Covid 19 Detection using X-Ray Images

PRML Major Project

Ved Brahmbhatt (B21EE075)
Samarth Patel (B21CS057)
Sai Kiran Reddy (B21CS056)

1. Introduction

The COVID-19 pandemic has become a major global health crisis, with millions of cases and fatalities reported worldwide. Early detection of COVID-19 is critical to prevent further spread of the disease and ensure timely treatment for infected individuals. In this regard, medical imaging techniques such as X-rays have been widely used for the diagnosis of COVID-19.

X-rays can provide valuable information about the lungs, which are often affected in COVID-19 cases. By analyzing X-ray images, doctors can detect abnormalities in the lungs such as infiltrates, consolidation, and ground-glass opacities, which are common in COVID-19 patients.

Therefore, the use of X-ray images for COVID-19 detection has gained significant attention in recent times. Researchers and healthcare professionals are exploring the potential of deep learning algorithms and other advanced techniques to accurately diagnose COVID-19 using X-ray images.

This project aims to explore the feasibility of using X-ray images for the detection of COVID-19 and develop a deep learning model that can accurately identify COVID-19 cases from X-ray images. The successful implementation of such a model can have a significant impact on early detection and effective management of the COVID-19 pandemic.

2.Dataset Description

The dataset contains images with covid positive X Rays, normal X Rays (Non Covid), Pneumonia affected X Rays and lung infection X Rays.

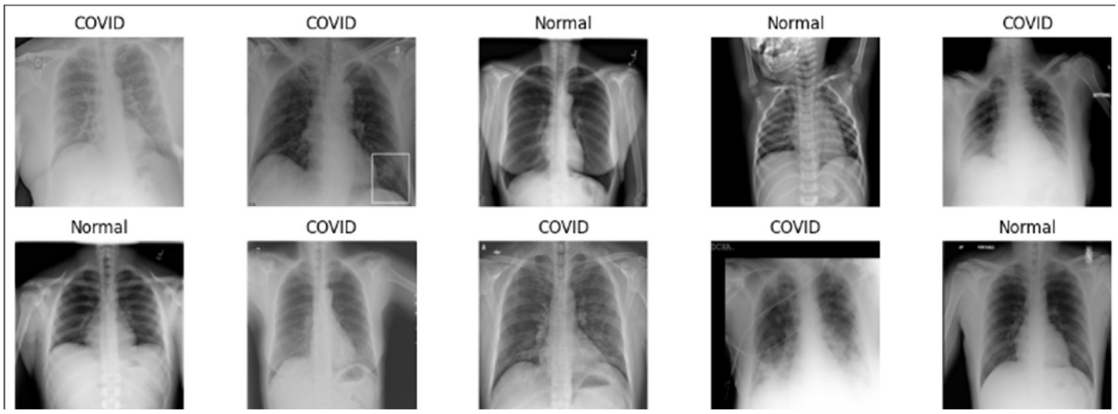
The number of images in each of the following cases are 3616, 10192, 1345 and 6012 respectively.

All the models built by us are trained on classifying between Covid Positive and Normal X Rays. Hence the total number of images in our dataset will be 10192 + 3616 = 13808.

3.Importing the Dataset and Exploratory Data Analysis

The Dataset was downloaded from Kaggle and defined the path for both *Covid* and *Non-Covid* Images. Then we import the dataset using class *CustomDataset(torch.utils.data.Dataset)* and apply the transformation of the Dataset. Then using *train_test_split()* function of Sklearn library we split the dataset into training and testing dataset in 80:20. We take the batch_size=32 and image width and height =224.

