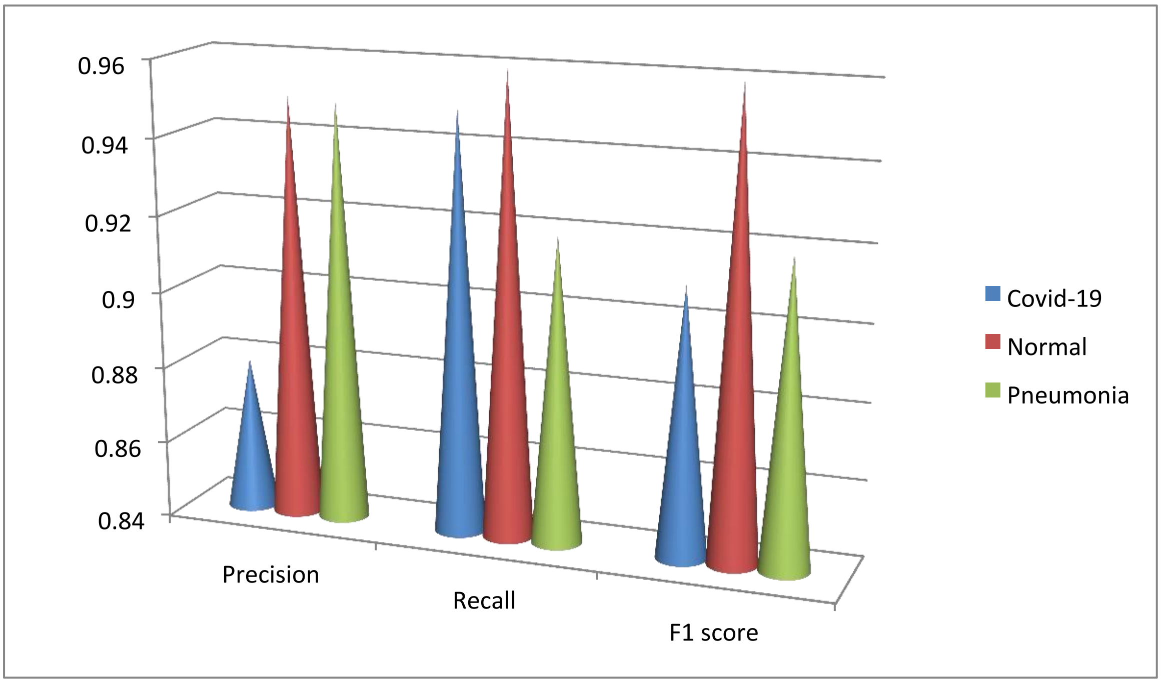
Inception V3

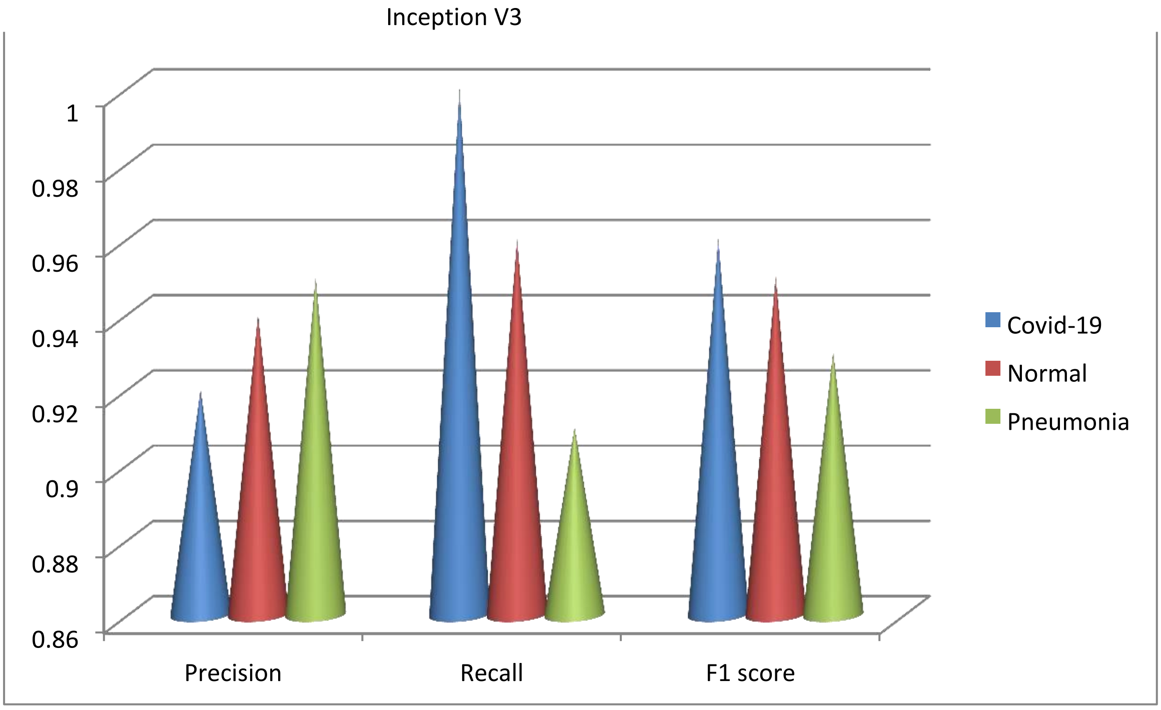
Confusion Matrix

	Confusion Matrix	Actual		
		COVID-19	Normal	Pneumonia
Predicated	COVID-19	22	2	0
	Normal	0	853	54
	Pneumonia	0	30	540

Classification Report

VGG 16





Cost Benefit Analysis - Business Perspective

PROJECT CREATION

For this analysis we are using google Collaboratory.
Importing libraries and downloading datasets.

```
import pandas as pd
import numpy as np
import os, random
import pickle
from imutils import paths

import keras
from keras.models import *
from keras.layers import *
from keras.preprocessing import image
from keras.callbacks import ModelCheckpoint

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score


