Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-Ray Images

A thesis

Submitted in partial fulfillment of the requirement for the Degree of

Master of Engineering in Software Engineering

of

Jadavpur University

by

Lopamudra Roy

Registration Number: 154458 of 2020-2021

Exam Roll Number: M4SWE22016

Class Roll Number: 002011002016

Under the Guidance of

Dr. Rohini Basak

Assistant Professor

Dept. of Information Technology

Jadavpur University

June 2022

Certificate of Submission

This is to certify that the thesis entitled "**Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-Ray Images**" has been carried out by **Lopamudra Roy** (University Registration Number: 154458 of 2020-2021, Examination Roll Number: M4SWE22016) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Engineering in Software Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

…….

Dr. Rohini Basak (Thesis Supervisor)
Assistant Professor
Department of Information Technology
Jadavpur University, Kolkata-700032

Countersigned

…….

Dr. Parama Bhaumik
Associate Professor and Head
Department of Information Technology
Jadavpur University, Kolkata-700032

…….

Prof. Chandan Mazumdar
Dean, Faculty of Engineering and Technology
Jadavpur University, Kolkata-700032

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled "**Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-Ray Images**" is a bonafide record of work carried out by **Lopamudra Roy** in partial fulfillment of the requirements for the award of the degree of Master of Engineering in Software Engineering in the Department of Information Technology, Jadavpur University during the period of July 2021 to June 2022. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, the opinion expressed, or conclusion drawn therein but approves the thesis only for the purpose for which it has been submitted.

……

Signature of Examiner 1
Date:

……

Signature of Examiner 2
Date:

*Only in case the thesis is approved.

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "**Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-Ray Images**" contains a literature survey and original research work by the undersigned candidate, as part of her Degree of Master of Engineering in Software Engineering.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: LOPAMUDRA ROY

Registration Number: 154458 of 2020-2021

Examination Roll Number: M4SWE22016

Class Roll Number: 002011002016

Thesis Title: Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-Ray Images

…….

Signature with Date

Acknowledgement

On the edge of completing my 2nd year of M.E. at Jadavpur University, I look back at the past days and can't help myself from thinking how fortunate I was that I had been given an opportunity to be a part of the family of Jadavpur University. I would like to express my gratitude to everyone who played an important role in the completion of this thesis.

I feel so grateful and would like to express my sincere gratitude to my respected guide and teacher **Dr. Rohini Basak** (Assistant Professor, Department of Information Technology, Jadavpur University) for her valuable guidance and support & consider myself fortunate enough for having her as my mentor.

I would also like to thank **Prof. Chandan Mazumdar** (Dean, Faculty of Engineering and Technology, Jadavpur University) and **Dr. Parama Bhaumik** (Head of the Department of Information Technology, Jadavpur University) for providing me with all the facilities and for their support of the activities of this project.

Last but not the least, I would like to thank my parents, for always being my source of inspiration and for teaching me to work with passion and dedication, and also to all of my family members and classmates, who always believe in me and provide me all kinds of moral supports endlessly.

…….

LOPAMUDRA ROY
Registration Number: 154458 of 2020-2021
Examination Roll Number: M4SWE22016
Class Roll Number: 002011002016
Department of Information Technology
Jadavpur University, Kolkata-700032

Table of Contents

Abstract 3
Chapter 1 : Introduction 4
Chapter 2 : Literature Survey 7
2.1 Specialized CNN Methods for COVID-19 7
2.1.1 ChexNet 7
2.1.2 COVID-Net 7
2.1.3 COVID-CAPS 7
2.1.4 CoroNet 8
2.1.5 COVIDPEN 8
2.2 Deep Features Fusion and Ranking Technique 8
2.3 Existing Pre-trained CNN Models 9
2.3.1 Residual Network (ResNet) 9
2.3.2 Visual Geometric Group (VGG) 9
2.3.3 Inception 9
2.3.4 Densely Connected Convolutional Networks (DenseNet) 10
Chapter 3 : The Proposed Method 11
3.1 Tools 11
3.1.1 Tensorflow 11
3.1.2 Keras 12
3.2 The Proposed Method 17
3.2.1 The Data-sets 17
3.2.2 Performance Analysis 20
3.2.3 The Method 22
Chapter 4 : Experiments and Results 40
4.1 Comparison of the Four Architectures in 3.2.3 40
Chapter 5 : Conclusions and Future Work 53
5.1 Conclusions 53
5.2 Future Research Directions 54
References 55

List of Figures

Figure 1 : Total Case and Total Cured (By month) from Worldometer 4
Figure 2 : Total Death (By month) from Worldometer 5
Figure 3 : TensorFlow Architecture for Building Models 11
Figure 4 : Countplot of the chest Radiography Dataset 18
Figure 5 : A sample of X-ray images of all categories 20
Figure 6 : Architecture 1 23
Figure 7 : Architecture 2 27
Figure 8 : Architecture 3 30
Figure 9 : Architecture 4 34
Figure 10 : Plot for COVID Category Performance of all models using 80-20 train test split 40
Figure 11 : Plot for COVID Category Performance of all models using 75-25 train test split 41
Figure 12 : Plot for COVID Category Performance of all models using 70-30 train test split 42
Figure 13 : Result comparison for training and testing phase of all architectures 43
Figure 14 : Training loss and accuracy evaluation of all models using 80-20 train test split 44
Figure 15 : Training loss and accuracy evaluation of all models using 75-25 train test split 45
Figure 16 : Training loss and accuracy evaluation of all models using 70-30 train test split 46
Figure 17 : Confusion matrix of all deep learning models using 80-20 train test split 48
Figure 18 : Confusion matrix of all deep learning models using 75-25 train test split 50
Figure 19 : Confusion matrix of all deep learning models using 70-30 train test split 52

List of Tables

Table 1 : No of images in COVID-19 Radiography Dataset.....17
Table 2 : No of images in Chest X-Ray Images (Pneumonia) Dataset.....18
Table 3 : No of images after the combination of two Dataset.....19
Table 4 : Classification performance of all models using 80-20-20 train validation test split.....40
Table 5 : Classification performance of all models using 75-20-25 train validation test split.....40
Table 6 : Classification performance of all models using 70-20-30 train validation test split.....41
Table 7 : Classification accuracy of all tested deep learning models on a GPU 43

ABSTRACT

The novel coronavirus (COVID-19) came up in Wuhan in December 2019. This deadly COVID-19 pandemic has become very fast-spreading and is currently present in several countries worldwide. A major reason for the widespread of COVID-19 is the slow pace of testing or lack of ability to detect the disease at the earliest. The sudden increase in COVID-19 patients is a major shock to our global health care systems. With the limited availability of test kits, all patients with respiratory infections can't be tested using RT-PCR. Testing also takes a long time. The early detection of COVID-19 from Chest X-Ray images could be helpful to separate all those infected patients who are at high risk while awaiting test results. Nowadays X-ray machines are generally available in most health care systems, and several modern X-Ray systems could be installed on the computer, so there is no travel time involved for this method. In this paper, we propose the use of chest X-Ray images to prioritize the selection of infected patients for further RT-PCR testing because our current systems have difficulty deciding whether to keep the patient in the ward with other patients or isolate them from COVID-19 areas. So with the help of X-Ray images, we can prevent this situation in hospitals. It also helps to identify patients with a high risk of COVID with false-positive RT-PCR which requires repeated testing.

In this thesis, a dataset of X-Ray images from COVID-infected patients, common pneumonia, lung_opacity, and normal images has been used to automatically detect the multiclass image classification. The main aim of this study is to evaluate the performance of Convolutional Neural Network (CNN) architectures for medical image classification. Specifically, the procedure referred to as transfer learning is used here.

Using the transfer learning method, the detection of several diseases in medical image datasets is an accomplishable target, usually yielding outstanding results. The dataset which is used during this work is a collection of 27022 X-Ray images having four categories. The data is collected from the public medical repositories. With the help of transfer learning, in this multiclass image classification problem, the highest accuracy of 97% is achieved on training data and 96% on testing data.

Keywords: Covid-19; Medical Images; Multiclass Image Classification; Transfer Learning; Convolution Neural Network

Chapter 1

Introduction

Covid-19 is an extended family of respiratory viruses that may cause several diseases, from the common cold to respiratory infections like as MERS (Middle East respiratory syndrome) and SARS (Severe acute respiratory syndrome).

These kinds of viruses are common in many animal species (such as camels and bats) but rarely in some cases, they can evolve and affect humans and then spread to the population. A new coronavirus strain that has never previously been identified in humans is the one that appeared at the end of 2019 i.e., the 2019 novel coronavirus (COVID-19, acronym of COronaVIrus Disease 19). The first cases were found during the COVID-19 pandemic of 2019-2020, which started around the end of December 2019 in the city of Wuhan, the capital of the Chinese province of Hubei, and later spread to many countries of the world. Figure 1 and 2 describe the total case, total cure and total death by COVID-19 from Worldometer data.

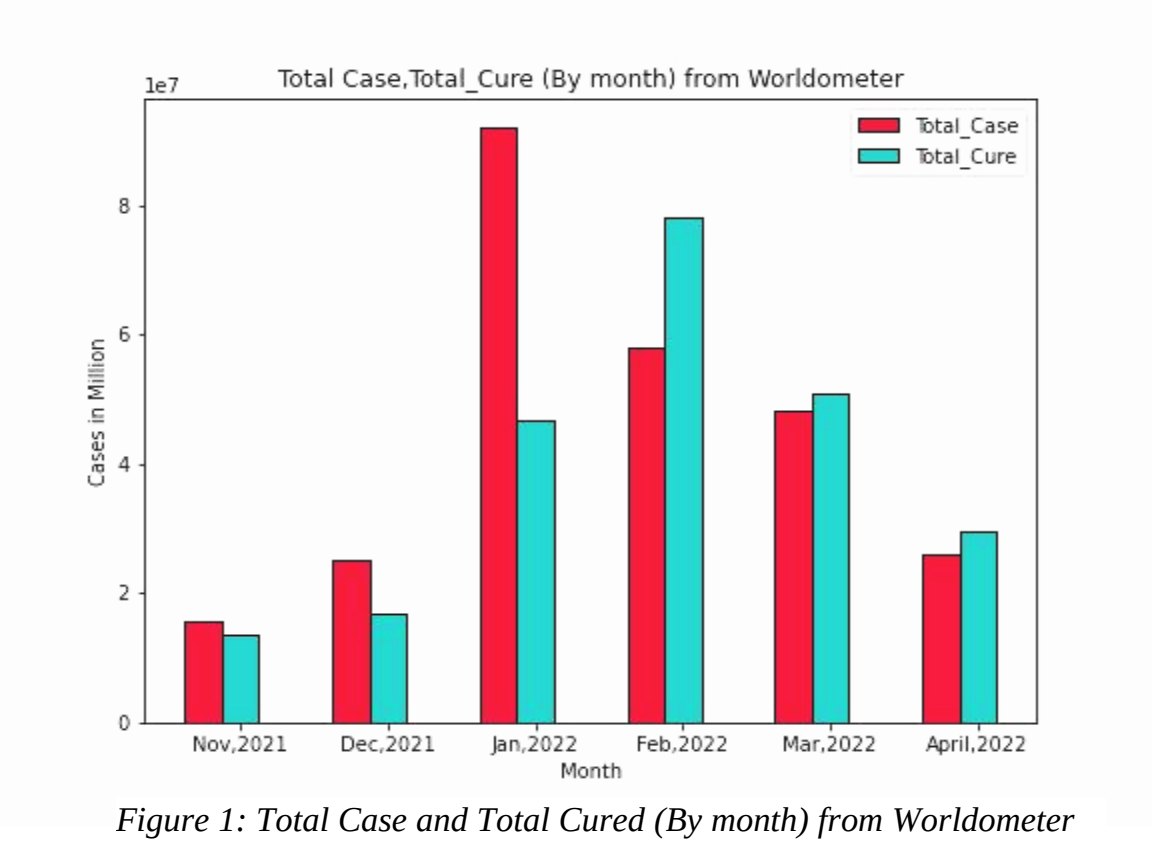


Figure 1: Total Case and Total Cured (By month) from Worldometer

Figure 2 : Total Death (By month) from Worldometer

Due to this sudden increase in COVID-19, several countries are currently facing a shortage of resources to battle this pandemic. In many lands, health care systems are in crisis, there are limited diagnostic kits, limited hospital beds for admission to such patients, limited medical equipment (PPE) for healthcare workers, and limited ventilators. It is therefore important to differentiate which patients with severe respiratory infections (SARI) may have COVID-19 infection to make the best use of the prescribed resources. Currently, the most effective tool for identifying the infected patients is Reverse transcription-polymerase Chain Reaction (RT-PCR) testing. RT-PCR tests are utilized to directly sight the presence of the COVID-19 antigen, rather than the body's invulnerable reaction, or antibodies. These tests can detect whether or not someone has this virus very early on, as they help to detect the viral RNA, which should be present before symptoms begin to appear or antibodies are formed in the body. However, this is a very complicated manual process that is extensively laborious, time-consuming, and also expensive. It also has a high false-negative rate that indicates the predicted result is negative but the actual result is positive.

Hence, Radiogram images (e.g. X-ray or CT scans) of the Chest are the next best alternative for detecting COVID-19. X-ray images display peculiar characteristics for COVID-19. The Chest X-rays replicated a certain pattern or opacities. To manually find such patterns, experienced and

adept radiologists are required. But, this is not practical or ideal due to the drastically increasing number of cases and less number of expert radiologists. The most logical solution to this is an automated method to detect such patterns or anomalies, which would help in diagnosing the patient at the earliest with increased accuracy.

Recently, deep learning and artificial intelligence have played a major role in medical fields like medical image classification, image processing, computer-aided diagnosis, image fusion, image interpretation, and image segmentation. In this thesis paper, we used the transfer learning method, which is quite popular in the field of image classification and NLP. Transfer learning is all about leveraging feature representations from a pre-trained model, therefore we don't have to train a whole model from the scratch. Pre-trained models are already trained on millions of images, therefore it saves a lot of computational power and training time. The weights that are obtained from the pre-trained models can be reused in other computer vision tasks. These models can be reused directly for making predictions on new tasks or integrated into the process of training a new model. Here we used 27022 X-Ray images in four categories. The main problem is, now a day the image size is very big, so if we directly used these images for our classification, it would be a very expensive computation and the training time will be very long. Therefore, using the advantages of Convolution Neural Networks, it is possible to reduce the image dimension and keep all the important features that will help for the good prediction.

The main aim is to train some significant convolutional neural network architectures, such as EfficientNet, ResNet50, VGG16, and VGG19, Xception on our data set, to detect COVID-19, Pneumonia, and Lung-opacity from chest X-ray images. We used fine-tuning to build a new fully-connected layer or to change some last few layers for our new problem.

Chapter 2

Literature Survey

2.1 Specialized CNN Methods for COVID-19

Many researchers developed some CNN architecture, especially for COVID-19 detection. All these CNN architectures have more capabilities to classify the images into different categories, therefore that result will be more accurate for this purpose rather than standard CNN architecture. In the primary stage, these models are trained on ImageNet data after that trained on different diseases.

2.1.1 ChexNet

In 2017, Rajpurkar et. al introduced a network called ChexNet. It is a 121-layer convolutional neural network. It was designed and trained to show symptoms of 14 different diseases such as mass, nodule, pneumonia, etc. The network is DenseNet121 with adjusted final layers which classify 14 diseases. The model ChexNet takes an X-Ray image as an input and produces output on the possibility of pneumonia and a heatmap in which regions are formed to identify the most likely areas of pneumonia or other thoracic illnesses. This CNN model was later specially used to diagnose COVID-19 [1,2].

2.1.2 COVID-Net

This CNN architecture is specially designed for the diagnosis of COVID-19 from chest X-Ray images. Therefore it has high architectural diversity and also selective long-range connectivity. In this COVID-Net architecture, the pattern projection-expansion-projection is used massively. COVID-Net architecture is incorporated into a heterogeneous mix of convolution layers. This proposed COVID-Net was pre-trained on the ImageNet dataset and after that, it is applied to the COVIDx dataset. Using this architecture, they got an accuracy of nearly 93.3% [3].