Random images from the dataset are shown below

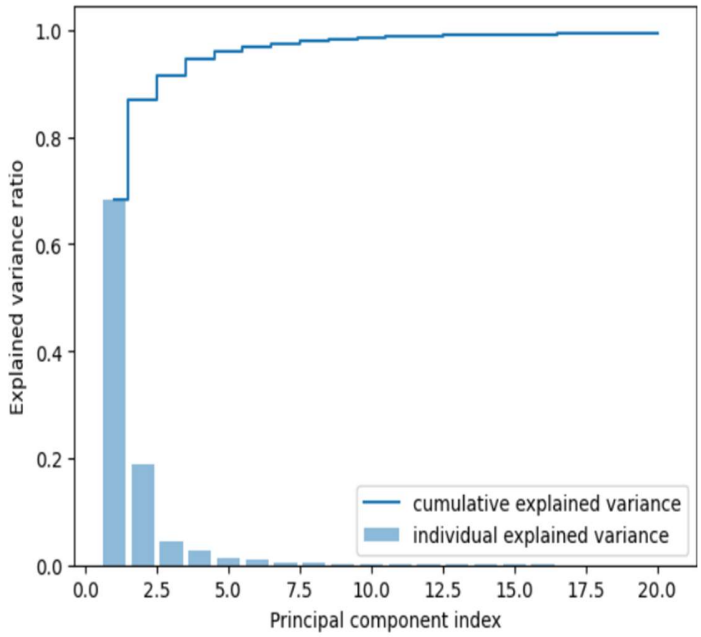


4.Dimensionality Reduction using PCA

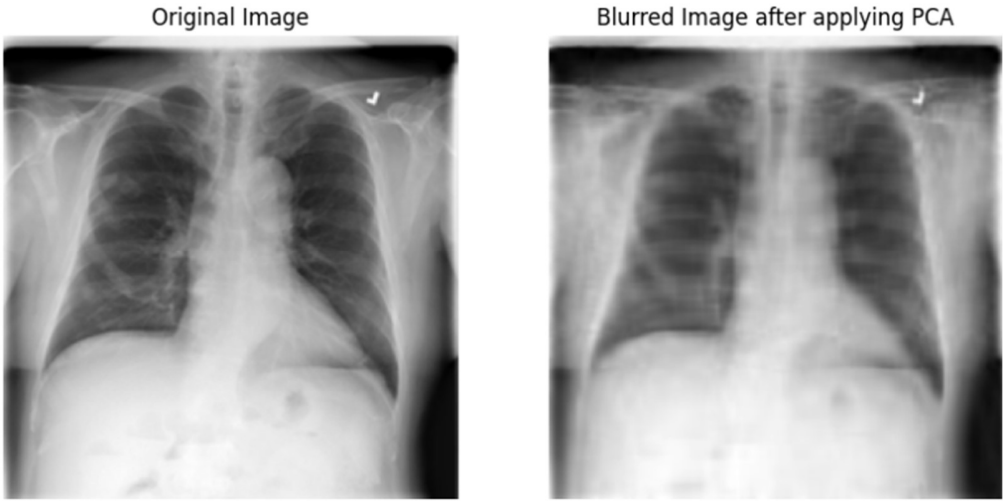
PCA was implemented to :

- Reduce the time and storage required.
- Remove the multicollinearity which improves the interpretation of the parameters of the machine learning model.

For deciding the number of principal components we plot a Scree Plot and threshold variance of 99% was set and image's Direction were reduced as shown.



n_components = 20 was used as they covered 99% of the cumulative variance.



5.Application of Machine Learning Models

5.1 Creating Dataset:

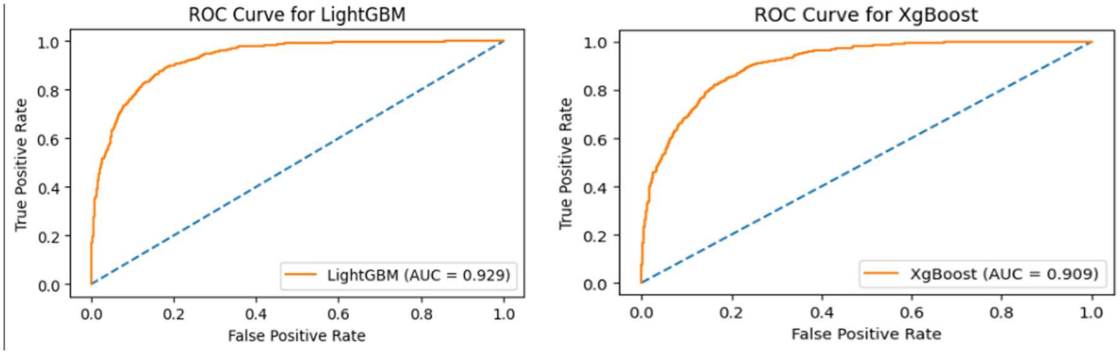
Dimensions of images are reduced to 224x20 after PCA, which is then flattened and stored in a numpy array.

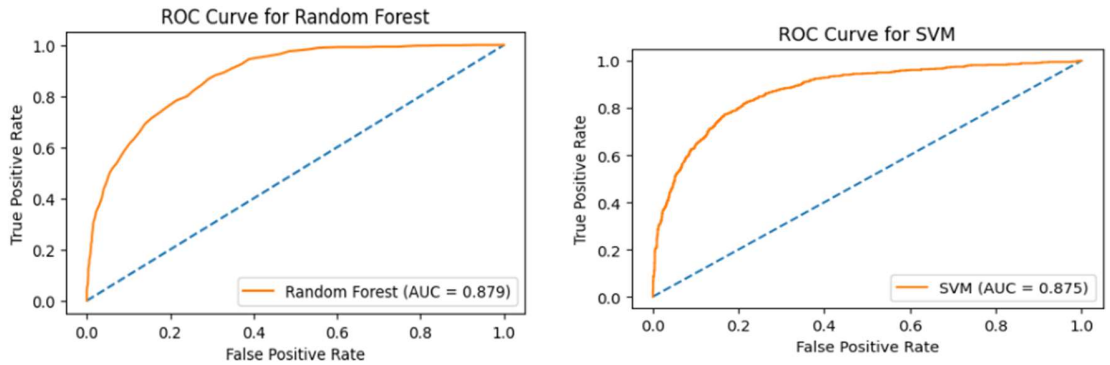
5.2 Train and Test Split:

Using train_test_split() of the Sklearn library the dataset is splitted into 80:20 ratio.