import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
tf.test.gpu_device_name()
```

Downloading dataset

https://github.com/vz-risk/veris

Importing dataset

I imported the data by cloning the entire VERIS repo on GitHub to my local environment. Because each incident is stored in an individual JSON object, I used the helpful verispy package to extract the thousands of JSON objects into a single pandas data frame:

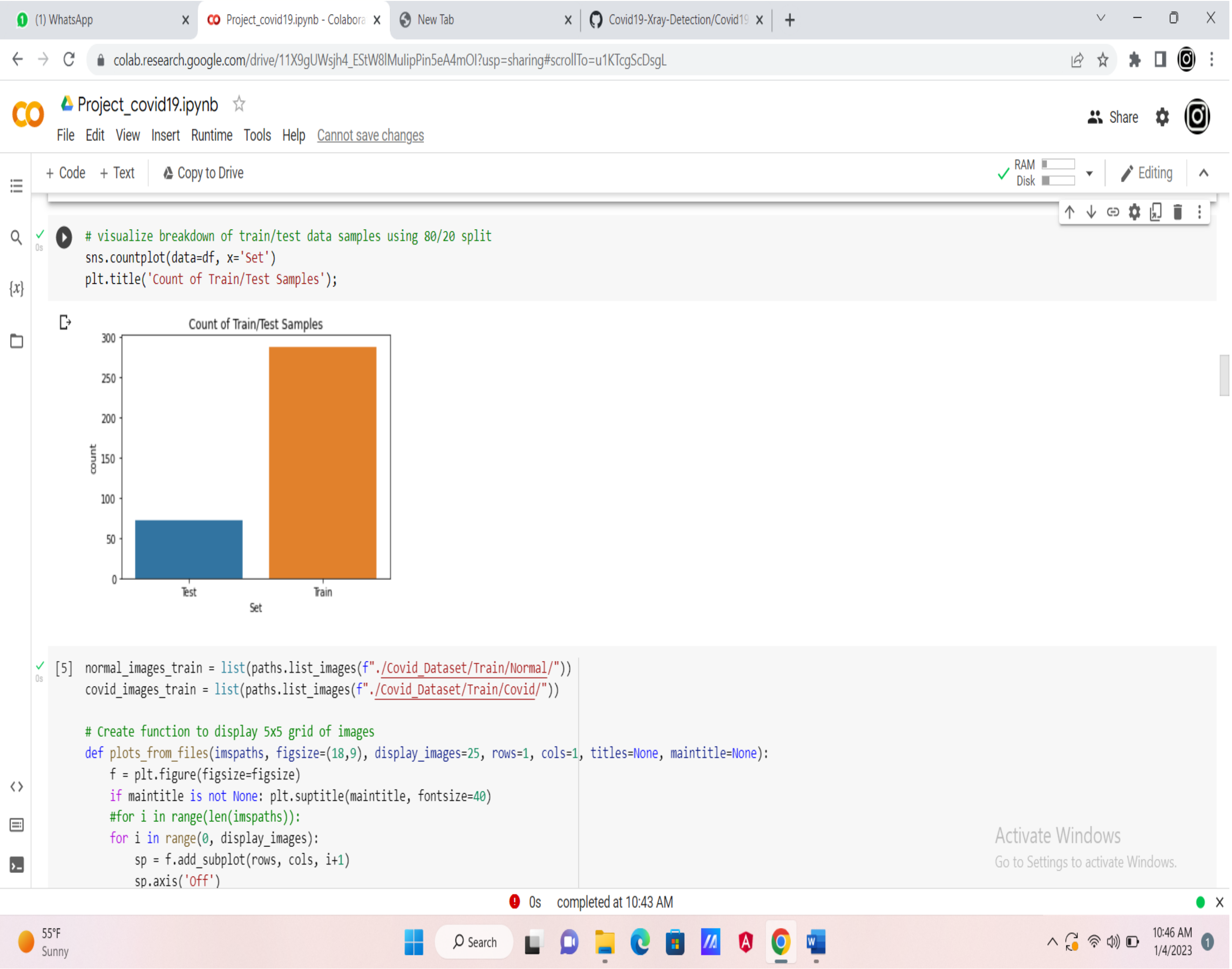
Inspecting the dataframe reveals the extent of the VERIS Community Dataset (as of Jan 2022):

```
!wget https://www.dropbox.com/s/oo9i9ttn3di0182/Covid_Dataset_2020_07_31.zip?dl=0
!unzip Covid_Dataset_2020_07_31.zip\?dl\=0
print("Data set successfully imported.")
```

