A CS737 Lab Report on

"Covid-19 Recognition Using CNN"

Submitted in partial fulfilment of the requirements for the degree of

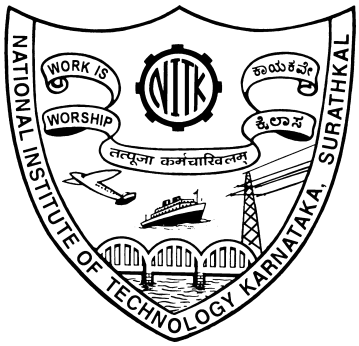
MASTER OF TECHNOLOGY

in

INFORMATION SECURITY

by

Mr. SHIVAM PANDEY
(202IS024)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE -575025

May, 2021

Abstract

The outbreak of Coronavirus disease 2019 (COVID-19), caused by severe acute respiratory syndrome (SARS) coronavirus 2 (SARS-CoV-2), has so far killed many people across the globe, resulting in catastrophe for humans.

So, we are trying to develop a deep learning model to detect the disease from CT Scan.In this deep learning model we are using VGG-16 architecture. The dataset contains both training and testing data. We are training our model on the train data and then it is tested on the test data on basis of different evaluation measures. We tried our best to develop the model in such a way that it can give good results in real time scenarios. We achieved an accuracy of 95-98% during training phase while during testing we got the accuracy of 63%.

Contents

List of Figures

Chapter 1

The Dataset

The dataset which we are using for this model is divided into parts i.e. train data and test data. These datasets are further divided into two parts i.e. COVID(samples of people infected of disease) and non-COVID(samples of people who are not infected of disease). Each individual sample is an image of 100 X 100 pixels.

We have performed ***normalisation*** and ***shuffling*** before using the dataset to train our model.

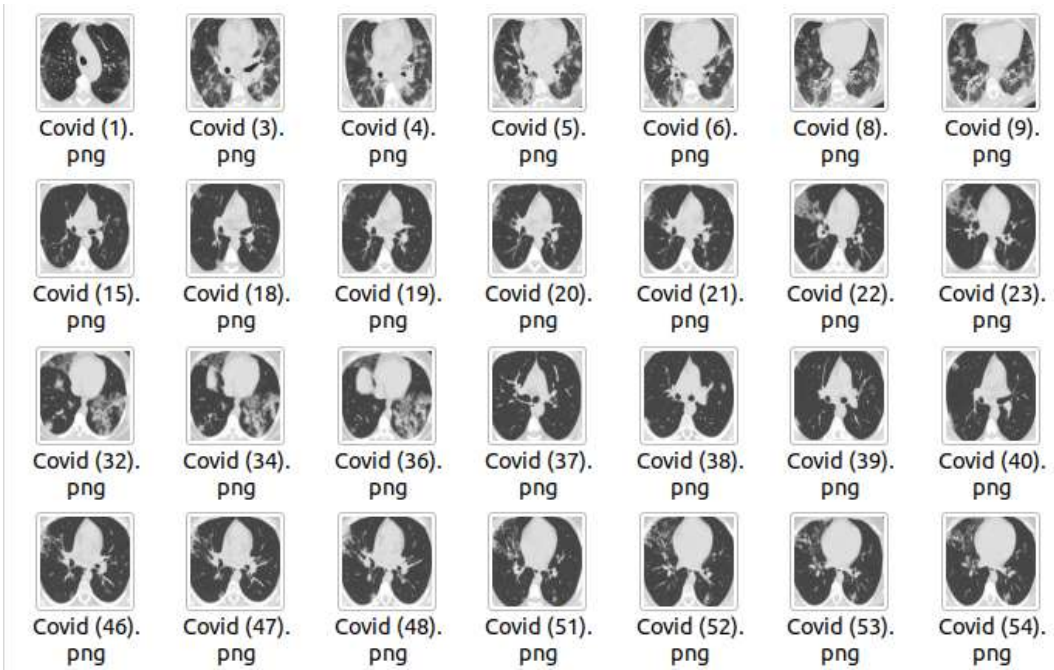


Figure 1.1: Snapshot of the Dataset

Chapter 2

The Architecture

To develop the model we are using ***VGG-16*** architecture along with ***Adamax*** optimizer.

2.1 Flow of the Model

- Mounting the drive and fetching the dataset
- Normalisation, Shuffling and Encoding of the dataset
- Implementing the CNN Model
- Calling the Optimizer
- Training & Testing the Model
- Evaluation of the Model

2.2 Architecture : VGG-16

We are using VGG-16 architecture to develop our model. Along with this we are using keras and numpy libraries. We are using **ReLU** activation function in the hidden layers to increase the convergence rate of the model. At the last layer we are using **Sigmoid** as it is bounded activation function. The last three layers in the model are fully connected. The filters are varying from 64 to 512 in different layers. Our model is trained for 15 epochs.