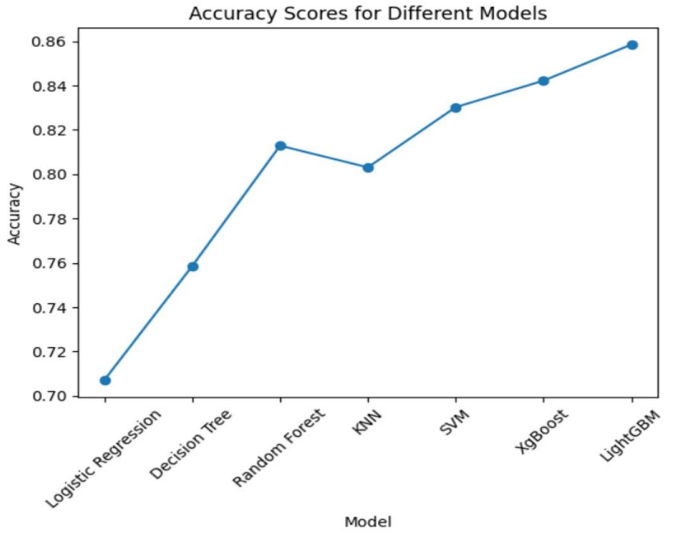
Index	Model	Accuracy	F1 Score	Precision	Recall	ROC AUC
0	Logistic Regression	0.707096	0.381025	0.436842	0.337856	0.714922
1	Decision Tree	0.758508	0.547797	0.547425	0.548168	0.691615
2	Random Forest	0.812817	0.526990	0.808989	0.390773	0.879427
3	KNN	0.803041	0.609195	0.647328	0.575305	0.846046
4	SVM	0.830196	0.615889	0.776860	0.510176	0.875286
5	XgBoost	0.842143	0.657771	0.780261	0.568521	0.909471
6	LightGBM	0.858436	0.694292	0.819188	0.602442	0.929397

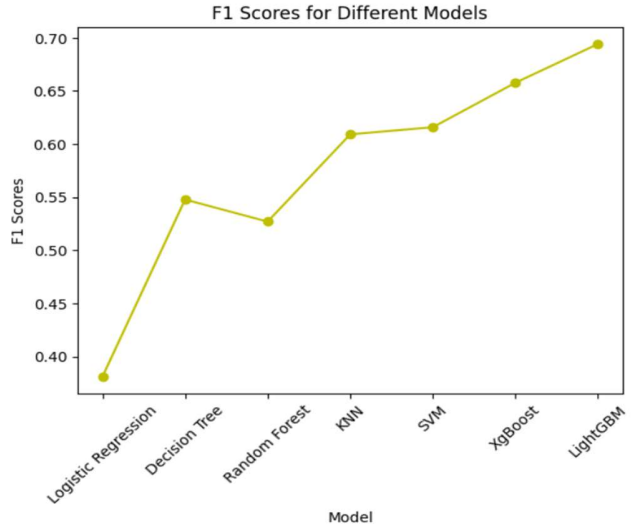
ROC AUC curves for top four performing models are shown below



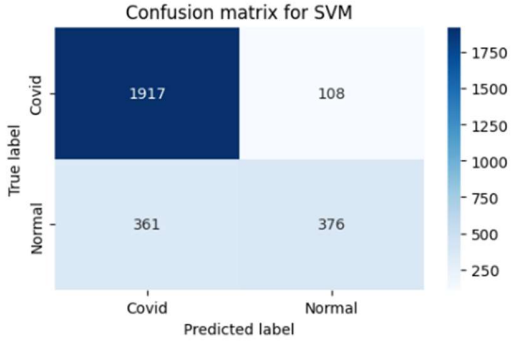


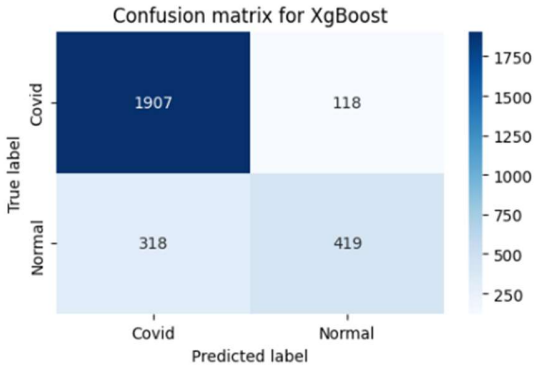
Comparision of results for various models is shown below

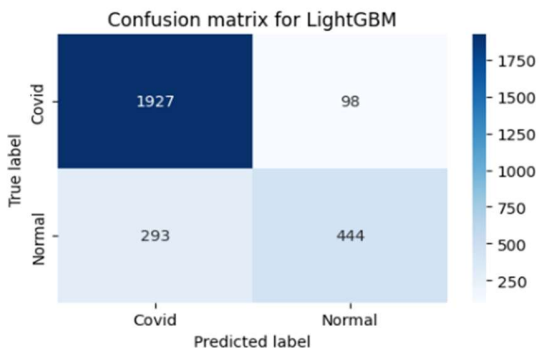


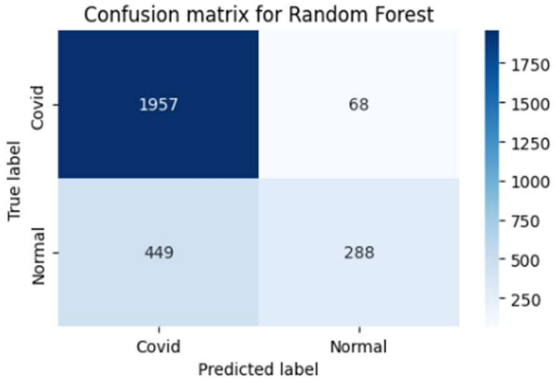


Confusion Matrix for various models are shown below









6.Deep Learning Models:

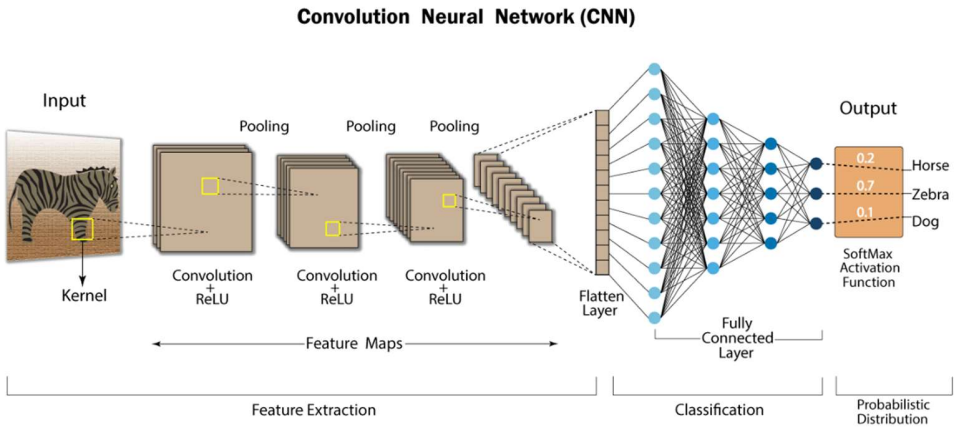
6.1 Creation of Dataset:

The same dataset which has been imported using the CustomDataset() is taken and used for training and validating different deep learning models.

6.2 CNN:

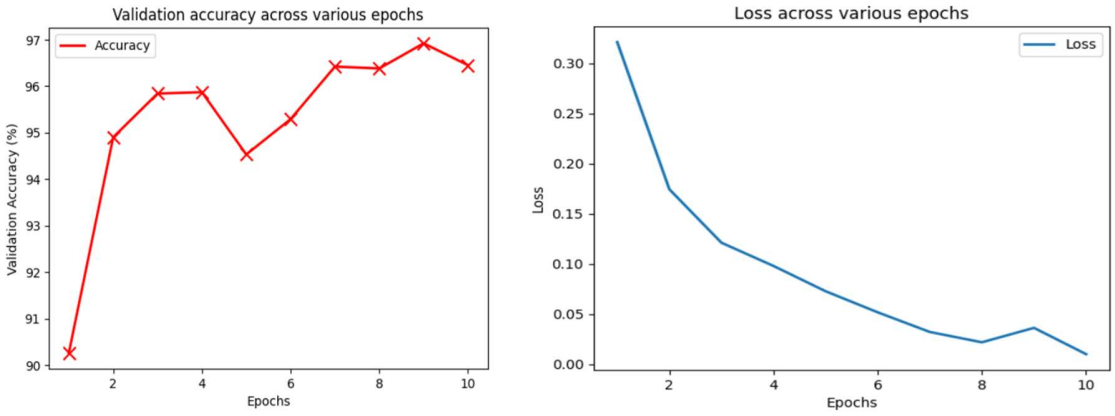
Convolutional Neural Network (CNN) is a deep learning algorithm that is widely used for image and video recognition tasks. CNNs use a series of convolutional layers to extract features from input images and then use pooling layers to reduce the dimensionality of the extracted features. This enables the network to learn spatial hierarchies of features, which are useful for recognizing patterns in images.

We have implemented a Convolutional Neural Network (CNN) model for image classification using PyTorch. The model architecture consists of two convolutional layers, each followed by a max pooling layer, and two fully connected layers. The input to the model is a 3-channel image, which is passed through the layers in a sequential manner. The activation function used throughout the model is ReLU. The output layer of the model uses a softmax activation function for binary classification. The model uses the cross-entropy loss function for training and backpropagation for optimization. This architecture is a commonly used approach for image classification tasks using CNNs.



6.2.1 Applying on the unmasked dataset:

Validation accuracy and loss obtained on running 20 epochs is shown below

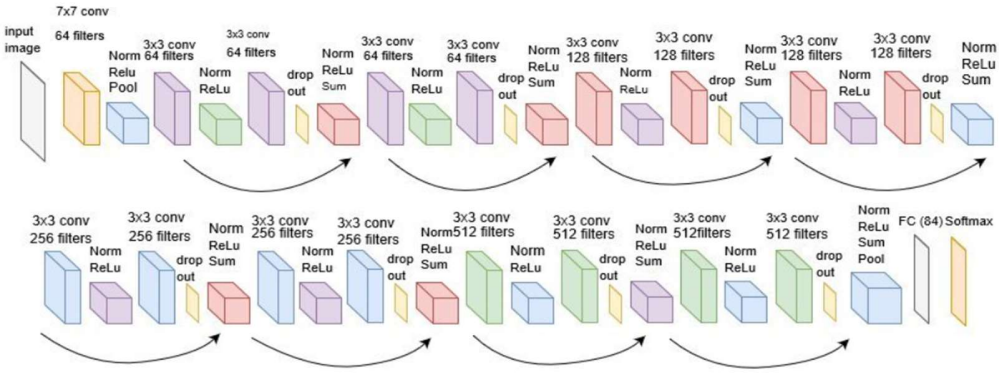


- The test accuracy of the best model : 98.99%
- The F1 score of the best model : 0.99