Using the directory filepath to classify the images

```
path = "./Covid_Dataset/"
filepath = []

for root, directories, files in os.walk(path, topdown=False):
  for name in files:
    filepath.append(os.path.join(root, name))
    #print(os.path.join(root, name))
  #for name in directories:
    #print(os.path.join(root, name))

set = []
diagnosis = []
imgname = []

for i in filepath:
  imgname.append(i.split("/")[4])
  set.append(i.split("/")[2])
  diagnosis.append(i.split("/")[3])

df = pd.DataFrame(list(zip(imgname, set, diagnosis)),
                  columns = ['Filename', 'Set', 'Diagnosis'])

# metadata exported to a csv
df.to_csv('covid_metadata.csv', index=False)

df.sample(5)
```

```
# visualize breakdown of train/test data samples using 80/20 split
sns.countplot(data=df, x='Set')
plt.title('Count of Train/Test Samples');
```



(1) WhatsApp
Project_covid19.ipynb - Colabora
New Tab
Covid19-Xray-Detection/Covid19
colab.research.google.com/drive/11X9gUWsjh4_EStW8lMulipPin5eA4mOl?usp=sharing#scrollTo=u1KTcgScDsgL
Project_covid19.ipynb
Share
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
RAM
Disk
Editing

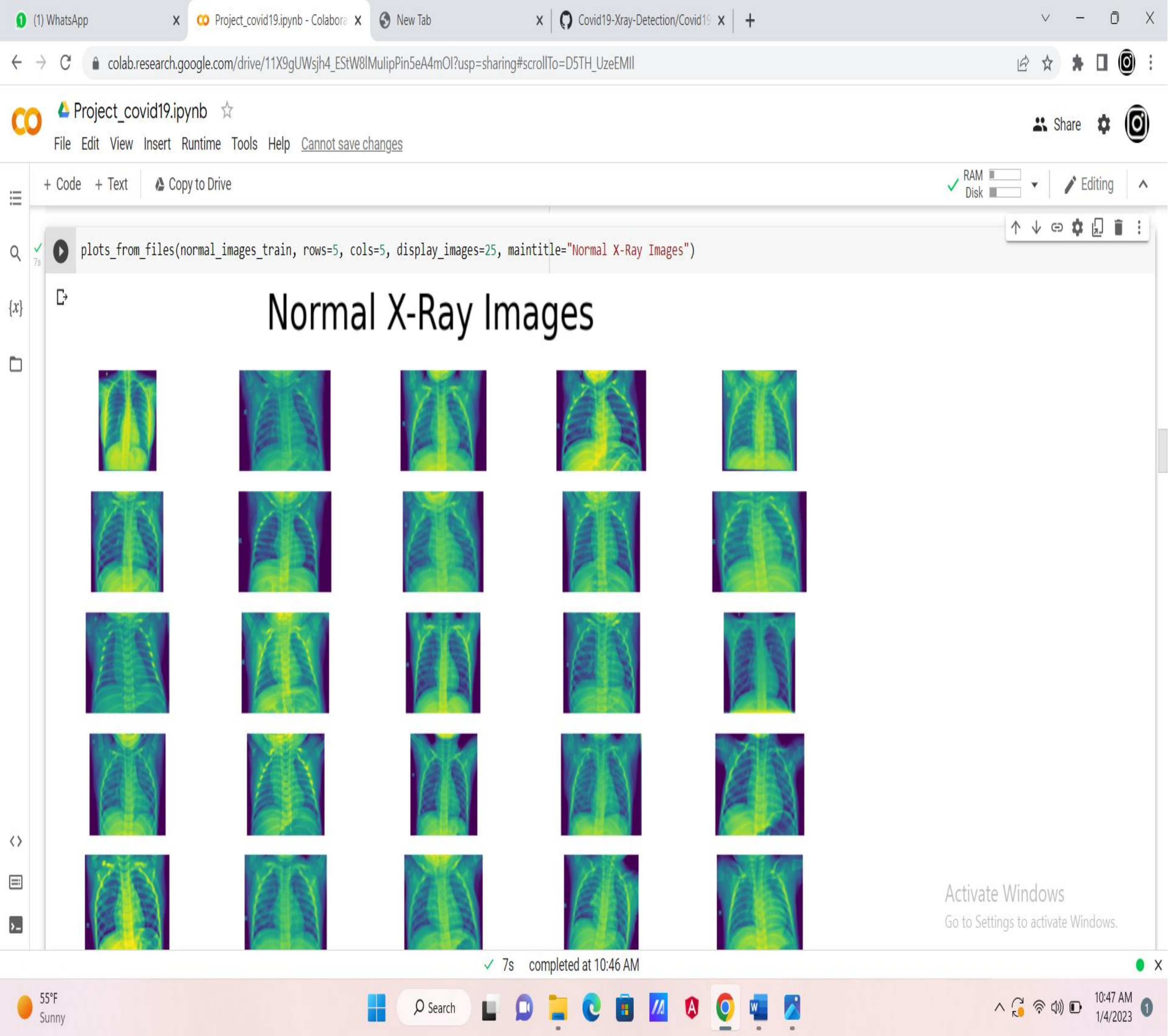
```
# visualize breakdown of train/test data samples using 80/20 split
sns.countplot(data=df, x='Set')
plt.title('Count of Train/Test Samples');
```