2.1.3 COVID-CAPS

This is basically a capsule-based architecture introduced by [4]. This model has four convolutional layers and three capsule layers. The input of this CNN architecture is 3-dimensional chest X-Ray images. The first layer is a convolutional layer, after that batch-normalization is applied. The second layer is also a convolutional layer, followed by a pooling layer. Similarly, the third and fourth layers are convolutional, and the fourth layer is reshaped as the first Capsule layer. All three Capsule layers are embedded to perform the routing and the last Capsule layer have the parameters of the two classes that are positive and negative COVID-19. The Total trainable

parameter is 295,488 for the COVID-CAPS model and using this model the researchers have 98.3% accuracy.

2.1.4 CoroNet

CoroNet[5] is a CNN architecture that is specially developed for the diagnosis of COVID-19 from chest X-ray images. CoroNet is based on Xception architecture which stands for the Extreme version of Inception. This is 71 layers deep CNN architecture trained on the ImageNet dataset. CoroNet uses Xception as a base model with a dropout layer and two dense layers added at the end of this model. This network has total 33,969,964 parameters, out of those parameters, 33,969,964 trainable and 54528 are non-trainable. With the help of this architecture, the researchers got 89.6% accuracy for four-class classification, 95% accuracy for three-class classification, and 99% accuracy for binary classification.

2.1.5 COVIDPEN

This model builds upon the original EfficientNet architecture which consists of 18 convolutional layers where each layer is equipped with a filter of dimensions (1,1), (3,3), and (5,5). This model is adopted by transfer learning on the pruned EfficientNet-B0 model, where it outputs the biases and weights and is used to classify the inputted frontal-view chest X-ray or CT images. Using this model the researchers have 96% accuracy on X-ray images and 85% accuracy on CT-Scan images for binary classification problems [6].

Apart from this architecture, some researchers used Multilevel Thresholding, COVID-ResNet [7], CovXNet [8], etc.

2.2 Deep Features Fusion and Ranking Technique

Reference [9] uses VGG-16, GoogleNet, and ResNet-50 models were used for feature extraction. The obtained feature vectors with these models were fused to obtain higher dimensional fusion features. As there is a certain level of correlation and excessive information among the features because of this it also increases consuming time and computational complexity. So, to rank the features, the t-test technique was used. It calculates the difference between the two features and determines their differences statistically. After the feature fusion and ranking functions were performed, the binary SVM classifier was trained for classification.

Reference [10] uses a fuzzy rank-based fusion of different CNN base models using the Gompertz function. To generate the initial decision scores three standard CNN models are used, VGG-11, Wide ResNet-50-2, and Inception v3, and a novel ensemble technique has been used to fuse the decision scores of the models. The researchers have proposed a Fuzzy ranking method using the Gompertz function here. The main advantage of such fusion is that it uses adaptive weights based on the confidence scores of each classifier used to form the ensemble in order to generate the final prediction of each sample.

2.3 Existing Pre-trained CNN Models

Most of the researchers used different pre-trained CNN architecture for COVID-19 diagnosis. There are so many pre-trained CNN models but the most commonly used are Residual Network (ResNet), Visual Geometry Group (VGG), Densely Connected Convolutional Network (DenseNet), Inception Network, etc. Some of the most commonly used existing pre-trained CNN models are described here.

2.3.1 Residual Network (ResNet)

Resnet is a Convolutional Neural Network architecture that is mainly designed to train hundreds or thousands of convolutional layers. Previously, CNN architectures had a drop off in the effectiveness of each additional layer, but ResNet can add a more number of layers with strong performance. ResNet first introduced the concept of skip connections. i.e. to fit the input from a previous layer to the next layer without changing the input. Deep networks are hard to train because of the vanishing gradient problem, but with the help of a skip connection, Resnet can solve the problem of vanishing gradient. It has different variants (ResNet18, ResNet169, ResNet50, ResNet152, etc.) and the image input size of this network is 224 X 224. ResNet is the architecture that is mostly used for both CT and X-Ray based research. Many authors [11,12,13] used ResNet in their proposed models for the diagnosis of COVID-19.

2.3.2 Visual Geometric Group (VGG)

Visual Geometric Group (VGG) is one of CNN's most important architectures. The VGG network has 16 or 19 convolutional layers and it is very commonly used because of its uniform structure. It has a 224 x 224 input image format. The most important thing about VGG is, that instead of having a large hyperparameter, it has only convolution layers of 3x3 filter with stride 1 and always used same padding to make the same dimensions of the input image and output image. The work proposed in [14,15] used VGG for COVID-19 detection purpose.

2.3.3 Inception

It is a transfer learning method that is very popular among researchers [15,16]. This consists of two parts: extracting features from images with the help of CNN and classification with softmax and fully connected layers. There are many versions of this network. The popular ones are InceptionV1, InceptionV2, InceptionV3, and InceptionV4. It has a 299 x 299 input image format.

2.3.4 Densely Connected Convolutional Networks (DenseNet)

Dense Convolutional Network was first introduced in 2016 by Huang et al [17]. This is another very deep neural network having dense blocks. The name of the network has come from the fact of its dense connectivity. It is quite similar to ResNet architecture but has some differences. It has a very simple architecture to ensure maximum information flow between layers in the network. By matching feature map sizes in the network, they connected all the layers directly to all of their subsequent layers - a Densely Connected Neural Network, simply known as DenseNet. The main difference from other networks like ResNet and all, is that DenseNet does not sum the output feature maps of the layer with the incoming feature maps. It just concatenates them. It has different types of versions (DenseNet101, DenseNet169, DenseNet201, etc.).It has a 224 x 224 input image format. Many researchers [15,18,19] used this architecture to diagnose COVID-19.

Chapter 3

The Proposed Method

Before describing several deep neural network architectures for our multiclass classification problem, a brief description of the tools and libraries that have been used in the work are presented.

3.1 Tools

The tools used are described in the following subsections:

3.1.1 Tensorflow

Before exploring Keras, Tensorflow should be discussed in detail. TensorFlow is a software library or framework, designed and developed by Google for the implementation of machine learning and deep learning applications. It combines computational algebra of optimization techniques for the easy calculation of many mathematical expressions. This library runs on CPU, GPU, or IoT processors. The initial public version was released in 2015, however, the stable version was released in 2017 under the Apache Open Source License. TensorFlow is also called a "Google" product. Tensorflow has a variety of machine learning and deep learning algorithms. It can train and run deep neural network architectures for image recognition, handwritten digit classification, word embedding, creation of various sequence models, and many more.

The Tensorflow architecture consists of three parts. Figure 3 describes the architecture of Tensorflow.

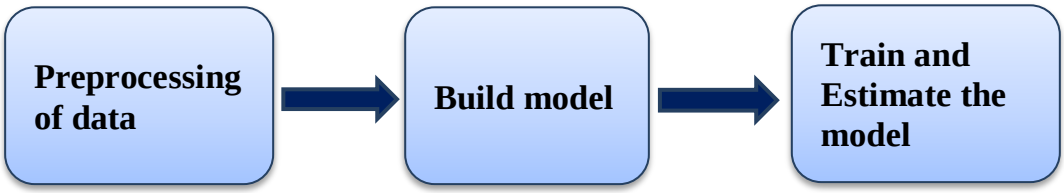


Figure 3 : TensorFlow Architecture for Building Models

At first, the input is a multidimensional array fed into Tensorflow, and these arrays are called tensors. All these tensors go through various operations and then produce output.

Developers mostly used tf as a notation to mention the Tensorflow. This has two phases. The first one is the Development phase and the last one is the Run phase. In the development phase, the data is trained and it is performed by high-performance workstations like laptops or PCs with the help of GPUs. The Running phase can be achieved by any kind of machine including PCs, laptops, cloud, etc.

3.1.2 Keras

Keras is a python based neural network API which runs on top of open-source machine libraries like TensorFlow, Theano, or Cognitive Toolkit (CNTK). Keras is mainly based on a minimal structure which provides a very clean and easy way to create the deep learning models based on TensorFlow or Theano. It is designed to define deep learning models and this is an optimal choice for the applications related to deep learning. In python, developers use Keras with the notation tf.keras.

Let's discuss the architecture of the Keras framework and how it can be helpful in our thesis work.

Keras API is divided into three main categories:

- Core Modules
- Model
- Layer

3.1.2.1 Keras Modules

Keras modules contain many predefined classes, variables, and functions. These are useful for algorithms in deep learning. Some of the modules are provided below.

- Initializers: It provides a list of initializer functions and in Keras, initializers are used to set the weight of each input.
- Constraints: It provides a list of functions to restrict and specify the range in which the weight of input data is generated.
- Regularizers: Try to optimize the layer by setting some penalties during the optimization phase. This module provides functions like L1 and L2 regularizer, etc.
- Activations: Transform the output to make it non-linear. This module provides a list of activator functions like Sigmoid, Relu, Softmax, etc.
- Losses: This module provides a list of loss functions like mean_squared_error, mean_absolute_error,mean_absolute_percentage_error, etc.
- Metrics: It is used to evaluate the performance of our model. Keras module provides a few metrics as a module, for example, accuracy, binary_accuracy, categorical_accuracy, sparse_categorical_accuracy, etc.
- Optimizers: Optimize the input weights by comparing the prediction and the loss function. This module provides several optimization functions like Adam, SGD, Rmsprop, etc.
- Callback: Used during the training phase to print the intermediate data and also to stop the training itself (Early Stopping method) based on some specific condition.
- Image processing: Keras has some functions to convert images into a NumPy array suitable for further processing. It is also used in the data preparation phase of machine learning or deep learning application.

- Text processing: Provides some functions to convert text into a NumPy array suitable for further processing. It is generally used in the data preparation phase of machine learning or deep learning application.
- Backend: Keras has some of the backend libraries like TensorFlow and Theano.By default, Keras uses TensorFlow as the backend.

Out of all these modules, we used Activation functions, Losses, Metrics, Optimizers, and Callback for stopping the training phase.

3.1.2.2 Keras Layers

Dense Layer: All layers are fully connected (dense) by the neurons in a network layer. Each neuron in one layer receives input from all the neurons in the previous layer.

In keras a dense layer has following signature:

```
Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
    )
```

The operation of the Dense Layer is, output = activation(dot(input, kernel) + bias) where activation represents the activation function, the dot represents the dot product of all its input and its corresponding weights, the kernel here is a weights matrix created by this layer and bias represents the bias vector. Bias is only applicable if use_bias is True.

Dropout Layer: This layer is used to fix the over-fitting issue by ignoring units (i.e. neurons) during the training phase of a certain set of neurons. If there is any unwanted data present in the input data (Noise), the Dropout layer will try to remove these noise data and prevent the overfitting problem.

The dropout layer in Keras has the following signature:

```
Dropout(rate, noise_shape=None, seed=None, **kwargs)
```

Where the rate represents the fraction of the input unit to be dropped. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, to prevent the overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Convolution Layers: These layers are the main building blocks of convolutional neural network architecture. Convolution layers have some properties which are listed below. These properties differentiate it from other layers.

- Filters: This represents the number of filters applied in the convolution. Based on the filter size the dimension of the output is changed.
- kernel size: This represents the length of the convolution window.
- Strides: In convolution, operation strides represent the length of the stride.
- Padding: Padding refers to the number of pixels added to an input image by the kernel in the convolution operation**.** There are three different types of padding which are as follows:
    - valid padding: No zero padding outside the edges when we apply the pooling.
    - causal means causal convolution.
    - same padding: Same padding refers to zero padding. The dimension of output should be the same as the input.
- Dilation Rate: Dilation rate to be applied for dilated convolution.

Following are all the convolution layers provided by Keras :

Conv1D: Conv1D is generally used for sequences. In Keras, a Conv1D layer has the following signature:

```
keras.layers.Conv1D (
        filters,
        Kernel_size,
        strides=1,
        padding='valid',
        data_format='channels_last',
        dilation_rate=1,
        activation=None,
        use_bias=True,
        kernel_initializer='glorot_uniform',
        bias_initializer='zeros',
        kernel_regularizer=None,
        bias_regularizer=None,
       activity_regularizer=None,
        kernel_constraint=None,
       bias_constraint=None
       )
```

Conv2D: Conv2D is generally used for images. In Keras, a Conv2D layer has the following signature:

```
keras.layers.Conv2D (
        Filters,
```

```
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
   bias_constraint=None
   )
```

Flatten Layers: Flatten layers are used for converting the multi-dimensional tensors/arrays into a one-dimensional array**.** When the flatten layer is applied to a layer that has an input shape as (batch_size, 2,2), then the output shape will be (batch_size, 4). Flatten has one argument as follows:

```
keras.layers.Flatten(data_format=None)
```

Here the optional argument is the data_format and it is only used for preserving weight ordering when switching from one data format to another data format.

Normalization Layer: Using the normalization techniques we can decrease the model's training time. Some of the benefits of using Normalization are:

- This layer helps to normalize each and every feature so that it maintains the contribution of every feature. So it helps to make our network unbiased because some features have bigger numerical value than others.
- Reduces Internal Covariate Shift. It represents the change in the distribution of network activations due to the change in network parameters during the training period. The training can be improved by reducing the internal covariate shift.
- Batch Normalization helps to make the loss surface smoother.
- Makes optimization faster because normalization does not allow weights to explode all over the place and restricts them to a certain range.
- It mainly helps the network in Regularization.

Batch-Normalization in keras has the following signature:

```
BatchNormalization(
    axis=-1,
    momentum=0.99,
```

```
epsilon=0.001,
center=True,
scale=True,
beta_initializer="zeros",
gamma_initializer="ones",
moving_mean_initializer="zeros",
moving_variance_initializer="ones",
beta_regularizer=None,
gamma_regularizer=None,
beta_constraint=None,
gamma_constraint=None,
renorm=False,
renorm_clipping=None,
renorm_momentum=0.99,
fused=None,
trainable=True,
virtual_batch_size=None,
adjustment=None,
name=None,
**kwargs
)
```

Pooling Layer: It usually does the max pooling operation on temporal data.MaxPooling1D function has the following signature:

```
keras.layers.MaxPooling1D
(pool_size=2,
strides=None,
padding='valid',
data_format='channels_last'
)
```

The pool_size here refers to max-pooling windows and strides refers to the factors for the downscale.MaxPooling3D and MaxPooling2D both are used for the operations of max pooling for spatial data.

Merge Layer: This layer is used for merging a list of inputs. This layer supports add(), subtract(),multiply(),average(),minimum(),maximum(), concatenate(), dot() functionalities. Some of the examples are given below.

- Adding a layer: Used to add two layers. The syntax is given below:

  ```
  keras.layers.add(inputs)
  ```

- Subtracting a layer: Used to subtract two layers. The syntax is given below:

  ```
  keras.layers.subtract(inputs)
  ```

3.2 The Proposed Method

The first sub-section of this section describes the data sets. It also explains our modifications to the data sets. The subsequent sub-section describes the performance parameters we use to compare various deep-neural architectures. Next, the different Convolution neural network architectures for our multiclass image classification are described.

3.2.1 The Data-sets

Two data sets are considered in the present work-

1. COVID-19 Radiography Dataset
2. Chest X-Ray Images (Pneumonia)

The COVID-19 Radiography Dataset [22] has four classes, Normal, Lung-opacity, Covid, and Pneumonia. Table 1 shows the number of images in each class.

Table 1: No of images in COVID-19 Radiography Dataset

Image Category	***No of images***
Normal	*10202*
Lung opacity	*6012*
Covid	*3616*
Pneumonia	*1345*

This dataset contains a total of 21165 X-Ray images for different categories. The description of all data sources of each category is given below.

- COVID-19 data: COVID data are collected from the different publicly accessible datasets, online sources, and published papers.

  -2473 CXR images are collected from the padchest dataset.

  -183 CXR images from a Germany medical school.

  -559 CXR image from SIRM, Github, Kaggle & Tweeter

  -400 CXR images from another Github source.

- Normal images: 10192 Normal data are collected from three different datasets.

  -8851 RSNA

  -1341 Kaggle
- Lung opacity images: 6012 Lung opacity CXR images are collected from the Radiological Society of North America (RSNA) CXR dataset.
- Viral Pneumonia images: 1345 Viral Pneumonia data are collected from the Chest X-Ray Images (pneumonia) database.

The countplot of the Chest Radiography Dataset is presented by Figure 4.