6.3 Resnet 18 :

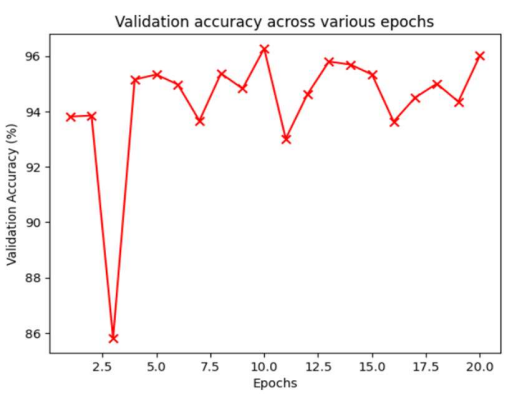
ResNet-18 is a deep convolutional neural network architecture that was introduced in 2015. It consists of 18 layers and uses residual blocks with skip connections to alleviate the problem of vanishing gradients in deep networks. ResNet-18 has been widely used for image classification tasks and has achieved state-of-the-art performance on several benchmark datasets such as ImageNet. It has also been used for transfer learning, where the pre-trained model is used for other computer vision tasks.

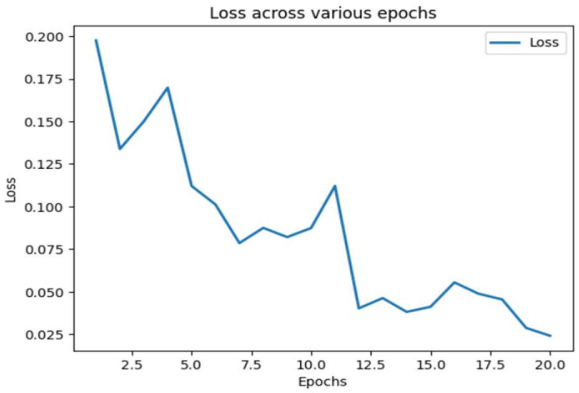
As the number of layers are less, it is computationally less expensive.We used the pre-trained Resnet-18 model with weights from imagenet to build up our framework.The model is compiled using stochastic gradient descent optimizer and sparse categorical cross entropy as loss function .



6.3.1Applying on the unmasked dataset:

Validation accuracy and loss obtained on running 10 epochs is shown below



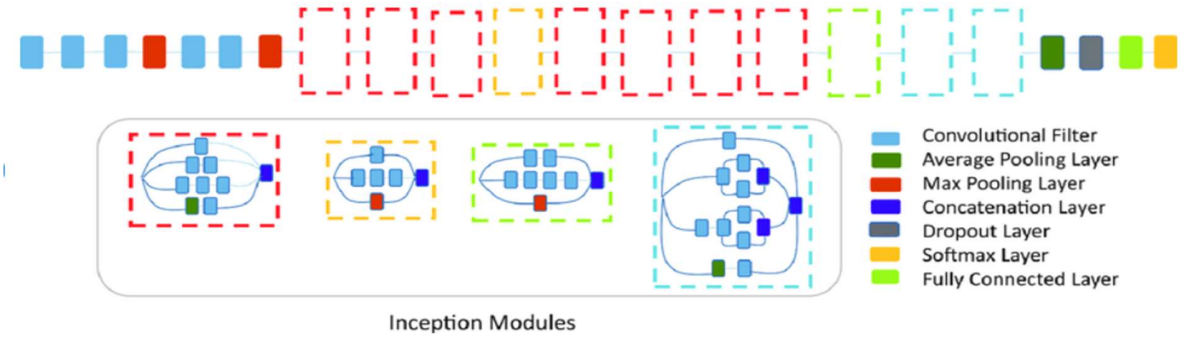


- The test accuracy of the best model : 96.02%
- The F1 score of the best model : 0.95

6.4 InceptionV3 :

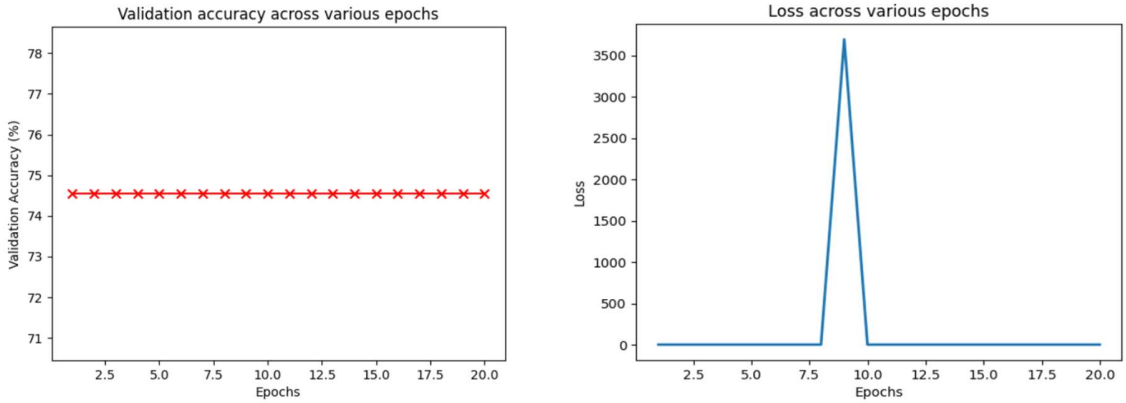
InceptionV3 is a convolutional neural network architecture that was introduced in 2015. It uses a deep network design with multiple layers and has a large number of parameters. InceptionV3 employs a unique approach to building the network by using a combination of different convolutional filters and pooling layers in parallel. This allows the network to capture features at different scales and resolutions, which can improve accuracy. InceptionV3 has been used for various computer vision tasks such as image classification, object detection, and semantic segmentation