Count of Train/Test Samples
300
250
200
150
100
50
0
count
Test
Train
Set

```
[5] normal_images_train = list(paths.list_images(f"./Covid_Dataset/Train/Normal/"))
    covid_images_train = list(paths.list_images(f"./Covid_Dataset/Train/Covid/"))

    # Create function to display 5x5 grid of images
    def plots_from_files(imspaths, figsize=(18,9), display_images=25, rows=1, cols=1, titles=None, maintitle=None):
        f = plt.figure(figsize=figsize)
        if maintitle is not None: plt.suptitle(maintitle, fontsize=40)
        #for i in range(len(imspaths)):
        for i in range(0, display_images):
            sp = f.add_subplot(rows, cols, i+1)
            sp.axis('Off')
```

```
normal_images_train = list(paths.list_images(f"./Covid_Dataset/Train/Normal/"))
covid_images_train = list(paths.list_images(f"./Covid_Dataset/Train/Covid/"))

# Create function to display 5x5 grid of images
def plots_from_files(imspaths, figsize=(18,9), display_images=25, rows=1, cols=1, t
itles=None, maintitle=None):
    f = plt.figure(figsize=figsize)
    if maintitle is not None: plt.suptitle(maintitle, fontsize=40)
    #for i in range(len(imspaths)):
    for i in range(0, display_images):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None: sp.set_title(titles[i], fontsize=16)
        img = plt.imread(imspaths[i])
        plt.imshow(img)
```

```
plots_from_files(normal_images_train, rows=5, cols=5, display_images=25, maintitle=
"Normal X-Ray Images")
```