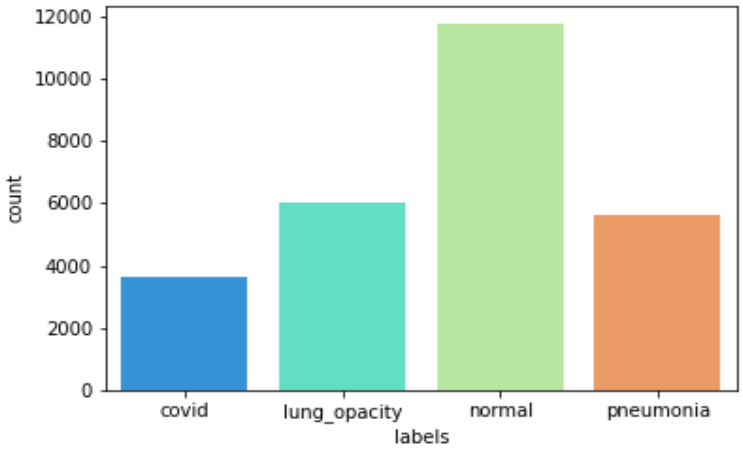


Figure 4 : Countplot of the chest Radiography Dataset

The second dataset (Chest X-Ray Images (Pneumonia) [23]) is divided into three parts, the train data-set, the validation data-set, and the test data-set. Each of these parts is further divided into two categories, Normal and Pneumonia. Total number of images in the Chest X-Ray (Pneumonia) Dataset is presented in Table 2.

Table 2 : No of images in Chest X-Ray Images (Pneumonia) Dataset

Dataset	***Image Category***	***No of Images***
Training	*Normal* *Pneumonia*	*1341* *3875*
Validation	*Normal* *Pneumonia*	*8* *8*
Testing	*Normal* *Pneumonia*	*234* *390*

This dataset contains a total of 5856 X-Ray images (JPEG). Out of all these images, 4273 images belong to the Normal category and 1583 images belong to the Pneumonia category. The Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care.

3.2.1.2 Data Pre-processing

Here we combined the two datasets. One is the COVID-19 Radiography Dataset and the other one is Chest X-Ray Images (Pneumonia) datasets. The Pneumonia images in the Chest X-Ray Images (Pneumonia) dataset are added to the pneumonia category in COVID-19 Radiography Dataset and similarly, the Normal images are added to the Normal category. So, our final dataset contains a total of 27022 X-ray images. Table 3 shows the number of images in each class of our final dataset.

Table 3 : No of images after the combination of two Datasets

Image Category	***No of Images***
Normal	*11775*
Lung opacity	*6012*
Pneumonia	*5619*
Covid	*3616*

All X-ray images have been collected after the combination of two datasets and then loaded for scaling at a fixed size of 224 X 224 pixels to be suitable for further processing within the deep learning pipeline.

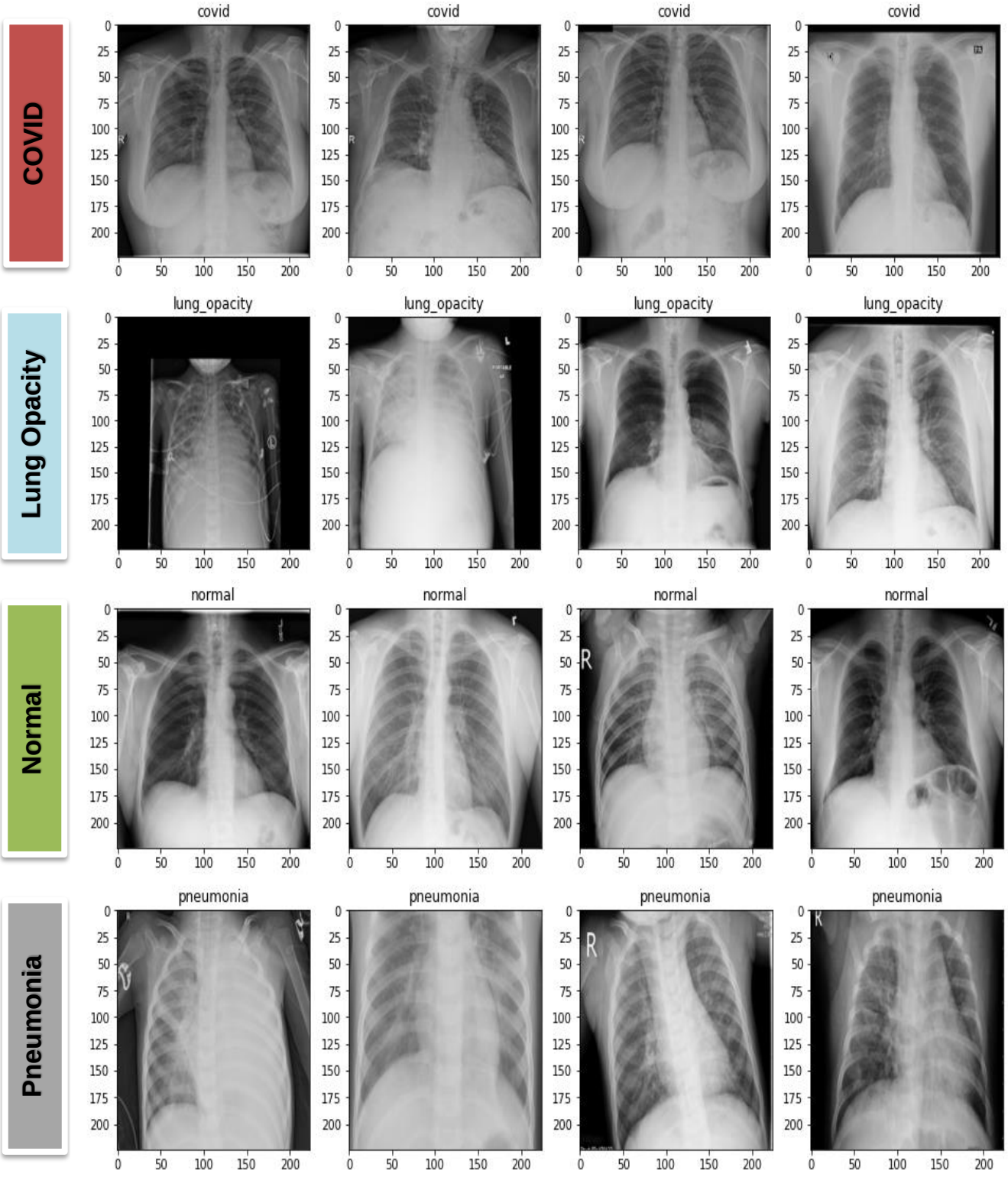


Figure 5 : A sample of X-ray images of all categories

3.2.2 Performance Analysis

To evaluate the performance of every deep learning model, several metrics are used in this thesis to detect the true or misclassification in the tested X-ray images. A detailed overview of the evaluation metrics is as follows**:**

1. Accuracy: It indicates how often predictions match the labels.

$$Accuracy = \frac{Correct\ classifications}{Total\ number\ of\ Samples}$$

2. Precision: It signifies what proportion of positive identifications was actually correct.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

3. Recall: It signifies what proportion of actual positives was identified correctly.

$$Recall = \frac{True\ positives}{True\ positives + False\ negetives}$$

4. F1 Score: It is the harmonic mean of precision and recall. The harmonic mean is often used instead of a simple average because it penalizes extreme values. This metric is used in situations where we want to find the optimal blend of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

3.2.3 The Method

In this thesis, a multiclass image classification technique for chest X-rays is proposed. Here our main goal is to identify those X-ray images which are infected by COVID. In this paper, much research related to the binary image classification of COVID-19 is present, but finding work that is related to the multiclass classification of COVID-19 is still difficult. Multiclass classification of images related to X-rays is still a challenging task as there are inter-class similarities and also the availability of datasets is one of the major issues. The datasets are also high-class imbalanced which are publicly available. Therefore, while working on multiclass classification, it is one of the other challenges to deal with. Different pre-trained models have been used in this work to evaluate their performance on chest X-rays for the classification of different diseases, including COVID-19.

3.2.3.1 Training Model and Validation

In order to start the training phase of selected and/or tuning one of the pre-trained deep learning models, first, we set the train-test split ratio **80-20** (17294 images used for training, 4323 images used for validation and 5405 images used for testing**)** according to the Pareto principle. After that, we set the train-test split ratio to **75-25** (16213 images used for training, 4053 images used for validation and 6756 images used for testing) and lastly **70-30** (15132 images used for training, 3783 images used for validation and 8107 images used for testing). For all splitting, 20% of the training data will be used for constructing validation sets. Subsample random selections of training image data for the deep learning classifier, and then apply evaluation metrics to show the recorded performance.

3.2.3.2 Classification

In the final step of the proposed work, the testing data is fed to the tuned deep learning classifier to categorize all the X-ray images into one of four cases: positive COVID-19 or Pneumonia or Lung-opacity, or normal case (negative COVID-19). At the end, the overall performance analysis for each deep learning classifier will be evaluated based on the metrics described in the following section.

For all the architectures we follow the same basic configurations mentioned as follows:

- Train-Test split ratio = 80 : 20 , 75 : 25 , 70 : 30
- Validation split = 0.2 of train data-set (part of training data-set is used for validation)
- Batch Size = 64
- Data Scaling Size = 224 x 224
- Epochs = 15
- Set some callback by monitoring the validation loss
- Early stopping (set patience = 3 or 5)
- Reduce the Learning rate by a factor 0.5

3.2.3.3 Pre-Trained Deep Learning Networks (Feature Extractor)

In order to train the model, pre-trained deep learning models have been used. Here, transfer learning has been used and this stage acts as a feature extractor for the proposed model. Also, the weights of pre-trained models are non-trainable, therefore these remain unchanged during training. The top layers of these pre-trained models have been excluded, so the output from this stage serves as input for the next stage. The classifier is built on top of this model and has been discussed in the next sub section. The pre-trained deep learning models which are used are as follows:

3.2.3.3.1 Architecture 1

In the first architecture, Resnet50 pre-trained model is used and the weights of this pre-trained model were trained on a huge dataset of images (Image Net). Fine-tuning is applied to change some last few layers for our new problem. The proposed model is presented by Figure 6.

For this architecture we follow the below configurations.

- Set patience = 3 for early stopping
- Optimizer = Adam
- Learning Rate = 0.001

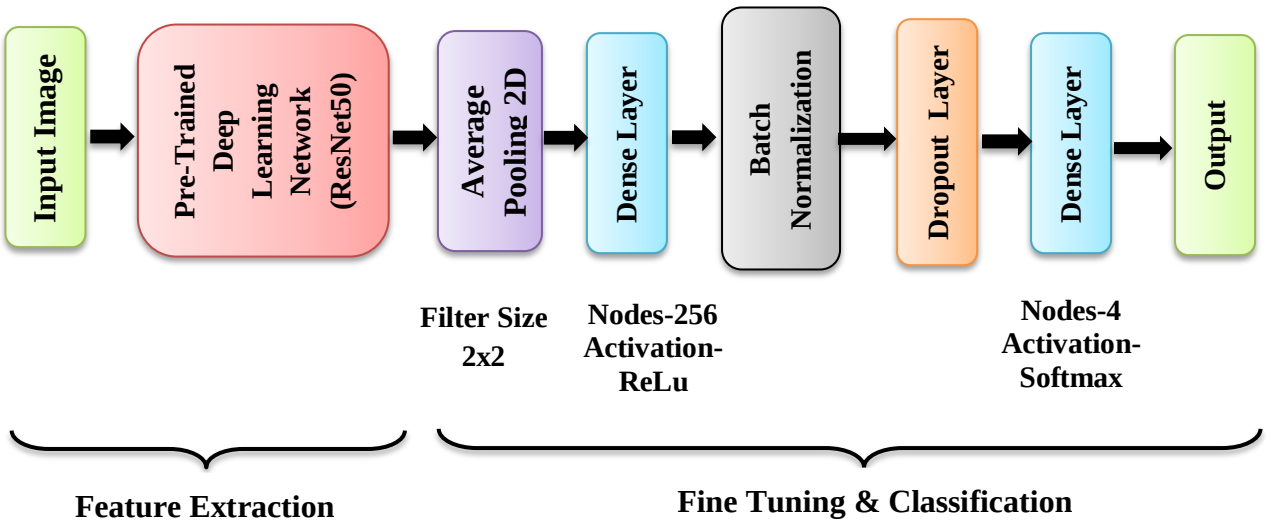


Figure 6 : Architecture 1

After the compilation we have trained the model using 80 : 20 , 75 : 25 and 70 : 30 train test split and 20% of the training data are used for validation. The snapshot of training this model using different train-test split is given below.

I. Using 80-20-20 train-validation-test split

```
Epoch 1/15
271/271 [==============================] - 5622s 21s/step - loss: 0.3747 -
accuracy: 0.8652 - val_loss: 0.3490 - val_accuracy: 0.8746 - lr: 0.0010
Epoch 2/15
271/271 [==============================] - 176s 650ms/step - loss: 0.2527 -
accuracy: 0.9078 - val_loss: 0.3326 - val_accuracy: 0.8883 - lr: 0.0010
Epoch 3/15
271/271 [==============================] - 175s 644ms/step - loss: 0.2223 -
accuracy: 0.9191 - val_loss: 0.2999 - val_accuracy: 0.8936 - lr: 0.0010
Epoch 4/15
271/271 [==============================] - 175s 646ms/step - loss: 0.2059 -
accuracy: 0.9246 - val_loss: 0.2673 - val_accuracy: 0.9093 - lr: 0.0010
Epoch 5/15
271/271 [==============================] - ETA: 0s - loss: 0.1834 - accuracy:
0.9340
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
271/271 [==============================] - 174s 642ms/step - loss: 0.1834 -
accuracy: 0.9340 - val_loss: 0.3542 - val_accuracy: 0.8707 - lr: 0.0010
Epoch 6/15
271/271 [==============================] - 175s 645ms/step - loss: 0.1574 -
accuracy: 0.9437 - val_loss: 0.2662 - val_accuracy: 0.9049 - lr: 5.0000e-04
Epoch 7/15
271/271 [==============================] - 175s 645ms/step - loss: 0.1428 -
accuracy: 0.9480 - val_loss: 0.2474 - val_accuracy: 0.9151 - lr: 5.0000e-04
Epoch 8/15
271/271 [==============================] - 173s 639ms/step - loss: 0.1366 -
accuracy: 0.9509 - val_loss: 0.2376 - val_accuracy: 0.9149 - lr: 5.0000e-04
Epoch 9/15
271/271 [==============================] - ETA: 0s - loss: 0.1342 - accuracy:
0.9507
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0002502000118743628.
271/271 [==============================] - 173s 638ms/step - loss: 0.1342 -
accuracy: 0.9507 - val_loss: 0.2621 - val_accuracy: 0.9072 - lr: 5.0000e-04
Epoch 10/15
271/271 [==============================] - ETA: 0s - loss: 0.1127 - accuracy:
0.9608
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0001251000059371814.
271/271 [==============================] - 173s 636ms/step - loss: 0.1127 -
accuracy: 0.9608 - val_loss: 0.2547 - val_accuracy: 0.9181 - lr: 2.5000e-04
Epoch 11/15
271/271 [==============================] - ETA: 0s - loss: 0.1004 - accuracy:
0.9661Restoring model weights from the end of the best epoch: 8.

Epoch 11: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
271/271 [==============================] - 173s 638ms/step - loss: 0.1004 -
accuracy: 0.9661 - val_loss: 0.2420 - val_accuracy: 0.9158 - lr: 1.2500e-04
Epoch 11: early stopping
```