We have taken the pre-trained InceptionV3 model



6.4.1 Applying on the unmasked dataset:

Validation accuracy and loss obtained on running 20 epochs is shown below



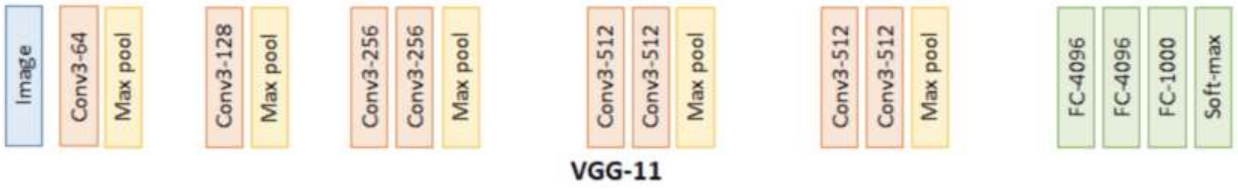
- The test accuracy of the best model : 74.87%
- The F1 score of the best model : 0.43

6.5 VGG - 11:

VGG-11 is a convolutional neural network architecture that was introduced in 2014 by the Visual Geometry Group (VGG) at the University of Oxford. It is a relatively simple and straightforward architecture, consisting of 11 layers, including 8 convolutional layers and 3 fully connected layers.

The key feature of VGG-11 is its use of small 3x3 convolutional filters, which allows it to learn more complex features than traditional larger filters. VGG-11 is also known for its uniform architecture, with each convolutional layer followed by a pooling layer and then another convolutional layer. This design pattern is repeated multiple times in the network.

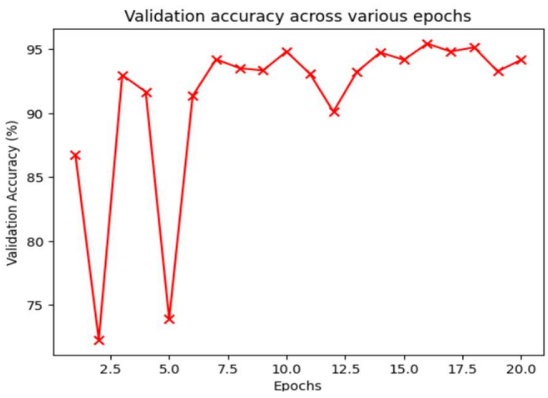
This variant is considered due to its less number of layers which reduces the time and computational expenses.We have taken the pre-trained VGG-11 model from pytorch.

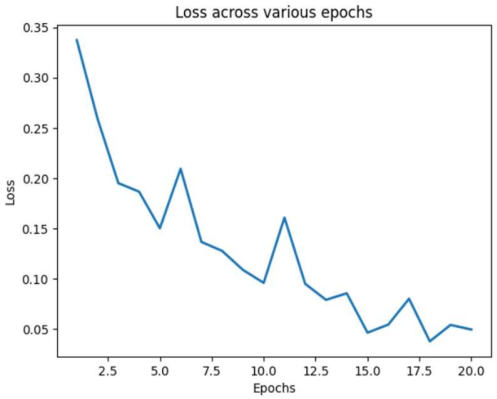


VGG-11

6.5.1 Applying on the unmasked dataset:

Validation accuracy and loss obtained on running 20 epochs is shown below





- The test accuracy of the best model : 98.55%
- The F1 score of the best model : 0.98

6.6 Alexnet :

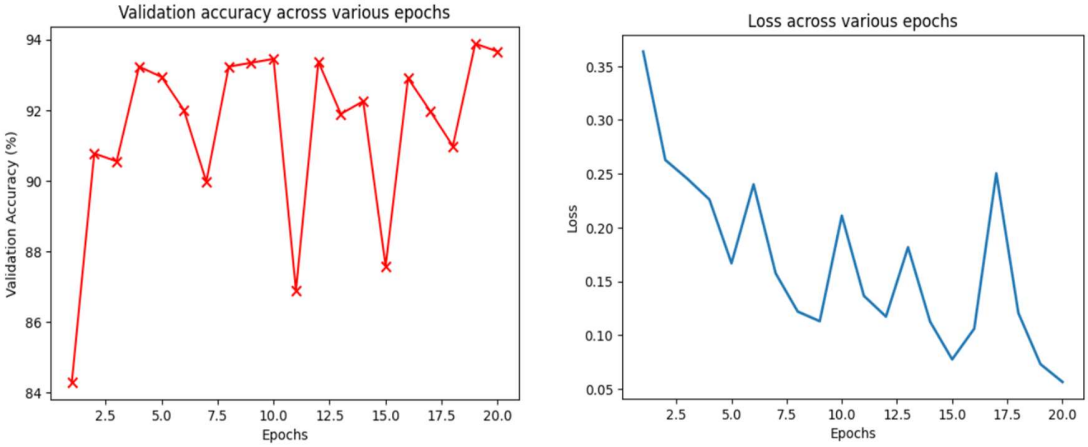
AlexNet is a convolutional neural network architecture that was introduced in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It was one of the first deep learning architectures to achieve significant improvements in image classification accuracy, winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.