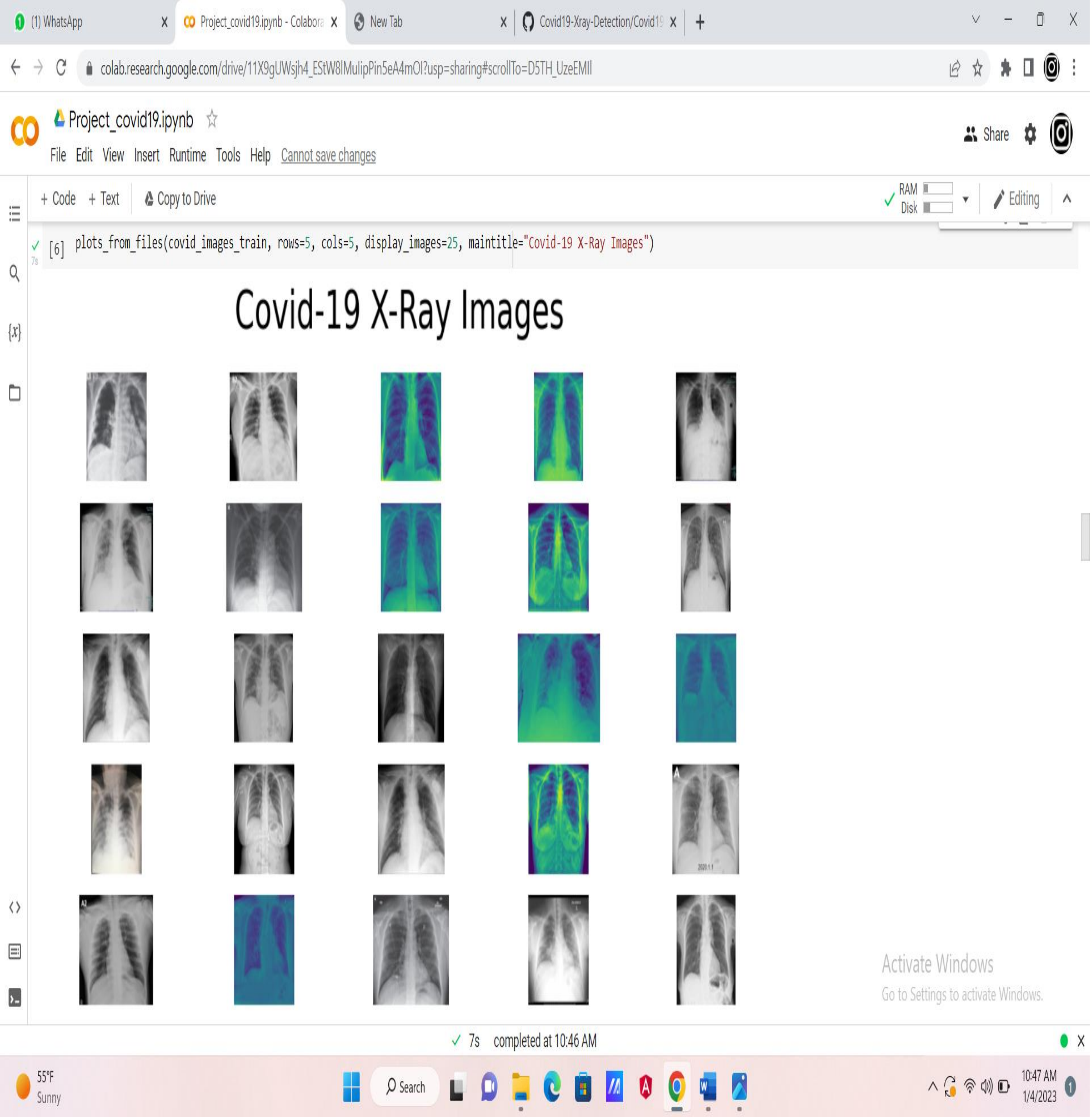
(1) WhatsApp
Project_covid19.ipynb - Colabora
New Tab
Covid19-Xray-Detection/Covid19
colab.research.google.com/drive/11X9gUWsjh4_EStW8lMulipPin5eA4mOl?usp=sharing#scrollTo=D5TH_UzeEMll
Project_covid19.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
RAM
Disk
Editing
Activate Windows
Go to Settings to activate Windows.
55°F
Sunny
Search
10:47 AM
1/4/2023

```
plots_from_files(normal_images_train, rows=5, cols=5, display_images=25, maintitle="Normal X-Ray Images")
```

Normal X-Ray Images

✓ 7s completed at 10:46 AM

```
plots_from_files(covid_images_train, rows=5, cols=5, display_images=25, maintitle="
Covid-19 X-Ray Images")
```



(1) WhatsApp
Project_covid19.ipynb - Colabora
New Tab
Covid19-Xray-Detection/Covid19
colab.research.google.com/drive/11X9gUWsjh4_EStW8lMulipPin5eA4mOl?usp=sharing#scrollTo=D5TH_UzeEMlI
Project_covid19.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
RAM
Disk
Editing
Covid-19 X-Ray Images
Activate Windows
Go to Settings to activate Windows.
✓ 7s completed at 10:46 AM
55°F
Sunny
Search
10:47 AM
1/4/2023

```
[6] plots_from_files(covid_images_train, rows=5, cols=5, display_images=25, maintitle="Covid-19 X-Ray Images")
```