II. Using 75-20-25 train-validation-test split

```
Epoch 1/15
254/254 [==============================] - 171s 656ms/step - loss: 0.3863 -
accuracy: 0.8615 - val_loss: 0.3277 - val_accuracy: 0.8779 - lr: 0.0010
Epoch 2/15
254/254 [==============================] - 164s 646ms/step - loss: 0.2545 -
accuracy: 0.9083 - val_loss: 0.2544 - val_accuracy: 0.9070 - lr: 0.0010
Epoch 3/15
254/254 [==============================] - ETA: 0s - loss: 0.2236 - accuracy:
0.9200
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
254/254 [==============================] - 164s 645ms/step - loss: 0.2236 -
accuracy: 0.9200 - val_loss: 0.2730 - val_accuracy: 0.9025 - lr: 0.0010
Epoch 4/15
254/254 [==============================] - 164s 644ms/step - loss: 0.1826 -
accuracy: 0.9342 - val_loss: 0.2448 - val_accuracy: 0.9092 - lr: 5.0000e-04
Epoch 5/15
254/254 [==============================] - 164s 644ms/step - loss: 0.1648 -
accuracy: 0.9404 - val_loss: 0.2443 - val_accuracy: 0.9107 - lr: 5.0000e-04
Epoch 6/15
254/254 [==============================] - 163s 640ms/step - loss: 0.1552 -
accuracy: 0.9438 - val_loss: 0.2123 - val_accuracy: 0.9245 - lr: 5.0000e-04
Epoch 7/15
254/254 [==============================] - ETA: 0s - loss: 0.1451 - accuracy:
0.9479
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
254/254 [==============================] - 165s 648ms/step - loss: 0.1451 -
accuracy: 0.9479 - val_loss: 0.2305 - val_accuracy: 0.9193 - lr: 5.0000e-04
Epoch 8/15
254/254 [==============================] - 165s 650ms/step - loss: 0.1230 -
accuracy: 0.9584 - val_loss: 0.2107 - val_accuracy: 0.9255 - lr: 2.5000e-04
Epoch 9/15
254/254 [==============================] - ETA: 0s - loss: 0.1137 - accuracy:
0.9635
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
254/254 [==============================] - 165s 648ms/step - loss: 0.1137 -
accuracy: 0.9635 - val_loss: 0.2257 - val_accuracy: 0.9255 - lr: 2.5000e-04
Epoch 10/15
254/254 [==============================] - ETA: 0s - loss: 0.1029 - accuracy:
0.9650
Epoch 10: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
254/254 [==============================] - 165s 649ms/step - loss: 0.1029 -
accuracy: 0.9650 - val_loss: 0.2151 - val_accuracy: 0.9284 - lr: 1.2500e-04
Epoch 11/15
254/254 [==============================] - ETA: 0s - loss: 0.0909 - accuracy:
0.9706Restoring model weights from the end of the best epoch: 8.

Epoch 11: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
```

```
254/254 [==============================] - 166s 650ms/step - loss: 0.0909 -
accuracy: 0.9706 - val_loss: 0.2125 - val_accuracy: 0.9265 - lr: 6.2500e-05
Epoch 11: early stopping
```

III. Using 70-20-30 train-validation-test split

```
Epoch 1/15
237/237 [==============================] - 165s 681ms/step - loss: 0.3867 -
accuracy: 0.8641 - val_loss: 0.4816 - val_accuracy: 0.8253 - lr: 0.0010
Epoch 2/15
237/237 [==============================] - 154s 650ms/step - loss: 0.2542 -
accuracy: 0.9067 - val_loss: 0.3536 - val_accuracy: 0.8705 - lr: 0.0010
Epoch 3/15
237/237 [==============================] - 154s 649ms/step - loss: 0.2236 -
accuracy: 0.9188 - val_loss: 0.2385 - val_accuracy: 0.9125 - lr: 0.0010
Epoch 4/15
237/237 [==============================] - ETA: 0s - loss: 0.2000 - accuracy:
0.9261
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
237/237 [==============================] - 154s 648ms/step - loss: 0.2000 -
accuracy: 0.9261 - val_loss: 0.3247 - val_accuracy: 0.8813 - lr: 0.0010
Epoch 5/15
237/237 [==============================] - 153s 643ms/step - loss: 0.1635 -
accuracy: 0.9414 - val_loss: 0.2170 - val_accuracy: 0.9196 - lr: 5.0000e-04
Epoch 6/15
237/237 [==============================] - ETA: 0s - loss: 0.1530 - accuracy:
0.9448
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
237/237 [==============================] - 152s 641ms/step - loss: 0.1530 -
accuracy: 0.9448 - val_loss: 0.2399 - val_accuracy: 0.9141 - lr: 5.0000e-04
Epoch 7/15
237/237 [==============================] - ETA: 0s - loss: 0.1322 - accuracy:
0.9538
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
237/237 [==============================] - 153s 643ms/step - loss: 0.1322 -
accuracy: 0.9538 - val_loss: 0.2575 - val_accuracy: 0.9040 - lr: 2.5000e-04
Epoch 8/15
237/237 [==============================] - ETA: 0s - loss: 0.1181 - accuracy:
0.9592Restoring model weights from the end of the best epoch: 5.

Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
237/237 [==============================] - 152s 641ms/step - loss: 0.1181 -
accuracy: 0.9592 - val_loss: 0.2190 - val_accuracy: 0.9157 - lr: 1.2500e-04
Epoch 8: early stopping
```

3.2.3.3.2 Architecture 2

VGG19 pre-trained model is used here. VGG19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 fully connected layer, 5 maxpool layers and 1 softmax layer). Like the previous architecture, here also fine-tuning is applied to change some last few layers for our new problem. The proposed model is shown in Figure 7.

For this architecture we follow the below configurations.

- Set patience = 3 for early stopping
- Optimizer = Adam
- Learning Rate = 0.001

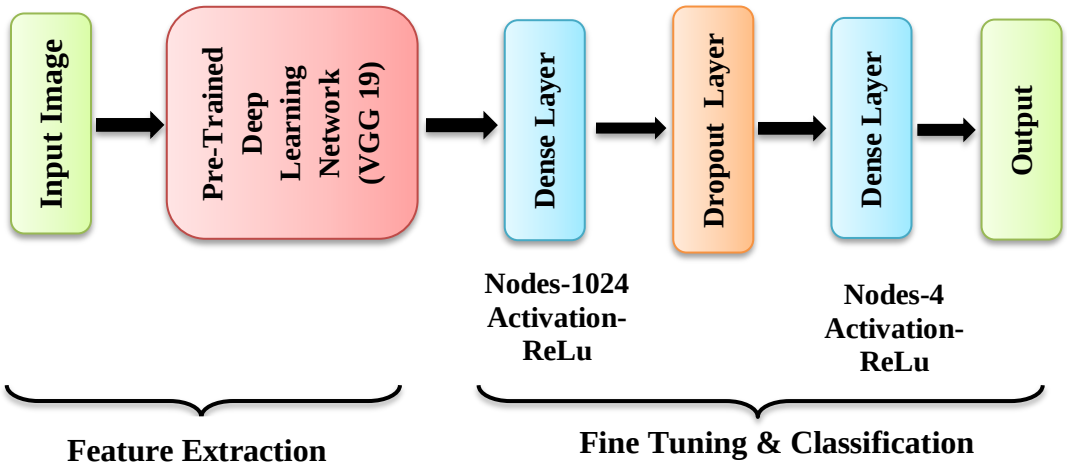


Figure 7 : Architecture 2

The snapshot of training this model using different train-test split is given below.

I. Using 80-20-20 train-validation-test split

```
Epoch 1/15
271/271 [==============================] - 139s 511ms/step - loss: 0.3011 -
accuracy: 0.9045 - val_loss: 0.2771 - val_accuracy: 0.9133 - lr: 0.0010
Epoch 2/15
271/271 [==============================] - ETA: 0s - loss: 0.2941 - accuracy:
0.9063
Epoch 2: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
271/271 [==============================] - 141s 521ms/step - loss: 0.2941 -
accuracy: 0.9063 - val_loss: 0.2980 - val_accuracy: 0.9107 - lr: 0.0010
Epoch 3/15
```

```
271/271 [==============================] - 139s 512ms/step - loss: 0.1952 -
accuracy: 0.9352 - val_loss: 0.2309 - val_accuracy: 0.9292 - lr: 5.0000e-04
Epoch 4/15
271/271 [==============================] - 138s 510ms/step - loss: 0.1576 -
accuracy: 0.9452 - val_loss: 0.2272 - val_accuracy: 0.9350 - lr: 5.0000e-04
Epoch 5/15
271/271 [==============================] - ETA: 0s - loss: 0.1291 - accuracy:
0.9550
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
271/271 [==============================] - 138s 510ms/step - loss: 0.1291 -
accuracy: 0.9550 - val_loss: 0.2491 - val_accuracy: 0.9278 - lr: 5.0000e-04
Epoch 6/15
271/271 [==============================] - ETA: 0s - loss: 0.1016 - accuracy:
0.9652
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
271/271 [==============================] - 138s 510ms/step - loss: 0.1016 -
accuracy: 0.9652 - val_loss: 0.2627 - val_accuracy: 0.9329 - lr: 2.5000e-04
Epoch 7/15
271/271 [==============================] - ETA: 0s - loss: 0.0830 - accuracy:
0.9692Restoring model weights from the end of the best epoch: 4.

Epoch 7: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
271/271 [==============================] - 138s 510ms/step - loss: 0.0830 -
accuracy: 0.9692 - val_loss: 0.2522 - val_accuracy: 0.9338 - lr: 1.2500e-04
Epoch 7: early stopping
```

II. Using 75-20-25 train-validation-test split

```
Epoch 1/15
254/254 [==============================] - 137s 536ms/step - loss: 0.3277 -
accuracy: 0.8859 - val_loss: 0.2752 - val_accuracy: 0.9050 - lr: 0.0010
Epoch 2/15
254/254 [==============================] - 130s 512ms/step - loss: 0.2862 -
accuracy: 0.8996 - val_loss: 0.2745 - val_accuracy: 0.9114 - lr: 0.0010
Epoch 3/15
254/254 [==============================] - 131s 513ms/step - loss: 0.2617 -
accuracy: 0.9086 - val_loss: 0.2659 - val_accuracy: 0.9065 - lr: 0.0010
Epoch 4/15
254/254 [==============================] - 130s 510ms/step - loss: 0.2532 -
accuracy: 0.9124 - val_loss: 0.2585 - val_accuracy: 0.9149 - lr: 0.0010
Epoch 5/15
254/254 [==============================] - 130s 511ms/step - loss: 0.2459 -
accuracy: 0.9107 - val_loss: 0.2497 - val_accuracy: 0.9191 - lr: 0.0010
Epoch 6/15
254/254 [==============================] - 130s 512ms/step - loss: 0.2234 -
accuracy: 0.9191 - val_loss: 0.2462 - val_accuracy: 0.9215 - lr: 0.0010
Epoch 7/15
```

```
254/254 [==============================] - ETA: 0s - loss: 0.2015 - accuracy:
0.9297
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
254/254 [==============================] - 130s 511ms/step - loss: 0.2015 -
accuracy: 0.9297 - val_loss: 0.2655 - val_accuracy: 0.9238 - lr: 0.0010
Epoch 8/15
254/254 [==============================] - ETA: 0s - loss: 0.1571 - accuracy:
0.9425
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
254/254 [==============================] - 130s 511ms/step - loss: 0.1571 -
accuracy: 0.9425 - val_loss: 0.2732 - val_accuracy: 0.9270 - lr: 5.0000e-04
Epoch 9/15
254/254 [==============================] - ETA: 0s - loss: 0.1115 - accuracy:
0.9575Restoring model weights from the end of the best epoch: 6.

Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
254/254 [==============================] - 130s 511ms/step - loss: 0.1115 -
accuracy: 0.9575 - val_loss: 0.2579 - val_accuracy: 0.9307 - lr: 2.5000e-04
Epoch 9: early stopping
```

III. Using 70-20-30 train-validation-test split

```
Epoch 1/15
237/237 [==============================] - 129s 540ms/step - loss: 6.1839 -
accuracy: 0.8348 - val_loss: 0.3123 - val_accuracy: 0.8964 - lr: 0.0010
Epoch 2/15
237/237 [==============================] - 122s 514ms/step - loss: 0.3121 -
accuracy: 0.8985 - val_loss: 0.2678 - val_accuracy: 0.9138 - lr: 0.0010
Epoch 3/15
237/237 [==============================] - 122s 515ms/step - loss: 0.2598 -
accuracy: 0.9112 - val_loss: 0.2425 - val_accuracy: 0.9186 - lr: 0.0010
Epoch 4/15
237/237 [==============================] - ETA: 0s - loss: 0.2237 - accuracy:
0.9261
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
237/237 [==============================] - 121s 511ms/step - loss: 0.2237 -
accuracy: 0.9261 - val_loss: 0.2627 - val_accuracy: 0.9172 - lr: 0.0010
Epoch 5/15
237/237 [==============================] - 124s 523ms/step - loss: 0.1612 -
accuracy: 0.9429 - val_loss: 0.2244 - val_accuracy: 0.9254 - lr: 5.0000e-04
Epoch 6/15
237/237 [==============================] - ETA: 0s - loss: 0.1206 - accuracy:
0.9566
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
237/237 [==============================] - 122s 514ms/step - loss: 0.1206 -
accuracy: 0.9566 - val_loss: 0.2749 - val_accuracy: 0.9212 - lr: 5.0000e-04
Epoch 7/15
237/237 [==============================] - ETA: 0s - loss: 0.0929 - accuracy:
0.9663
```

```
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
237/237 [==============================] - 121s 509ms/step - loss: 0.0929 -
accuracy: 0.9663 - val_loss: 0.2471 - val_accuracy: 0.9278 - lr: 2.5000e-04
Epoch 8/15
237/237 [==============================] - ETA: 0s - loss: 0.0708 - accuracy:
0.9745Restoring model weights from the end of the best epoch: 5.

Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
237/237 [==============================] - 121s 510ms/step - loss: 0.0708 -
accuracy: 0.9745 - val_loss: 0.2650 - val_accuracy: 0.9291 - lr: 1.2500e-04
Epoch 8: early stopping
```

3.2.3.3.3 Architecture 3

Like other previous architectures, here we also used a fine-tuned pre-trained model that is EfficientnetB0. EfficientnetB0 is a variant of EfficientNet and it is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The architecture of this model is shown in Figure 8.

For this architecture we follow the below configurations.

- Set patience = 5 for early stopping
- Optimizer = RMSprop
- Learning Rate = 0.001

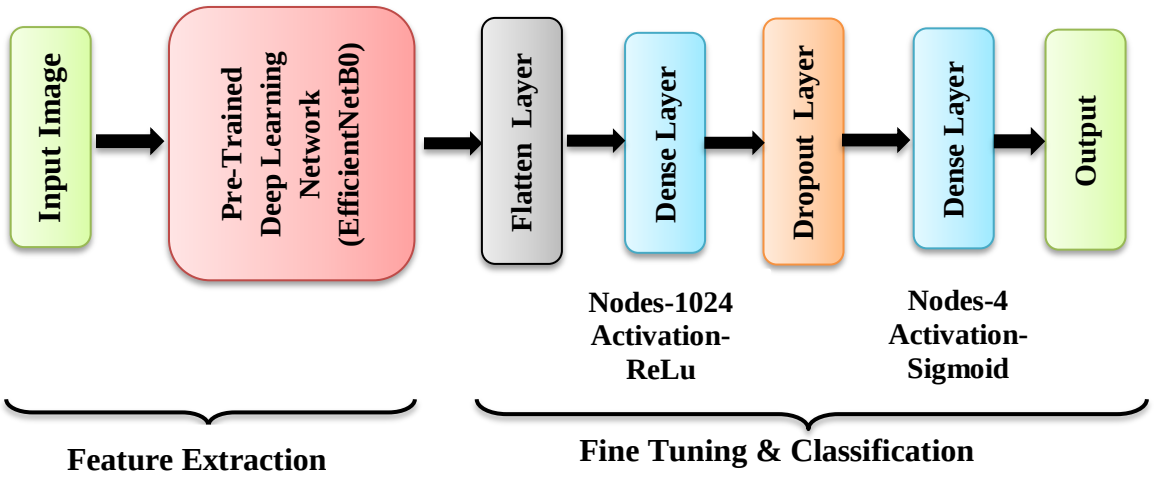


Figure 8 : Architecture 3

I. Using 80-20-20 train-validation-test split

```
Epoch 1/15
271/271 [==============================] - 6863s 25s/step - loss: 3.7134 -
accuracy: 0.7849 - val_loss: 0.3335 - val_accuracy: 0.8952 - lr: 0.0010
Epoch 2/15
271/271 [==============================] - 123s 453ms/step - loss: 0.5599 -
accuracy: 0.8515 - val_loss: 0.2963 - val_accuracy: 0.9024 - lr: 0.0010
Epoch 3/15
271/271 [==============================] - 122s 450ms/step - loss: 0.4832 -
accuracy: 0.8737 - val_loss: 0.2695 - val_accuracy: 0.9216 - lr: 0.0010
Epoch 4/15
271/271 [==============================] - ETA: 0s - loss: 0.4357 - accuracy:
0.8870
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
271/271 [==============================] - 122s 450ms/step - loss: 0.4357 -
accuracy: 0.8870 - val_loss: 0.2744 - val_accuracy: 0.9181 - lr: 0.0010
Epoch 5/15
271/271 [==============================] - 124s 459ms/step - loss: 0.2761 -
accuracy: 0.9202 - val_loss: 0.2368 - val_accuracy: 0.9283 - lr: 5.0000e-04
Epoch 6/15
271/271 [==============================] - ETA: 0s - loss: 0.2503 - accuracy:
0.9288
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
271/271 [==============================] - 122s 451ms/step - loss: 0.2503 -
accuracy: 0.9288 - val_loss: 0.2540 - val_accuracy: 0.9297 - lr: 5.0000e-04
Epoch 7/15
271/271 [==============================] - ETA: 0s - loss: 0.1996 - accuracy:
0.9396
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
271/271 [==============================] - 121s 447ms/step - loss: 0.1996 -
accuracy: 0.9396 - val_loss: 0.2402 - val_accuracy: 0.9299 - lr: 2.5000e-04
Epoch 8/15
271/271 [==============================] - ETA: 0s - loss: 0.1590 - accuracy:
0.9493
Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
271/271 [==============================] - 121s 446ms/step - loss: 0.1590 -
accuracy: 0.9493 - val_loss: 0.2439 - val_accuracy: 0.9343 - lr: 1.2500e-04
Epoch 9/15
271/271 [==============================] - ETA: 0s - loss: 0.1515 - accuracy:
0.9534
Epoch 9: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
271/271 [==============================] - 120s 444ms/step - loss: 0.1515 -
accuracy: 0.9534 - val_loss: 0.2425 - val_accuracy: 0.9371 - lr: 6.2500e-05
Epoch 10/15
271/271 [==============================] - ETA: 0s - loss: 0.1370 - accuracy:
0.9569Restoring model weights from the end of the best epoch: 5.

Epoch 10: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
```

```
271/271 [==============================] - 121s 446ms/step - loss: 0.1370 -
accuracy: 0.9569 - val_loss: 0.2450 - val_accuracy: 0.9387 - lr: 3.1250e-05
Epoch 10: early stopping
```