AlexNet consists of 8 layers, including 5 convolutional layers and 3 fully connected layers. It is known for its use of large 11x11 and 5x5 convolutional filters, which allow it to learn more complex features than traditional smaller filters. AlexNet also introduced the concept of using ReLU activation functions instead of sigmoid functions, which allowed for faster training of deep neural networks.

We have taken the pre-trained Alexnet model from pytorch.

Applying on the unmasked dataset:

Validation accuracy and loss obtained on running 20 epochs is shown below



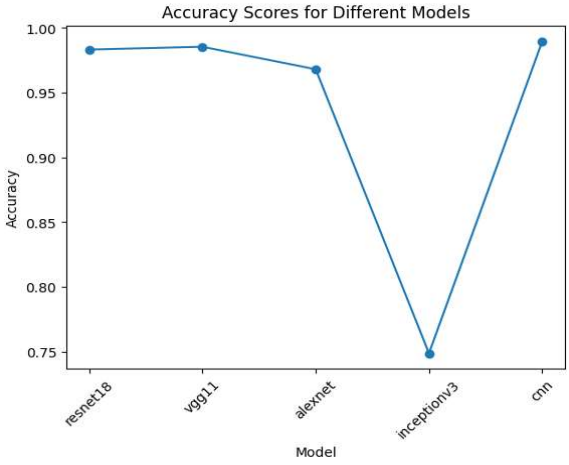
- The test accuracy of the best model : 96.81%
- The F1 score of the best model : 0.96

6.7 Comparative Analysis:

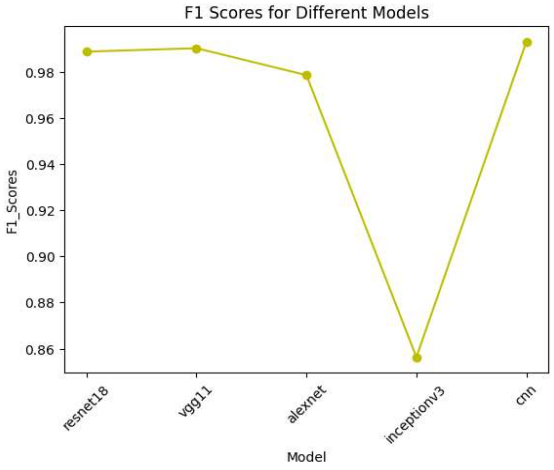
After applying all these models to our dataset, we create a dataframe for comparing all the deep learning models which can be visualized below.

Index	Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
0	ResNet18	0.983345	0.982524	0.995084	0.988764	0.972817
1	VGG11	0.985518	0.987366	0.993157	0.990253	0.978422
2	AlexNet	0.968139	0.989179	0.968223	0.978589	0.968053
3	InceptionV3	0.748733	0.748733	1.000000	0.856315	0.500000
4	CNN	0.989862	0.989226	0.997038	0.993117	0.983574

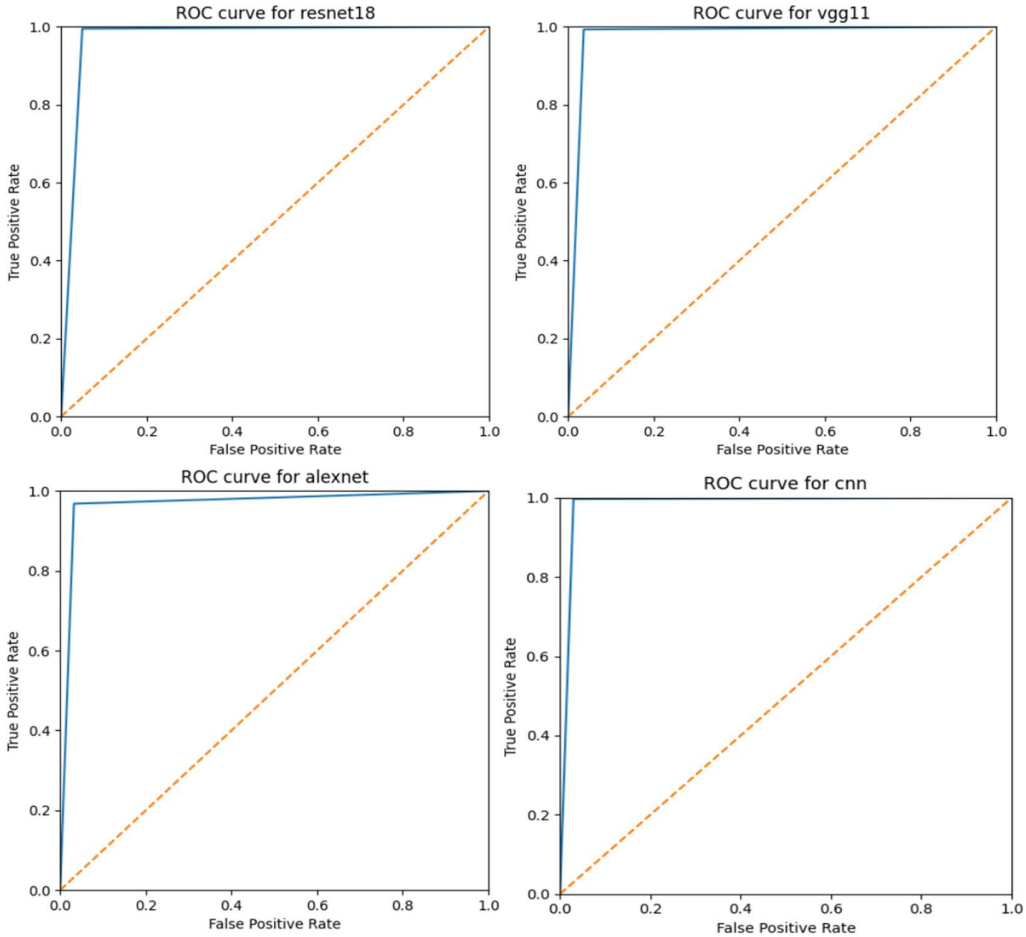
Accuracy comparison for all the deep learning models is shown below.