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(224,224,3))
)
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss=keras.losses.binary_crossentropy, optimizer='adam', metrics=['ac
curacy'])
```

SUMMARY OF MODEL

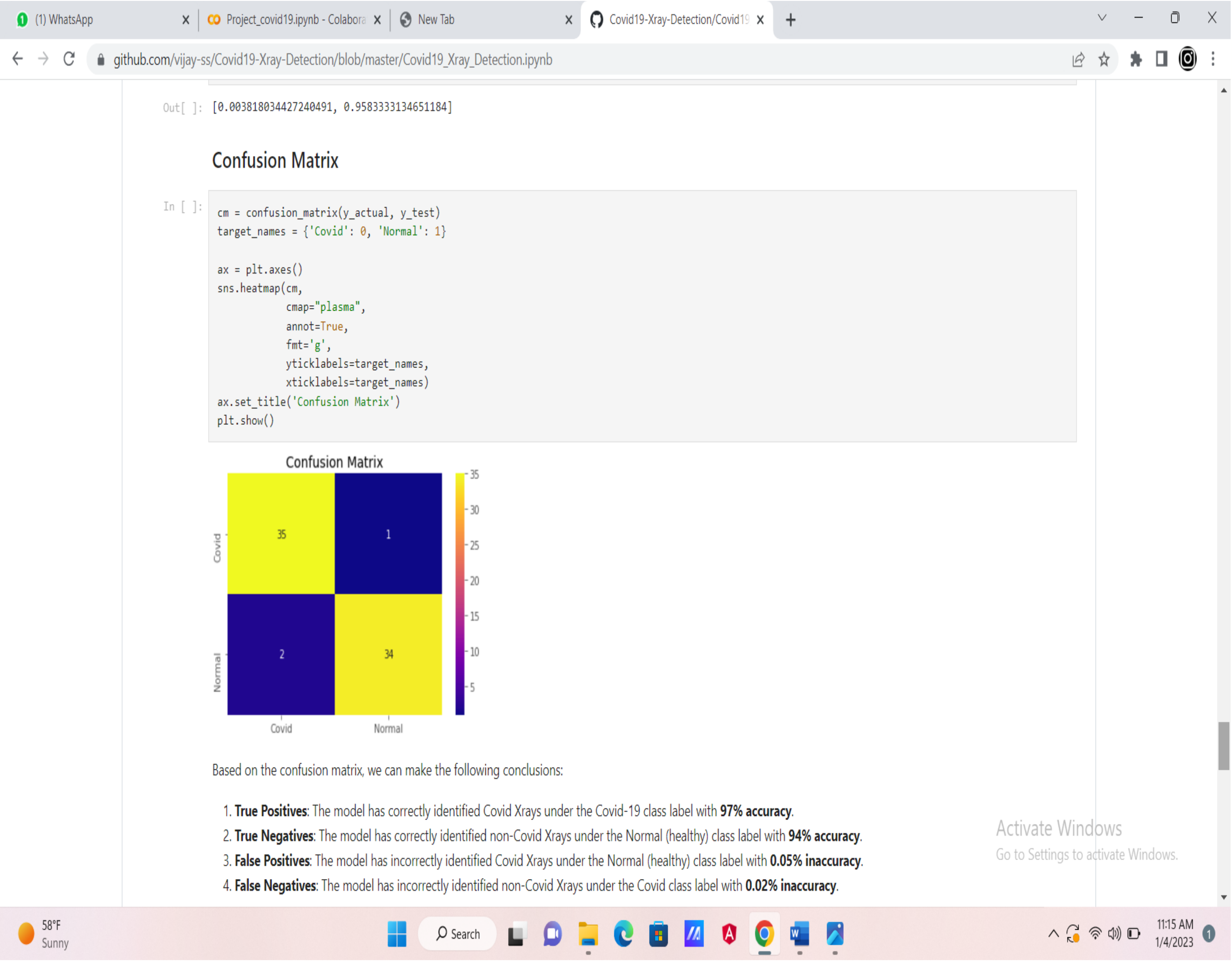
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 conv2d_1 (Conv2D)           (None, 220, 220, 64)      18496

 max_pooling2d (MaxPooling2D  (None, 110, 110, 64)     0
 )

 dropout (Dropout)           (None, 110, 110, 64)      0

 conv2d_2 (Conv2D)           (None, 108, 108, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 54, 54, 64)        0

 conv2d_3 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 26, 26, 128)       0

 flatten (Flatten)           (None, 86528)             0

 dense (Dense)               (None, 64)                5537856

 dropout_3 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 5,668,097
Trainable params: 5,668,097
Non-trainable params: 0
```

```
# intialize number of epochs and batch size to train for
EPOCHS = 5
BS = 32
```

```
# prepping & augmenting the images for loading into the model as inputs for training
train_data = image.ImageDataGenerator(
    rescale = 1./255, #normalize images
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True
)

test_data = image.ImageDataGenerator(rescale = 1./255)

# converting image size, set the batch size (standard 224 by 224)
# feed the augmented images from previous step into the generator
train_generator = train_data.flow_from_directory(
    'Covid_Dataset/Train',
    target_size = (224,224),
    batch_size = BS,
    class_mode = 'binary'
)

# print label for each class
train_generator.class_indices

test_generator = test_data.flow_from_directory(
    'Covid_Dataset/Test',
    target_size = (224,224),
    batch_size = BS,
    class_mode = 'binary'
)



%%time

hist = model.fit_generator(
    train_generator,
    #steps_per_epoch = len(next(os.walk("./Covid_Dataset/Train/Covid/"))[2]) // BS,
    steps_per_epoch = 8,
    epochs = EPOCHS,
    validation_data = test_generator,
    validation_steps = 2
    #validation_steps = len(next(os.walk("./Covid_Dataset/Test/Covid/"))[2]) // BS
)

# checkpoint to save highest accuracy
"""
filepath="weights.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
"""

# save the model to file
```

```
if os.path.exists("covid19_model_adv.h5"):
   os.remove("covid19_model_adv.h5")
   print("Previous model deleted.")
else :
   print("The file does not exist.")

model.save("covid19_model_adv.h5")
print("Most recent model saved as: covid19_model_adv.h5")


# save model weights
model.save_weights("covid19_model_weights.h5")
print("Model weights saved.")


# save to JSON format
model_json = model.to_json()
with open('covid19_model_adv.json', 'w') as json_file:
  json_file.write(model_json)
print('Model saved to JSON format.')


model = load_model("covid19_model_adv.h5")
print('Model loaded from file.')


train_generator.class_indices


y_actual = []
y_test = []


y_actual = np.array(y_actual)
y_test = np.array(y_test)

model.evaluate_generator(train_generator)

model.evaluate_generator(test_generator)



cm = confusion_matrix(y_actual, y_test)
target_names = {'Covid': 0, 'Normal': 1}

ax = plt.axes()

ax.set_title('Confusion Matrix')
plt.show()
```