II. Using 75-20-25 train validation test split

```
Epoch 1/15
254/254 [==============================] - 128s 481ms/step - loss: 3.8178 -
accuracy: 0.7812 - val_loss: 0.4187 - val_accuracy: 0.8678 - lr: 0.0010
Epoch 2/15
254/254 [==============================] - 119s 469ms/step - loss: 0.5587 -
accuracy: 0.8486 - val_loss: 0.3462 - val_accuracy: 0.8961 - lr: 0.0010
Epoch 3/15
254/254 [==============================] - ETA: 0s - loss: 0.4558 - accuracy:
0.8769
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
254/254 [==============================] - 118s 465ms/step - loss: 0.4558 -
accuracy: 0.8769 - val_loss: 0.3725 - val_accuracy: 0.8961 - lr: 0.0010
Epoch 4/15
254/254 [==============================] - 118s 466ms/step - loss: 0.2850 -
accuracy: 0.9136 - val_loss: 0.2972 - val_accuracy: 0.9245 - lr: 5.0000e-04
Epoch 5/15
254/254 [==============================] - ETA: 0s - loss: 0.2619 - accuracy:
0.9221
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
254/254 [==============================] - 116s 458ms/step - loss: 0.2619 -
accuracy: 0.9221 - val_loss: 0.3244 - val_accuracy: 0.9208 - lr: 5.0000e-04
Epoch 6/15
254/254 [==============================] - 117s 459ms/step - loss: 0.1982 -
accuracy: 0.9373 - val_loss: 0.2675 - val_accuracy: 0.9302 - lr: 2.5000e-04
Epoch 7/15
254/254 [==============================] - 118s 463ms/step - loss: 0.1796 -
accuracy: 0.9414 - val_loss: 0.2456 - val_accuracy: 0.9329 - lr: 2.5000e-04
Epoch 8/15
254/254 [==============================] - ETA: 0s - loss: 0.1817 - accuracy:
0.9425
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
254/254 [==============================] - 118s 464ms/step - loss: 0.1817 -
accuracy: 0.9425 - val_loss: 0.2573 - val_accuracy: 0.9346 - lr: 2.5000e-04
Epoch 9/15
254/254 [==============================] - ETA: 0s - loss: 0.1410 - accuracy:
0.9516
Epoch 9: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
254/254 [==============================] - 116s 455ms/step - loss: 0.1410 -
accuracy: 0.9516 - val_loss: 0.2516 - val_accuracy: 0.9336 - lr: 1.2500e-04
Epoch 10/15
254/254 [==============================] - ETA: 0s - loss: 0.1316 - accuracy:
0.9561
Epoch 10: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
```

```
254/254 [==============================] - 115s 451ms/step - loss: 0.1316 -
accuracy: 0.9561 - val_loss: 0.2692 - val_accuracy: 0.9358 - lr: 6.2500e-05
Epoch 11/15
254/254 [==============================] - ETA: 0s - loss: 0.1231 - accuracy:
0.9602
Epoch 11: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
254/254 [==============================] - 115s 451ms/step - loss: 0.1231 -
accuracy: 0.9602 - val_loss: 0.2689 - val_accuracy: 0.9356 - lr: 3.1250e-05
Epoch 12/15
254/254 [==============================] - ETA: 0s - loss: 0.1203 - accuracy:
0.9605Restoring model weights from the end of the best epoch: 7.

Epoch 12: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
254/254 [==============================] - 115s 453ms/step - loss: 0.1203 -
accuracy: 0.9605 - val_loss: 0.2708 - val_accuracy: 0.9351 - lr: 1.5625e-05
Epoch 12: early stopping
```

III. Using 70-20-30 train-validation-test split

```
Epoch 1/15
237/237 [==============================] - 117s 469ms/step - loss: 4.4725 -
accuracy: 0.7704 - val_loss: 0.4829 - val_accuracy: 0.8377 - lr: 0.0010
Epoch 2/15
237/237 [==============================] - 108s 455ms/step - loss: 0.6068 -
accuracy: 0.8411 - val_loss: 0.3752 - val_accuracy: 0.8877 - lr: 0.0010
Epoch 3/15
237/237 [==============================] - 107s 453ms/step - loss: 0.4923 -
accuracy: 0.8696 - val_loss: 0.2890 - val_accuracy: 0.9051 - lr: 0.0010
Epoch 4/15
237/237 [==============================] - ETA: 0s - loss: 0.4295 - accuracy:
0.8859
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
237/237 [==============================] - 108s 456ms/step - loss: 0.4295 -
accuracy: 0.8859 - val_loss: 0.2963 - val_accuracy: 0.9141 - lr: 0.0010
Epoch 5/15
237/237 [==============================] - 108s 456ms/step - loss: 0.2708 -
accuracy: 0.9196 - val_loss: 0.2389 - val_accuracy: 0.9276 - lr: 5.0000e-04
Epoch 6/15
237/237 [==============================] - ETA: 0s - loss: 0.2484 - accuracy:
0.9249
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
237/237 [==============================] - 107s 452ms/step - loss: 0.2484 -
accuracy: 0.9249 - val_loss: 0.2582 - val_accuracy: 0.9241 - lr: 5.0000e-04
Epoch 7/15
237/237 [==============================] - ETA: 0s - loss: 0.1973 - accuracy:
0.9391
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
237/237 [==============================] - 107s 452ms/step - loss: 0.1973 -
accuracy: 0.9391 - val_loss: 0.2616 - val_accuracy: 0.9329 - lr: 2.5000e-04
```

```
Epoch 8/15
237/237 [==============================] - ETA: 0s - loss: 0.1593 - accuracy:
0.9492
Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
237/237 [==============================] - 106s 449ms/step - loss: 0.1593 -
accuracy: 0.9492 - val_loss: 0.2440 - val_accuracy: 0.9381 - lr: 1.2500e-04
Epoch 9/15
237/237 [==============================] - ETA: 0s - loss: 0.1393 - accuracy:
0.9545
Epoch 9: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
237/237 [==============================] - 107s 451ms/step - loss: 0.1393 -
accuracy: 0.9545 - val_loss: 0.2470 - val_accuracy: 0.9368 - lr: 6.2500e-05
Epoch 10/15
237/237 [==============================] - ETA: 0s - loss: 0.1290 - accuracy:
0.9581Restoring model weights from the end of the best epoch: 5.

Epoch 10: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
237/237 [==============================] - 107s 452ms/step - loss: 0.1290 -
accuracy: 0.9581 - val_loss: 0.2497 - val_accuracy: 0.9366 - lr: 3.1250e-05
Epoch 10: early stopping
```

3.2.3.3.4 Architecture 4

Here we used the Xception model which is 71 layers deep. This model is also trained on millions of images from the imagenet database. We applied fine-tuning like other models to change some last few layers. Figure 9 describes the proposed model architecture.

For this architecture, we follow the below configurations.

- Set patience = 5 for early stopping
- Optimizer = Adam
- Learning Rate = 0.001

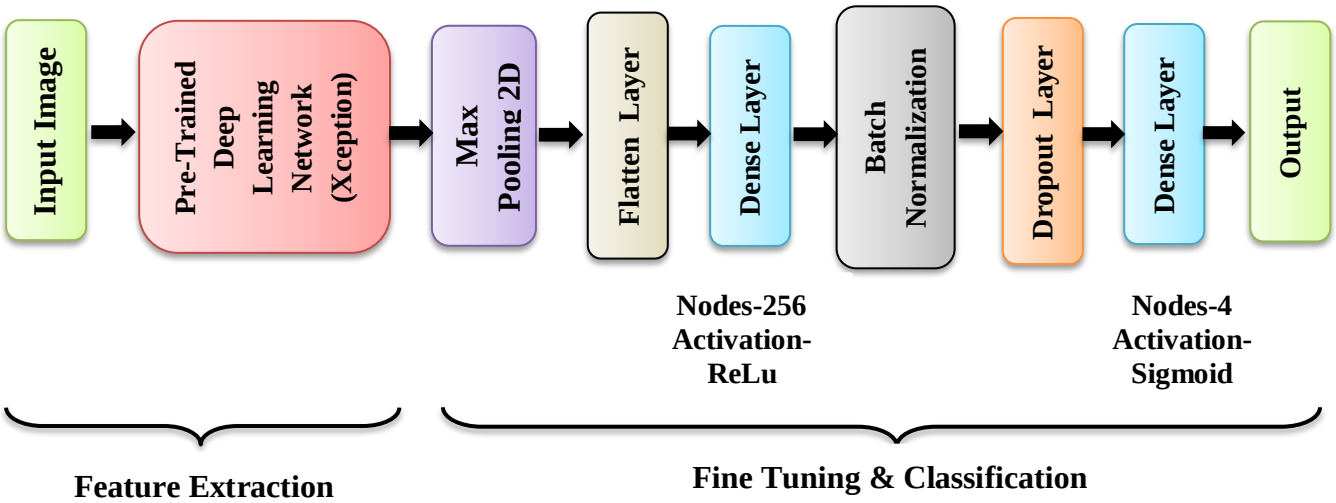


Figure 9 : Architecture 4

I. Using 80-20-20 train-validation-test split

```
Epoch 1/15
271/271 [==============================] - 2300s 9s/step - loss: 0.6787 -
accuracy: 0.7496 - val_loss: 0.6410 - val_accuracy: 0.7437 - lr: 0.0010
Epoch 2/15
271/271 [==============================] - 130s 479ms/step - loss: 0.5511 -
accuracy: 0.7907 - val_loss: 0.5724 - val_accuracy: 0.7752 - lr: 0.0010
Epoch 3/15
271/271 [==============================] - ETA: 0s - loss: 0.5289 - accuracy:
0.8010
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
271/271 [==============================] - 130s 478ms/step - loss: 0.5289 -
accuracy: 0.8010 - val_loss: 0.6684 - val_accuracy: 0.7650 - lr: 0.0010
Epoch 4/15
271/271 [==============================] - ETA: 0s - loss: 0.4900 - accuracy:
0.8154
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
271/271 [==============================] - 129s 475ms/step - loss: 0.4900 -
accuracy: 0.8154 - val_loss: 0.6040 - val_accuracy: 0.7654 - lr: 5.0000e-04
Epoch 5/15
271/271 [==============================] - 129s 476ms/step - loss: 0.4638 -
accuracy: 0.8257 - val_loss: 0.5231 - val_accuracy: 0.8018 - lr: 2.5000e-04
Epoch 6/15
271/271 [==============================] - ETA: 0s - loss: 0.4585 - accuracy:
0.8297
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
271/271 [==============================] - 129s 474ms/step - loss: 0.4585 -
accuracy: 0.8297 - val_loss: 0.5783 - val_accuracy: 0.7849 - lr: 2.5000e-04
Epoch 7/15
271/271 [==============================] - 129s 474ms/step - loss: 0.4431 -
accuracy: 0.8335 - val_loss: 0.4747 - val_accuracy: 0.8166 - lr: 1.2500e-04
Epoch 8/15
271/271 [==============================] - ETA: 0s - loss: 0.4325 - accuracy:
0.8384
Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
271/271 [==============================] - 129s 477ms/step - loss: 0.4325 -
accuracy: 0.8384 - val_loss: 0.4792 - val_accuracy: 0.8217 - lr: 1.2500e-04
Epoch 9/15
271/271 [==============================] - 129s 477ms/step - loss: 0.4261 -
accuracy: 0.8422 - val_loss: 0.4675 - val_accuracy: 0.8258 - lr: 6.2500e-05
Epoch 10/15
271/271 [==============================] - 129s 476ms/step - loss: 0.4204 -
accuracy: 0.8458 - val_loss: 0.4668 - val_accuracy: 0.8217 - lr: 6.2500e-05
Epoch 11/15
271/271 [==============================] - ETA: 0s - loss: 0.4203 - accuracy:
0.8445
Epoch 11: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
```

```
271/271 [==============================] - 129s 476ms/step - loss: 0.4203 -
accuracy: 0.8445 - val_loss: 0.4708 - val_accuracy: 0.8196 - lr: 6.2500e-05
Epoch 12/15
271/271 [==============================] - 129s 477ms/step - loss: 0.4135 -
accuracy: 0.8458 - val_loss: 0.4605 - val_accuracy: 0.8281 - lr: 3.1250e-05
Epoch 13/15
271/271 [==============================] - ETA: 0s - loss: 0.4128 - accuracy:
0.8443
Epoch 13: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
271/271 [==============================] - 128s 473ms/step - loss: 0.4128 -
accuracy: 0.8443 - val_loss: 0.4658 - val_accuracy: 0.8260 - lr: 3.1250e-05
Epoch 14/15
271/271 [==============================] - ETA: 0s - loss: 0.4120 - accuracy:
0.8476
Epoch 14: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
271/271 [==============================] - 129s 475ms/step - loss: 0.4120 -
accuracy: 0.8476 - val_loss: 0.4615 - val_accuracy: 0.8274 - lr: 1.5625e-05
Epoch 15/15
271/271 [==============================] - ETA: 0s - loss: 0.4089 - accuracy:
0.8486
Epoch 15: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
271/271 [==============================] - 128s 473ms/step - loss: 0.4089 -
accuracy: 0.8486 - val_loss: 0.4604 - val_accuracy: 0.8277 - lr: 7.8125e-06
```