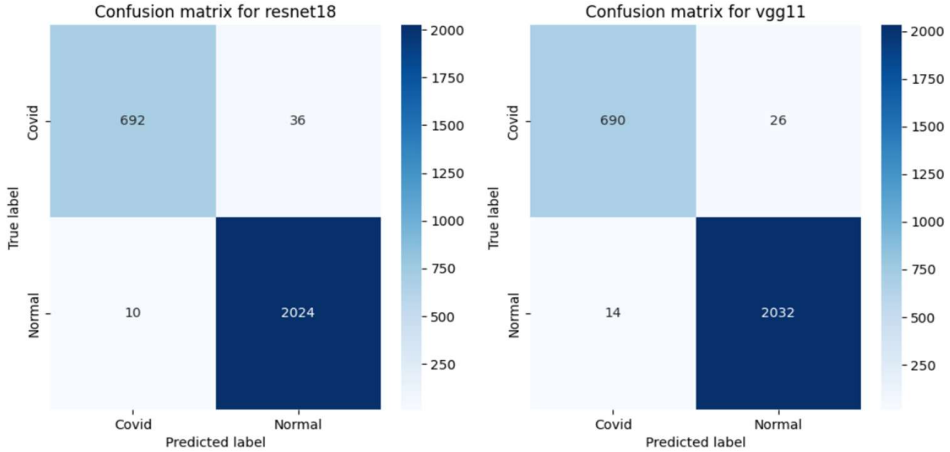
F1 Score comparison for all the deep learning models is shown below

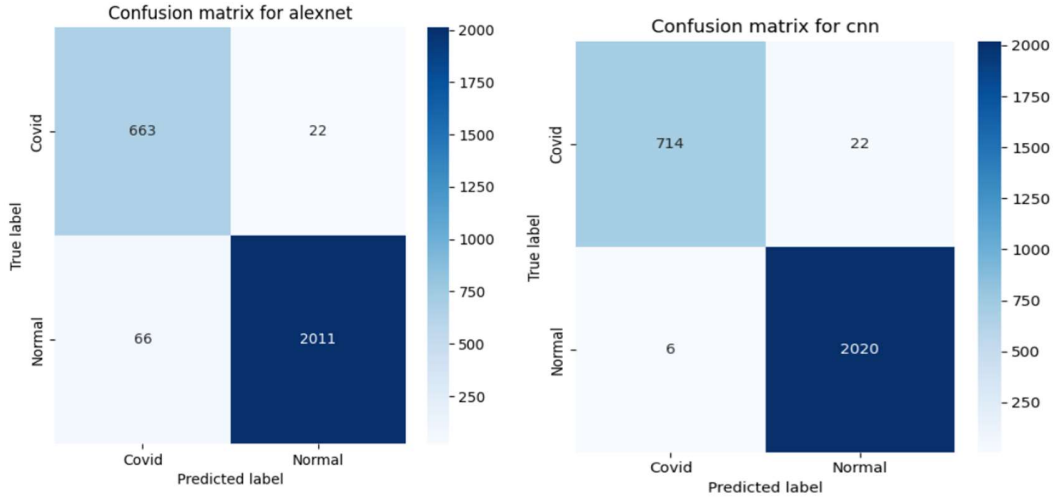


ROC curves for the best 4 models are shown below



Confusion matrices for the best 4 models are shown below.





7. Conclusion:

We can see from the confusion matrix that the CNN model with convolution layers manually added performed the best and hence overall F1 score is also highest among all the tested models.

Also the deep learning models are performing much better than the supervised machine learning models.

Best Model observed : CNN model with following layers : (module): Net(

(conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

(pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)

(conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

(fc1): Linear(in_features=100352, out_features=128, bias=True)

(fc2): Linear(in_features=128, out_features=2, bias=True))

8. Contribution:

Major of the word is done by three of us together by discussing with each other, it was hard to split which is done by whom.

Samarth Patel : Tested and deployed deep learning models like ResNet18, InceptionV3 and CNN from scratch. Also worked on other aspects of the project like dataloaders and dataset visualization.

Sai Kiran Reddy : Tested and deployed deep learning models like AlexNet and VGG11. Also worked on other supervised learning and deep learning models and model comparisons.

Ved Brahmbhatt : Tested and deployed Supervised Machine Learning models and worked along with Sai Kiran and Samarth on deep learning models. Also worked on data visualization and model comparisons.

9. References:

Alexnet architecture

InceptionV3 architecture

CNN architecture

VGG-11 architecutre

Resnet-18 architecture