(1) WhatsApp
Project_covid19.ipynb - Colabora
New Tab
Covid19-Xray-Detection/Covid19
github.com/vijay-ss/Covid19-Xray-Detection/blob/master/Covid19_Xray_Detection.ipynb
Confusion Matrix
35
1
2
34
Covid
Normal
Covid
Normal
35
30
25
20
15
10
5
Activate Windows
Go to Settings to activate Windows.
58°F
Sunny
Search
11:15 AM
1/4/2023

```
Out[ ]: [0.003818034427240491, 0.9583333134651184]
```

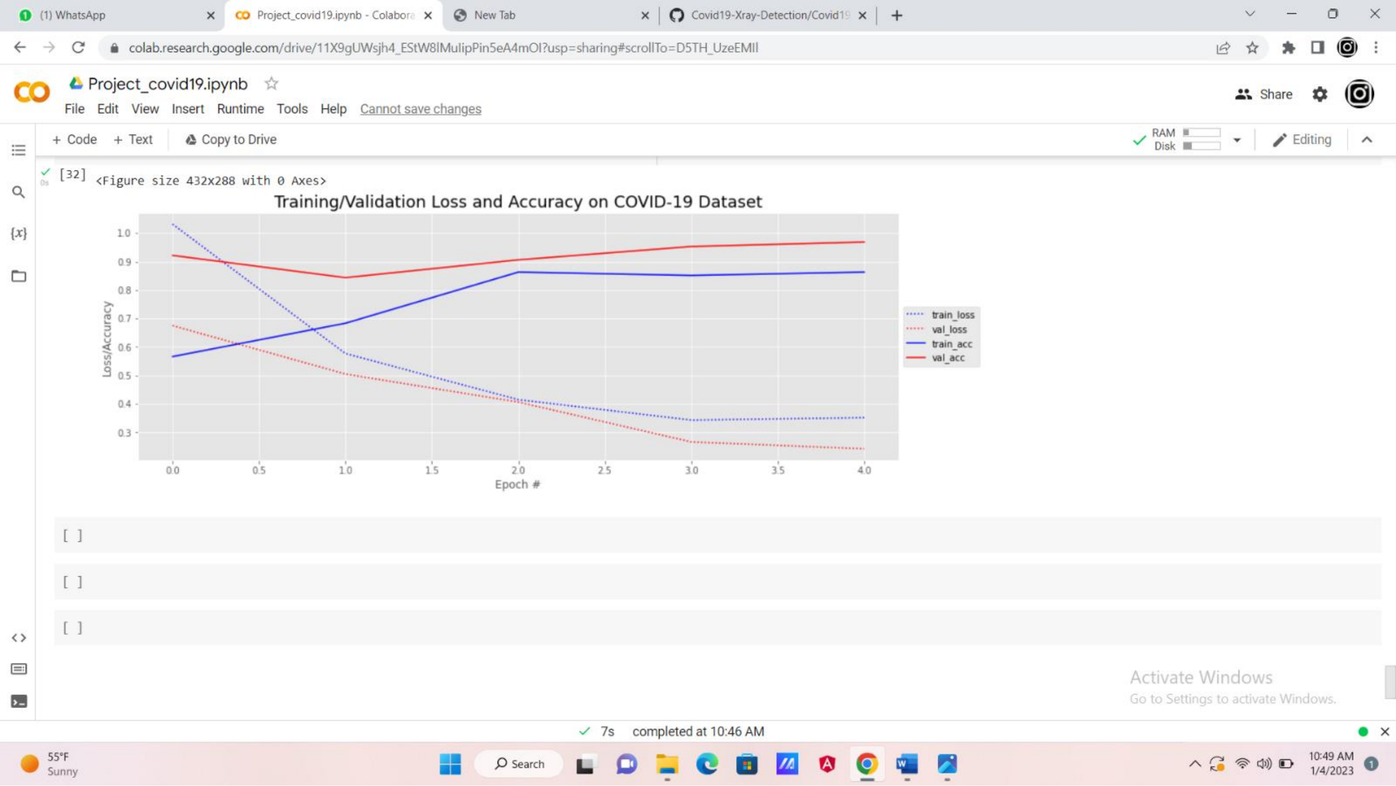
Confusion Matrix

```
In [ ]: cm = confusion_matrix(y_actual, y_test)
        target_names = {'Covid': 0, 'Normal': 1}

        ax = plt.axes()
        sns.heatmap(cm,
                    cmap="plasma",
                    annot=True,
                    fmt='g',
                    yticklabels=target_names,
                    xticklabels=target_names)
        ax.set_title('Confusion Matrix')
        plt.show()
```