II. Using 75-20-25 train-validation-test split

```
Epoch 1/15
254/254 [==============================] - 4138s 16s/step - loss: 0.6965 -
accuracy: 0.7359 - val_loss: 0.5465 - val_accuracy: 0.7876 - lr: 0.0010
Epoch 2/15
254/254 [==============================] - ETA: 0s - loss: 0.5520 - accuracy:
0.7873
Epoch 2: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
254/254 [==============================] - 121s 478ms/step - loss: 0.5520 -
accuracy: 0.7873 - val_loss: 0.7139 - val_accuracy: 0.7237 - lr: 0.0010
Epoch 3/15
254/254 [==============================] - ETA: 0s - loss: 0.4938 - accuracy:
0.8119
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
254/254 [==============================] - 120s 470ms/step - loss: 0.4938 -
accuracy: 0.8119 - val_loss: 0.7471 - val_accuracy: 0.7084 - lr: 5.0000e-04
Epoch 4/15
254/254 [==============================] - ETA: 0s - loss: 0.4623 - accuracy:
0.8234
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
254/254 [==============================] - 119s 469ms/step - loss: 0.4623 -
accuracy: 0.8234 - val_loss: 0.6581 - val_accuracy: 0.7597 - lr: 2.5000e-04
Epoch 5/15
```

```
254/254 [==============================] - 120s 470ms/step - loss: 0.4427 -
accuracy: 0.8330 - val_loss: 0.4534 - val_accuracy: 0.8295 - lr: 1.2500e-04
Epoch 6/15
254/254 [==============================] - ETA: 0s - loss: 0.4311 - accuracy:
0.8392
Epoch 6: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
254/254 [==============================] - 119s 470ms/step - loss: 0.4311 -
accuracy: 0.8392 - val_loss: 0.4740 - val_accuracy: 0.8179 - lr: 1.2500e-04
Epoch 7/15
254/254 [==============================] - 119s 469ms/step - loss: 0.4244 -
accuracy: 0.8407 - val_loss: 0.4473 - val_accuracy: 0.8384 - lr: 6.2500e-05
Epoch 8/15
254/254 [==============================] - ETA: 0s - loss: 0.4197 - accuracy:
0.8418
Epoch 8: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
254/254 [==============================] - 121s 474ms/step - loss: 0.4197 -
accuracy: 0.8418 - val_loss: 0.4536 - val_accuracy: 0.8312 - lr: 6.2500e-05
Epoch 9/15
254/254 [==============================] - ETA: 0s - loss: 0.4122 - accuracy:
0.8454
Epoch 9: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
254/254 [==============================] - 120s 471ms/step - loss: 0.4122 -
accuracy: 0.8454 - val_loss: 0.4563 - val_accuracy: 0.8302 - lr: 3.1250e-05
Epoch 10/15
254/254 [==============================] - 120s 473ms/step - loss: 0.4097 -
accuracy: 0.8464 - val_loss: 0.4419 - val_accuracy: 0.8344 - lr: 1.5625e-05
Epoch 11/15
254/254 [==============================] - 120s 471ms/step - loss: 0.4070 -
accuracy: 0.8493 - val_loss: 0.4386 - val_accuracy: 0.8389 - lr: 1.5625e-05
Epoch 12/15
254/254 [==============================] - 120s 473ms/step - loss: 0.4081 -
accuracy: 0.8475 - val_loss: 0.4333 - val_accuracy: 0.8384 - lr: 1.5625e-05
Epoch 13/15
254/254 [==============================] - ETA: 0s - loss: 0.4075 - accuracy:
0.8470
Epoch 13: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
254/254 [==============================] - 119s 467ms/step - loss: 0.4075 -
accuracy: 0.8470 - val_loss: 0.4387 - val_accuracy: 0.8364 - lr: 1.5625e-05
Epoch 14/15
254/254 [==============================] - ETA: 0s - loss: 0.4054 - accuracy:
0.8463
Epoch 14: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
254/254 [==============================] - 118s 466ms/step - loss: 0.4054 -
accuracy: 0.8463 - val_loss: 0.4335 - val_accuracy: 0.8391 - lr: 7.8125e-06
Epoch 15/15
254/254 [==============================] - 119s 469ms/step - loss: 0.4065 -
accuracy: 0.8459 - val_loss: 0.4330 - val_accuracy: 0.8374 - lr: 3.9063e-06
```

III. Using 70-20-30 train-validation-test split

```
Epoch 1/15
237/237 [==============================] - 119s 493ms/step - loss: 0.6942 -
accuracy: 0.7400 - val_loss: 0.8066 - val_accuracy: 0.6838 - lr: 0.0010
Epoch 2/15
237/237 [==============================] - ETA: 0s - loss: 0.5603 - accuracy:
0.7877
Epoch 2: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
237/237 [==============================] - 114s 480ms/step - loss: 0.5603 -
accuracy: 0.7877 - val_loss: 0.9477 - val_accuracy: 0.7359 - lr: 0.0010
Epoch 3/15
237/237 [==============================] - 112s 474ms/step - loss: 0.5062 -
accuracy: 0.8103 - val_loss: 0.5925 - val_accuracy: 0.7602 - lr: 5.0000e-04
Epoch 4/15
237/237 [==============================] - 112s 473ms/step - loss: 0.4887 -
accuracy: 0.8158 - val_loss: 0.5270 - val_accuracy: 0.7983 - lr: 5.0000e-04
Epoch 5/15
237/237 [==============================] - ETA: 0s - loss: 0.4775 - accuracy:
0.8222
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
237/237 [==============================] - 113s 478ms/step - loss: 0.4775 -
accuracy: 0.8222 - val_loss: 0.6146 - val_accuracy: 0.7700 - lr: 5.0000e-04
Epoch 6/15
237/237 [==============================] - 113s 475ms/step - loss: 0.4580 -
accuracy: 0.8277 - val_loss: 0.4483 - val_accuracy: 0.8332 - lr: 2.5000e-04
Epoch 7/15
237/237 [==============================] - ETA: 0s - loss: 0.4484 - accuracy:
0.8326
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
237/237 [==============================] - 113s 478ms/step - loss: 0.4484 -
accuracy: 0.8326 - val_loss: 0.5058 - val_accuracy: 0.8113 - lr: 2.5000e-04
Epoch 8/15
237/237 [==============================] - ETA: 0s - loss: 0.4295 - accuracy:
0.8401
Epoch 8: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
237/237 [==============================] - 113s 476ms/step - loss: 0.4295 -
accuracy: 0.8401 - val_loss: 0.4728 - val_accuracy: 0.8226 - lr: 1.2500e-04
Epoch 9/15
237/237 [==============================] - 113s 478ms/step - loss: 0.4229 -
accuracy: 0.8428 - val_loss: 0.4367 - val_accuracy: 0.8395 - lr: 6.2500e-05
Epoch 10/15
237/237 [==============================] - ETA: 0s - loss: 0.4180 - accuracy:
0.8436
Epoch 10: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
237/237 [==============================] - 112s 472ms/step - loss: 0.4180 -
accuracy: 0.8436 - val_loss: 0.4427 - val_accuracy: 0.8316 - lr: 6.2500e-05
Epoch 11/15
```

```
237/237 [==============================] - ETA: 0s - loss: 0.4104 - accuracy:
0.8479
Epoch 11: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
237/237 [==============================] - 112s 474ms/step - loss: 0.4104 -
accuracy: 0.8479 - val_loss: 0.4377 - val_accuracy: 0.8419 - lr: 3.1250e-05
Epoch 12/15
237/237 [==============================] - 114s 479ms/step - loss: 0.4085 -
accuracy: 0.8461 - val_loss: 0.4362 - val_accuracy: 0.8406 - lr: 1.5625e-05
Epoch 13/15
237/237 [==============================] - 112s 474ms/step - loss: 0.4075 -
accuracy: 0.8501 - val_loss: 0.4349 - val_accuracy: 0.8382 - lr: 1.5625e-05
Epoch 14/15
237/237 [==============================] - 112s 473ms/step - loss: 0.4055 -
accuracy: 0.8489 - val_loss: 0.4338 - val_accuracy: 0.8430 - lr: 1.5625e-05
Epoch 15/15
237/237 [==============================] - ETA: 0s - loss: 0.4048 - accuracy:
0.8481
Epoch 15: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
237/237 [==============================] - 113s 475ms/step - loss: 0.4048 -
accuracy: 0.8481 - val_loss: 0.4373 - val_accuracy: 0.8401 - lr: 1.5625e-05
```

Chapter 4

Experiments and Results

This section discusses the classification performance of the different models on a multiclass chest X-ray image dataset in detail.

4.1 Comparison of the Four Architectures in section 3.2.3

Table 4 : Comparative classification performance of all tested deep learning models on a GPU using 80-20-20 train validation test split

Classifier	Patient Status	Precision	Recall	F1-Score
ResNet50	COVID-19 Lung Opacity Normal Pneumonia	0.87 0.87 0.92 0.98	0.93 0.87 0.91 0.96	0.90 0.87 0.91 0.97
VGG19	COVID-19 Lung Opacity Normal Pneumonia	0.75 0.52 0.69 0.98	0.92 0.90 0.51 0.62	0.83 0.66 0.59 0.76
EfficientNetB0	COVID-19 Lung Opacity Normal Pneumonia	0.94 0.82 0.91 0.99	0.93 0.86 0.89 0.97	0.94 0.84 0.90 0.98
Xception	COVID-19 Lung Opacity Normal Pneumonia	0.67 0.80 0.85 0.93	0.65 0.76 0.87 0.94	0.66 0.78 0.86 0.94

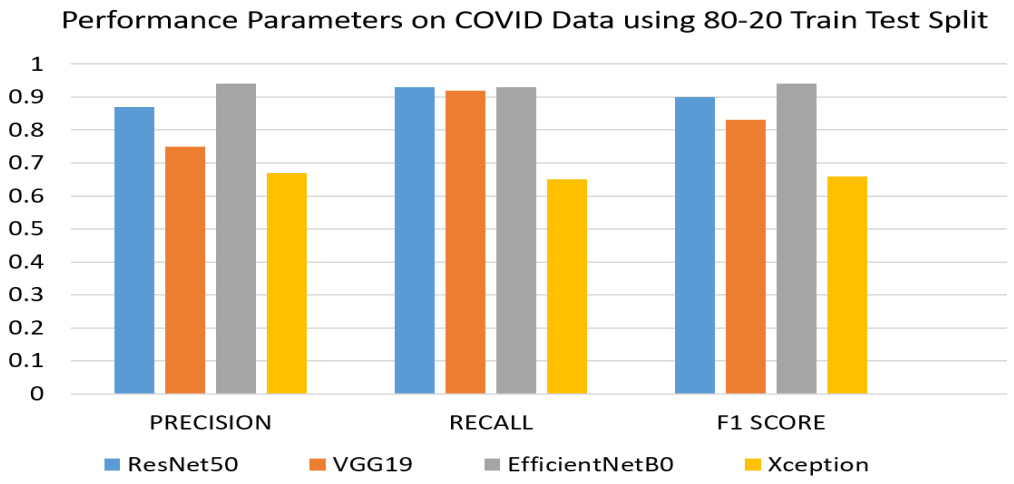


Figure 10 : Plot for COVID Category Performance of all models using 80-20 train test split

Table 5 : Comparative classification performance of all tested deep learning models on a GPU using 75-20-25 train validation test split

Classifier	Patient Status	Precision	Recall	F1-Score
ResNet50	COVID-19	0.90	0.93	0.91
	Lung Opacity	0.90	0.83	0.86
	Normal	0.91	0.94	0.93
	Pneumonia	0.98	0.98	0.98
VGG19	COVID-19	0.94	0.92	0.93
	Lung Opacity	0.88	0.83	0.85
	Normal	0.82	0.93	0.87
	Pneumonia	0.98	0.76	0.86
EfficientNetB0	COVID-19	0.92	0.95	0.93
	Lung Opacity	0.78	0.87	0.82
	Normal	0.92	0.86	0.89
	Pneumonia	0.97	0.98	0.98
Xception	COVID-19	0.73	0.61	0.67
	Lung Opacity	0.80	0.77	0.78
	Normal	0.82	0.87	0.85
	Pneumonia	0.92	0.93	0.93

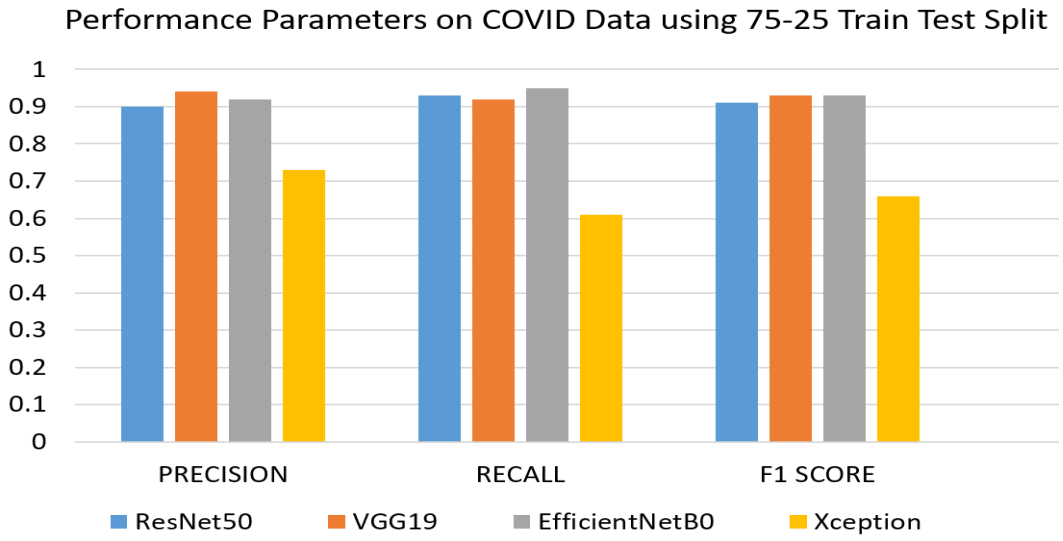


Figure 11 : Plot for COVID Category Performance of all models using 75-25 train test split

Table 6 : Comparative classification performance of all tested deep learning models on a GPU using 70-20-30 train validation test split

Classifier	Patient Status	Precision	Recall	F1-Score
ResNet50	COVID-19	0.95	0.96	0.96
	Lung Opacity	0.94	0.88	0.91
	Normal	0.94	0.97	0.95
	Pneumonia	0.99	0.99	0.99
VGG19	COVID-19	0.97	0.97	0.97
	Lung Opacity	0.87	0.92	0.90
	Normal	0.85	0.93	0.89
	Pneumonia	0.99	0.75	0.85
EfficientNetB0	COVID-19	0.94	0.95	0.94
	Lung Opacity	0.91	0.85	0.88
	Normal	0.91	0.95	0.93
	Pneumonia	0.98	0.97	0.98
Xception	COVID-19	0.74	0.59	0.65
	Lung Opacity	0.81	0.77	0.79
	Normal	0.81	0.89	0.85
	Pneumonia	0.93	0.92	0.93

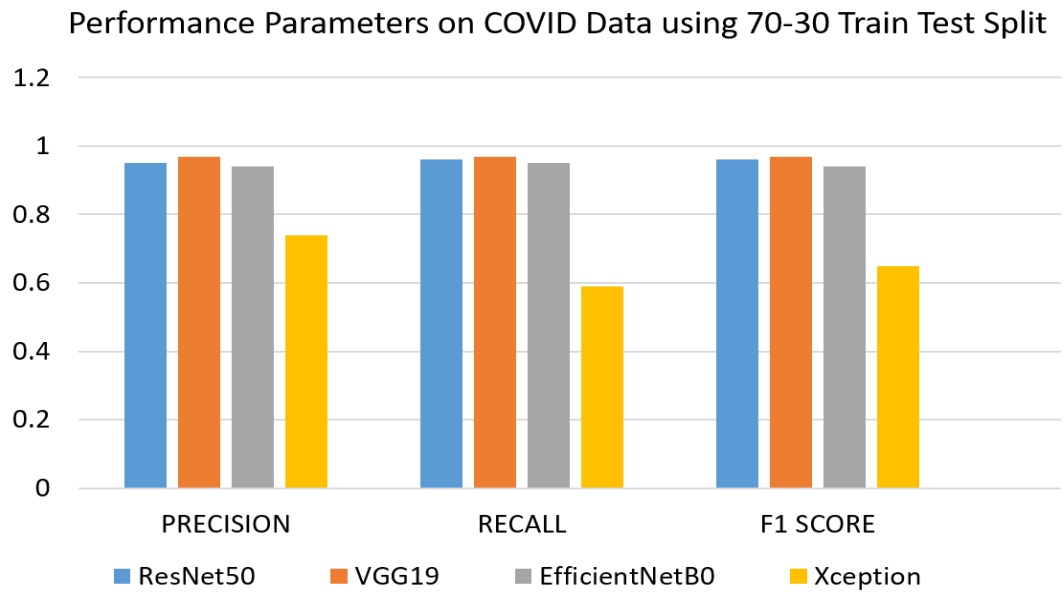


Figure 12 : Plot for COVID Category Performance of all models using 70-30 train test split