2.3 Optimizer : Adamax

We are using Adamax optimizer to optimize this model. It increases the convergence rate of our model. VGG-16 along with adamax is giving good results.

```
model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=1, activation="sigmoid"))

model.summary()
```

Figure 2.1: The Architecture : VGG-16

Chapter 3

Results

The model gave an accuracy of 95-98% during training phase while during the testing it gave the accuracy of 63%. We evaluated our model based on following measures-

- Accuracy : 63%
- F1 Score : 60%
- Recall : 63%
- Precision : 68%

We also plotted some graphs and created confusion matrix for better understanding of results.

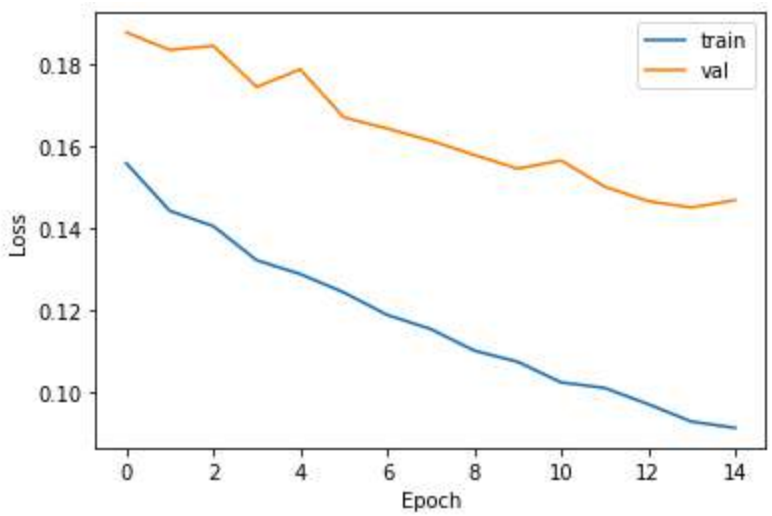


Figure 3.1: Loss Vs Epoch

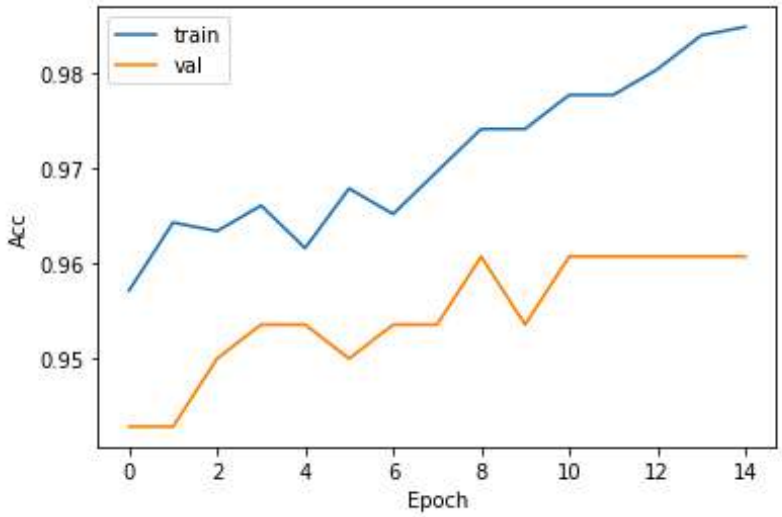


Figure 3.2: Accuracy Vs Epoch

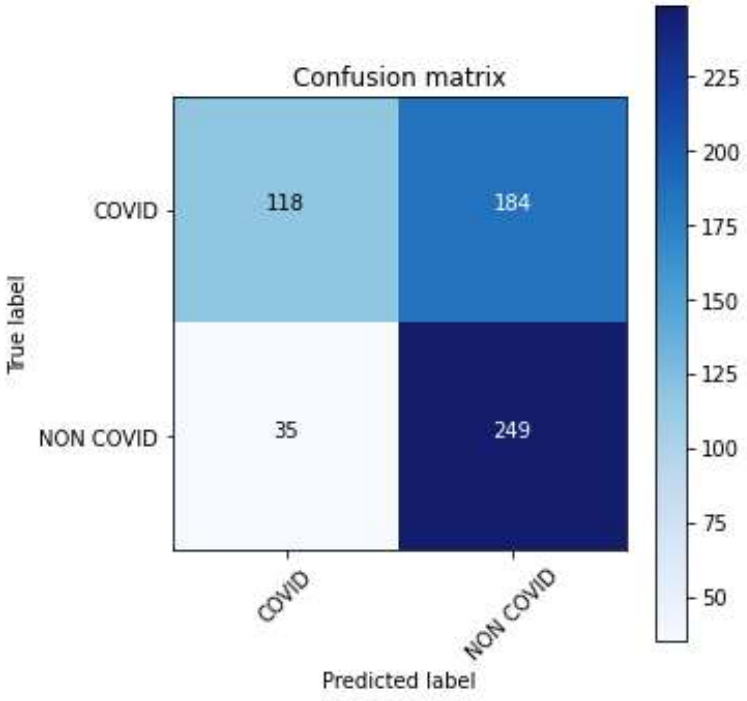


Figure 3.3: Confusion Matrix

Chapter 4

Conclusion

We successfully developed a deep learning model using ***VGG-16*** architecture and ***Adamax*** optimizer over the Covid-19 dataset. In the end, we achieved 98% accuracy in the training phase and 63% accuracy in the testing phase.