Based on the confusion matrix, we can make the following conclusions:

1. **True Positives**: The model has correctly identified Covid Xrays under the Covid-19 class label with **97% accuracy**.
2. **True Negatives**: The model has correctly identified non-Covid Xrays under the Normal (healthy) class label with **94% accuracy**.
3. **False Positives**: The model has incorrectly identified Covid Xrays under the Normal (healthy) class label with **0.05% inaccuracy**.
4. **False Negatives**: The model has incorrectly identified non-Covid Xrays under the Covid class label with **0.02% inaccuracy**.

```
print("Precision score: {:.4f}".format(precision_score(y_actual, y_test)))
print("Recall score: {:.4f}".format(recall_score(y_actual, y_test)))
print("F1 Score: {:.4f}".format(f1_score(y_actual, y_test)))


N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.figure(figsize=(15,5))
plt.plot(np.arange(0, N), hist.history["loss"], label="train_loss", color='blue', linestyle=':')
plt.plot(np.arange(0, N), hist.history["val_loss"], label="val_loss", color='red', linestyle=':')
plt.plot(np.arange(0, N), hist.history["accuracy"], label="train_acc", color='blue')
plt.plot(np.arange(0, N), hist.history["val_accuracy"], label="val_acc", color='red')
plt.title("Training/Validation Loss and Accuracy on COVID-19 Dataset", fontsize=18)
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5),prop={'size':10})
plt.show()
```


(1) WhatsApp
Project_covid19.ipynb - Colabora
New Tab
Covid19-Xray-Detection/Covid19
colab.research.google.com/drive/11X9gUWsjh4_EStW8lMulipPin5eA4mOI?usp=sharing#scrollTo=D5TH_UzeEMIl
Project_covid19.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text
Copy to Drive
RAM
Disk
Editing
[32]
Training/Validation Loss and Accuracy on COVID-19 Dataset
Loss/Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch #
train_loss
val_loss
train_acc
val_acc
[ ]
[ ]
[ ]
Activate Windows
Go to Settings to activate Windows.
7s completed at 10:46 AM
55°F
Sunny
Search
10:49 AM
1/4/2023

```
<Figure size 432x288 with 0 Axes>
```

CONCLUSION

- An effective approach that can assist healthcare staff in the diagnosis of chest x-rays.
- For an example , an expert radiologist will analyse 1000 chest x-rays in the day (100 x-rays per hour , analysing 1000 x-rays would incur an annual cost of $36,500($10*10hrs*365days)
- A well trained machine learning algorithm using convolution Neural Networks with image classification techniques take minutes to classify the same volume of chest x-rays.
- Our COVIDx-19 NET AI product will help to ramp up the testing by 4 to 5 times thus saving the hung time on manual review with a very minimum involvement of the radiologist .

Conclusion :

Model	**Training set**	**No. of parameters**	**Train Accuracy**	**Test Accuracy**	**Comment**
VGG16	3 classes with13512 images	138 M	92.00	94.00	Given good accuracy .But extremely slow in training and testing.
Inception v3	3 classes with13,512 Images.	6.5M	93.23	97.09	Given good accuracy . Reasonable fast in training and testing.

Accuracy vs Computing Time :

Accuracy may take precedence over computation time in this use case because it is tied to the healthcare industry and directly affects people's lives, however this is disputed in consultation with health care specialists.

- Using traditional CNN models provided us with extremely high accuracy.
- VGG16, a pre-trained model from Keras on these photos, performed somewhat better for image classification than Inception V3 (94% accuracy vs.

95% accuracy).

BIBLIOGRAPHY

www.Wikipedia.com
www.youtube.com
www.codewithstein.com
www.itsourcecode.com
www.geeksforgeeks.com
www.javapoint.com
www.tutorialspoint.com