Table 7: Classification accuracy of all tested deep learning models on a GPU

Classifier	Train Test Split	Training accuracy	Validation accuracy	Test accuracy
ResNet50	80 : 20 75 : 25 70 : 30	96.6 % 97 % 95.9 %	91.5% 92.6% 91.5%	91.5 % 92.2 % 95.1 %
VGG19	80 : 20 75 : 25 70 : 30	96.9 % 95.7 % 97.4 %	93.3% 93% 92.9%	93.3 % 92.4 % 95.8 %
EfficientNetB0	80 : 20 75 : 25 70 : 30	95.6 % 96 % 95.8 %	93.8% 93.5% 93.6%	92.9 % 92.8 % 93.1 %
Xception	80 : 20 75 : 25 70 : 30	84.8 % 84.5 % 84.8 %	82.7% 83.7% 84%	83.2 % 82.7 % 83 %

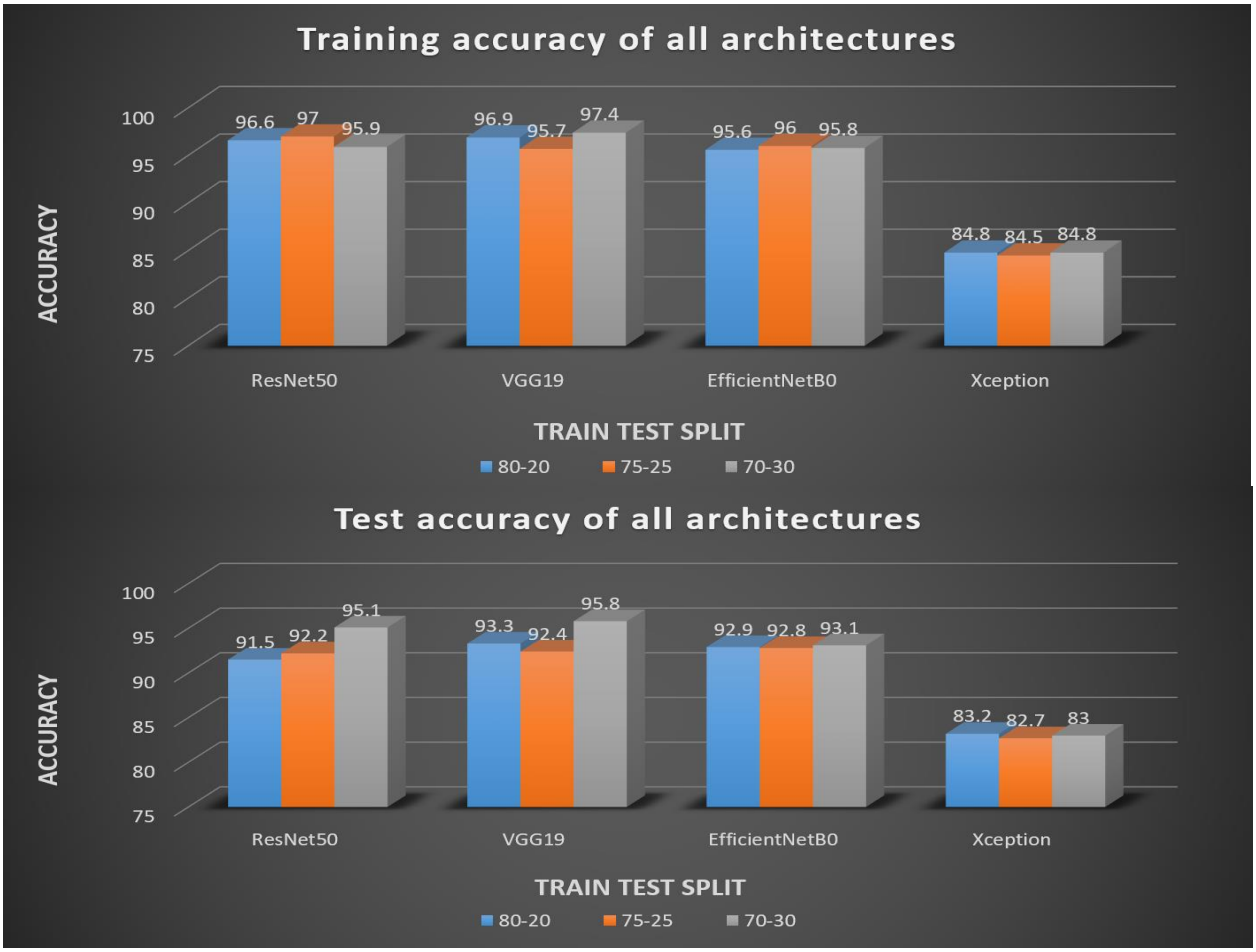
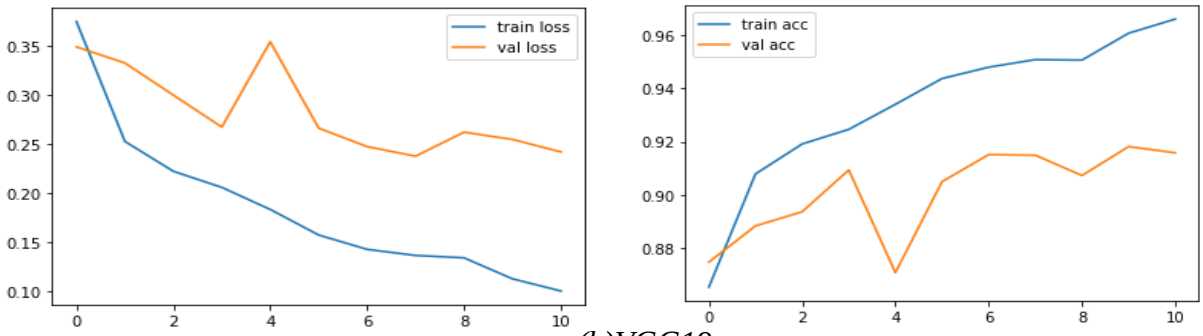
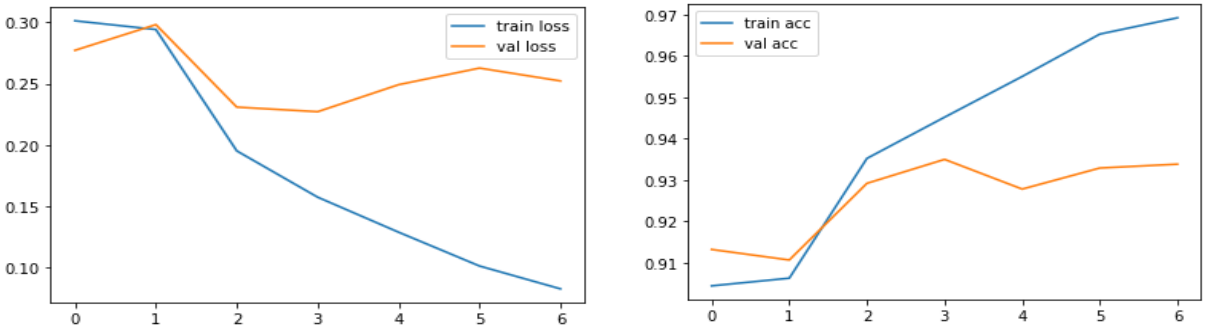


Figure 13 : Result comparison for training and testing phase of all architectures

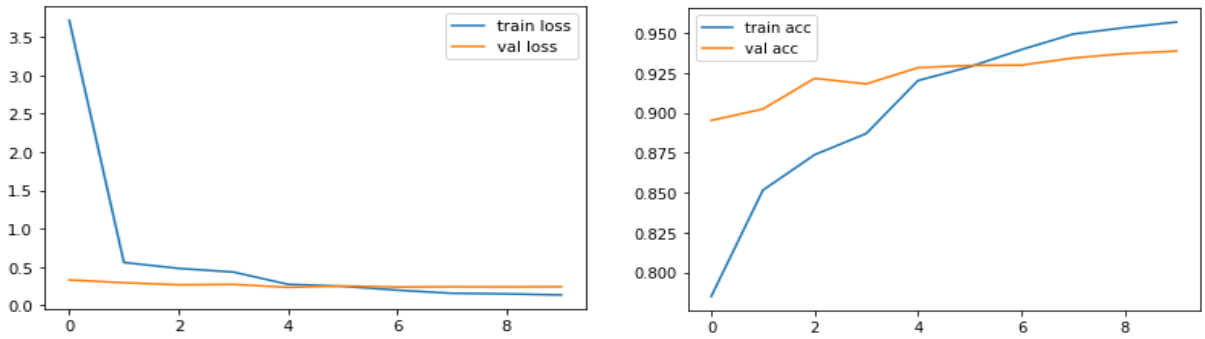
(a)ResNet50



(b)VGG19



(c)EfficientNetB0



(d)Xception

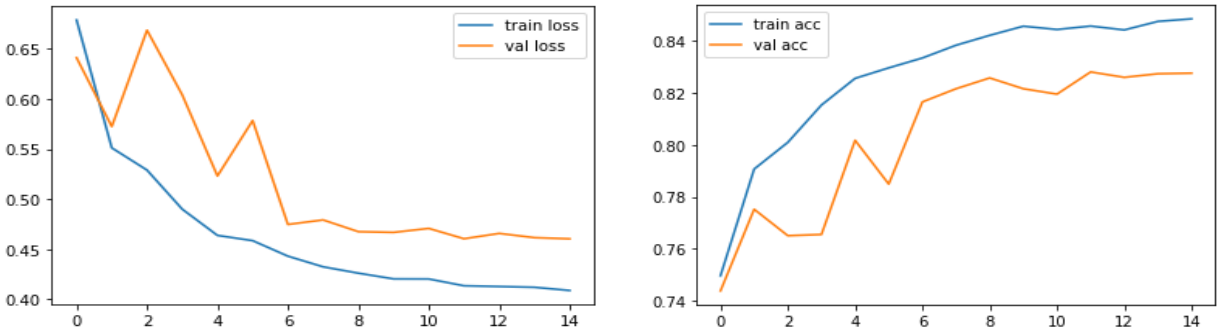
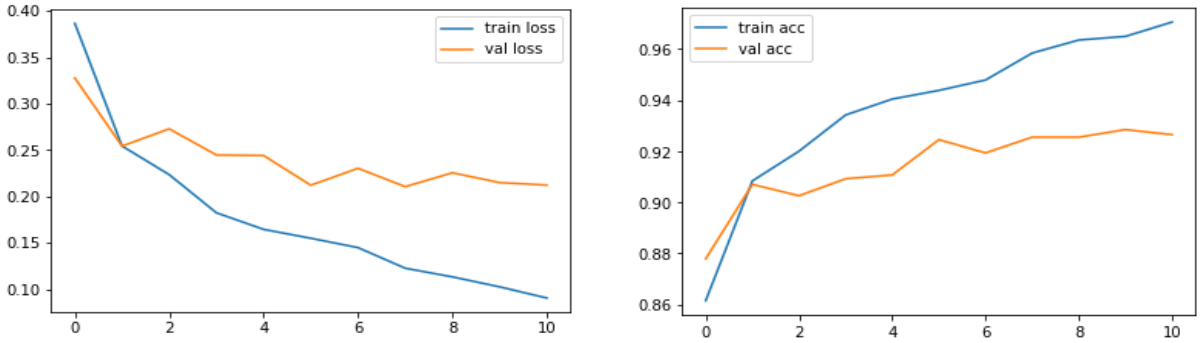
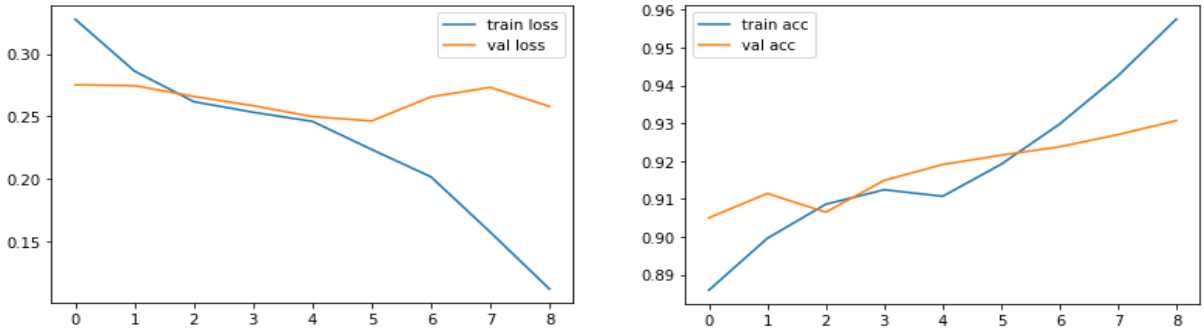


Figure 14 : Training loss and accuracy evaluation of all deep learning models using 80-20 train test split

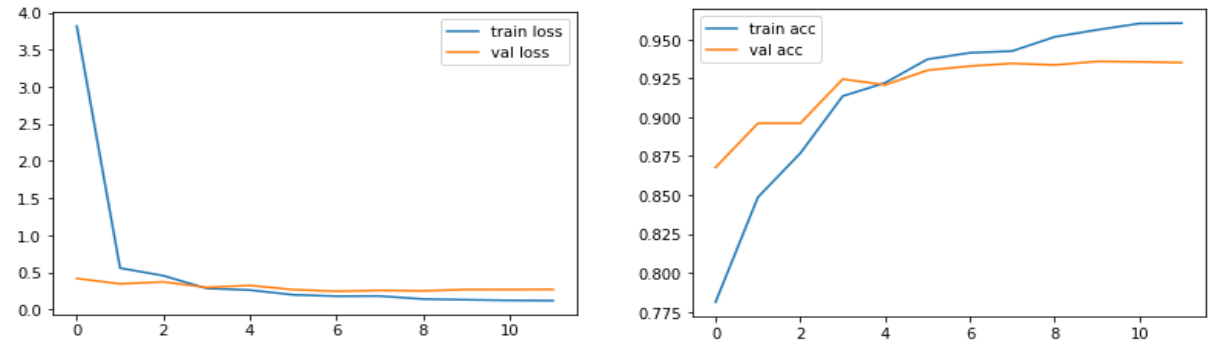
(a)ResNet50



(b)VGG19



(c)EfficientNetB0



(d)Xception

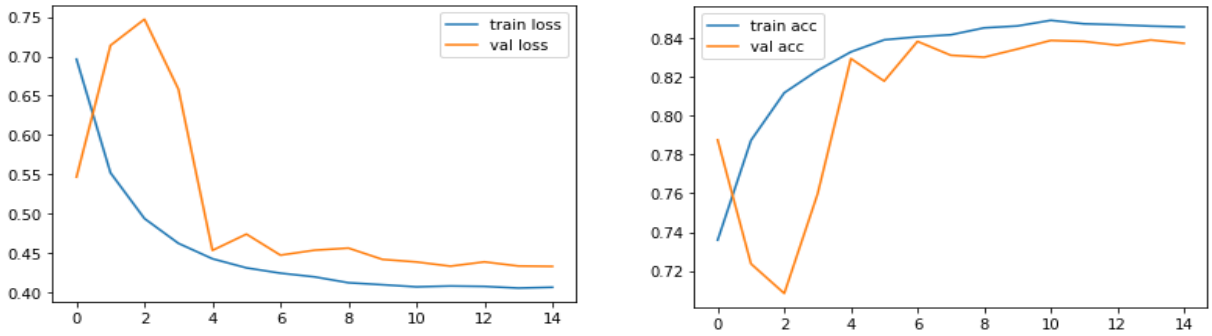
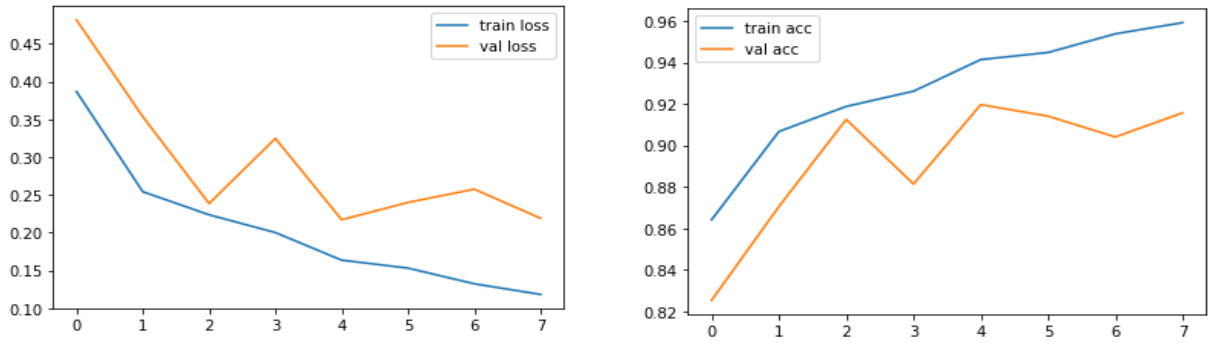
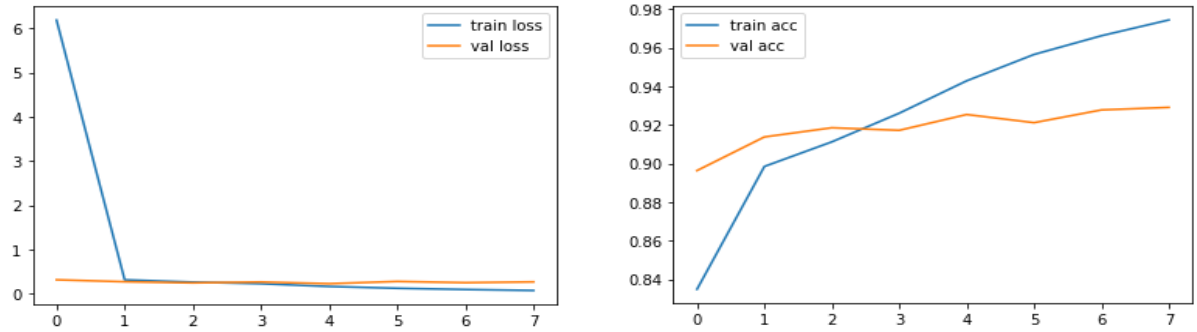


Figure 15 : Training loss and accuracy evaluation of all deep learning models using 75-25 train test split

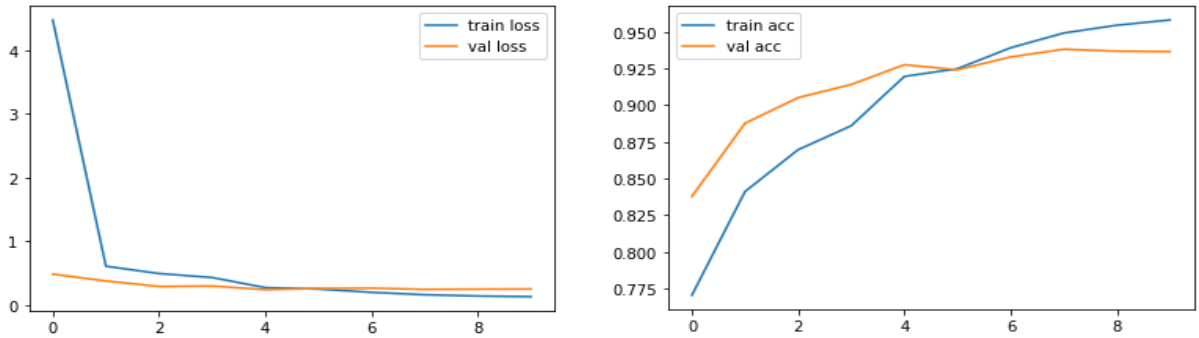
(a)ResNet50



(b)VGG19



(c)EfficientNetB0



(d)Xception

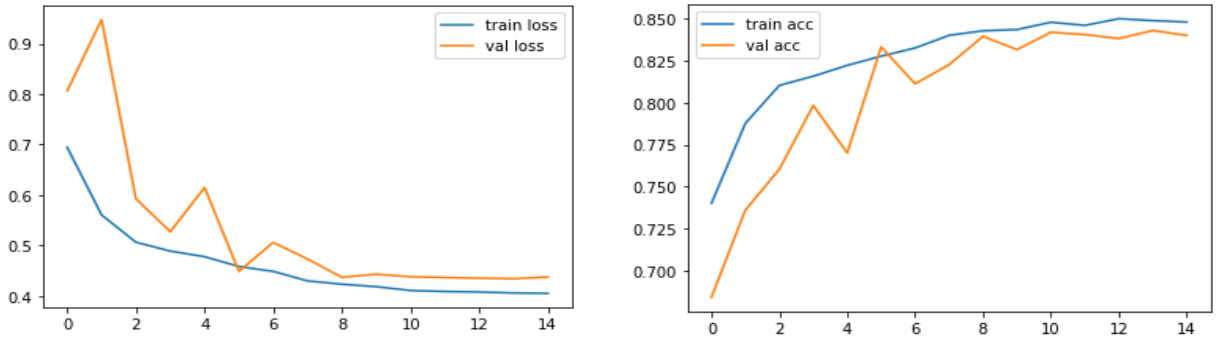
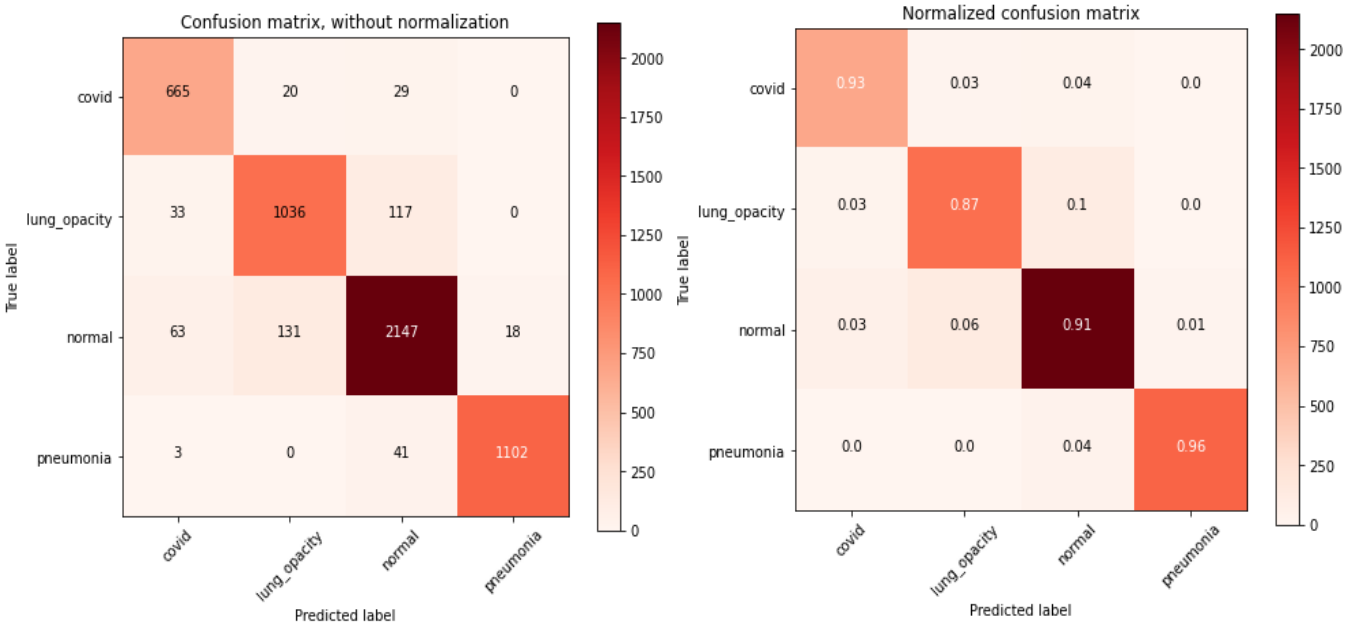
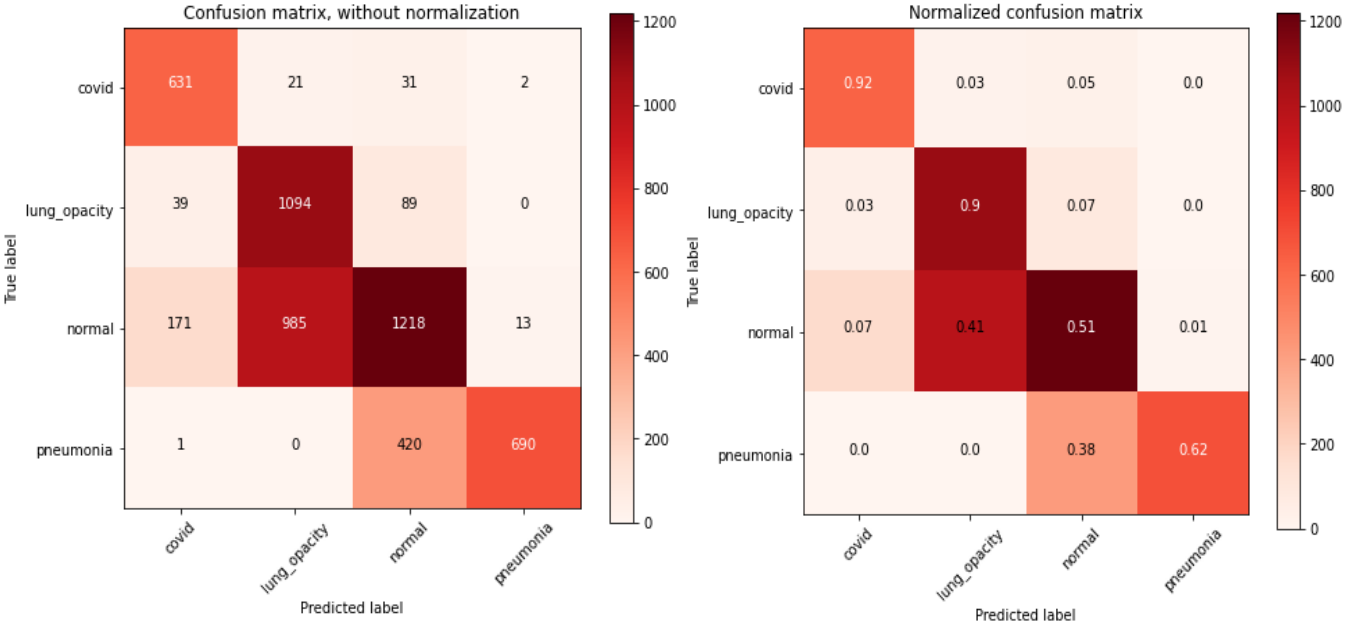


Figure 16 : Training loss and accuracy evaluation of all deep learning models using 70-30 train test split

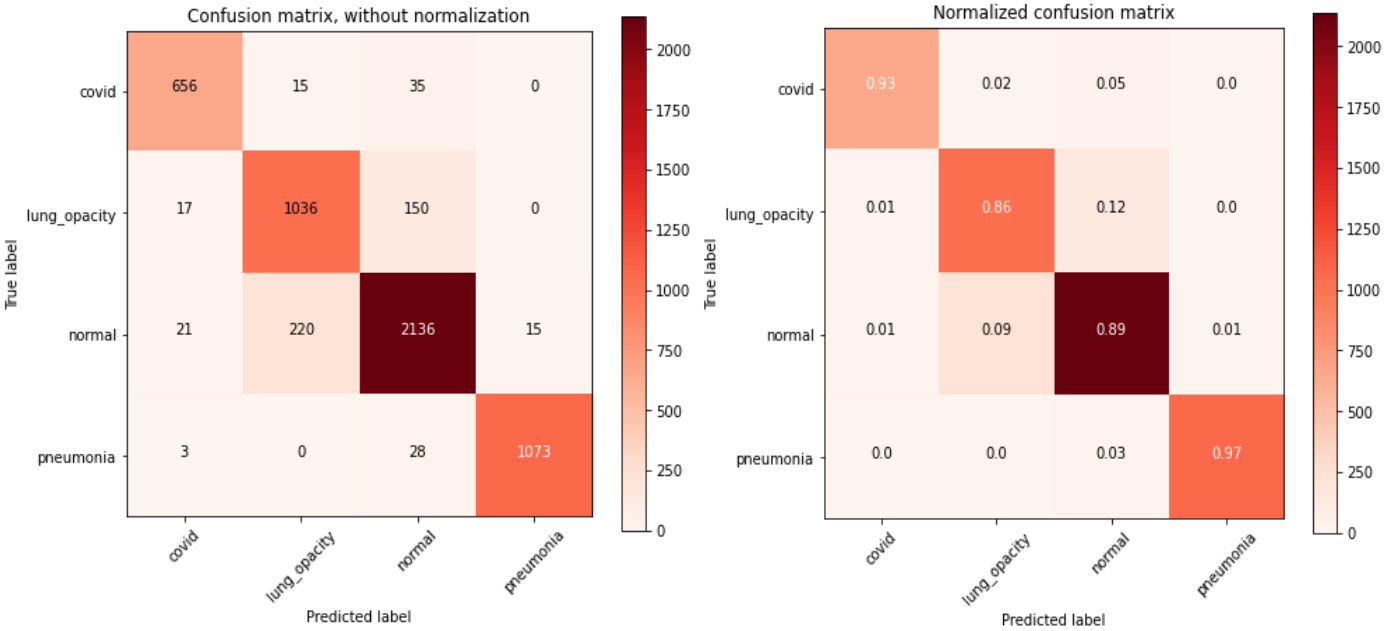
(a)ResNet50



(b)VGG19



(c)EfficientNetB0



(d)Xception

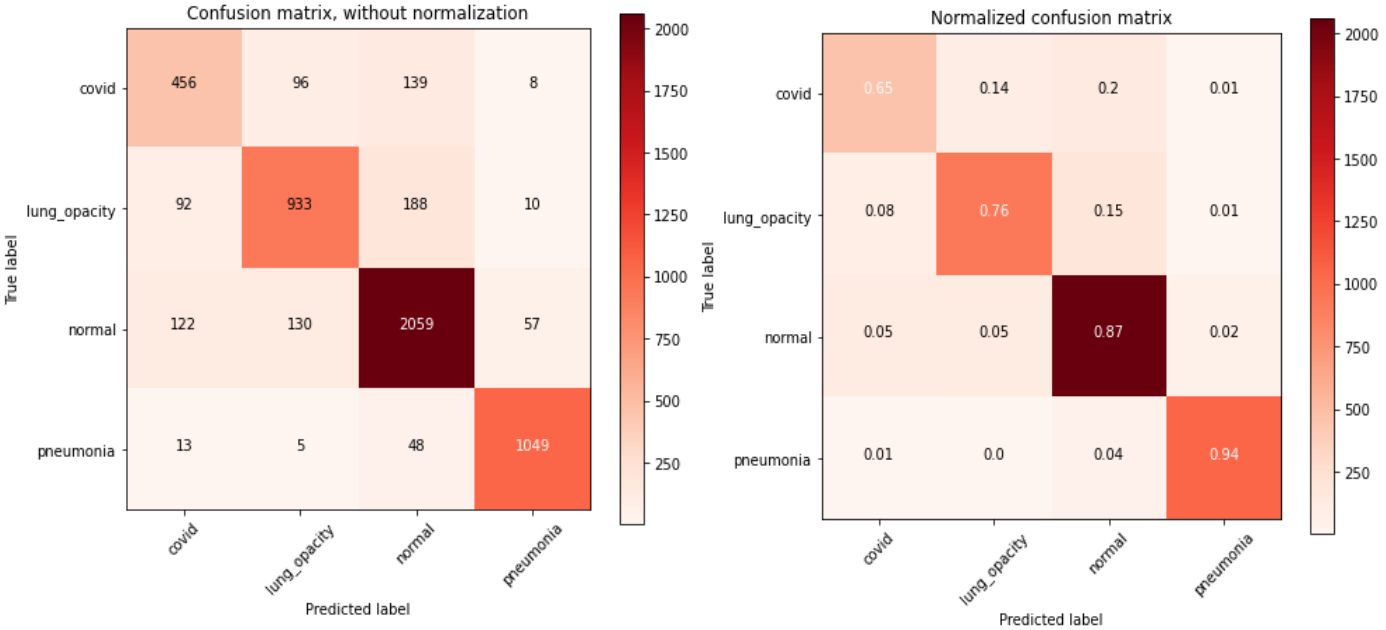
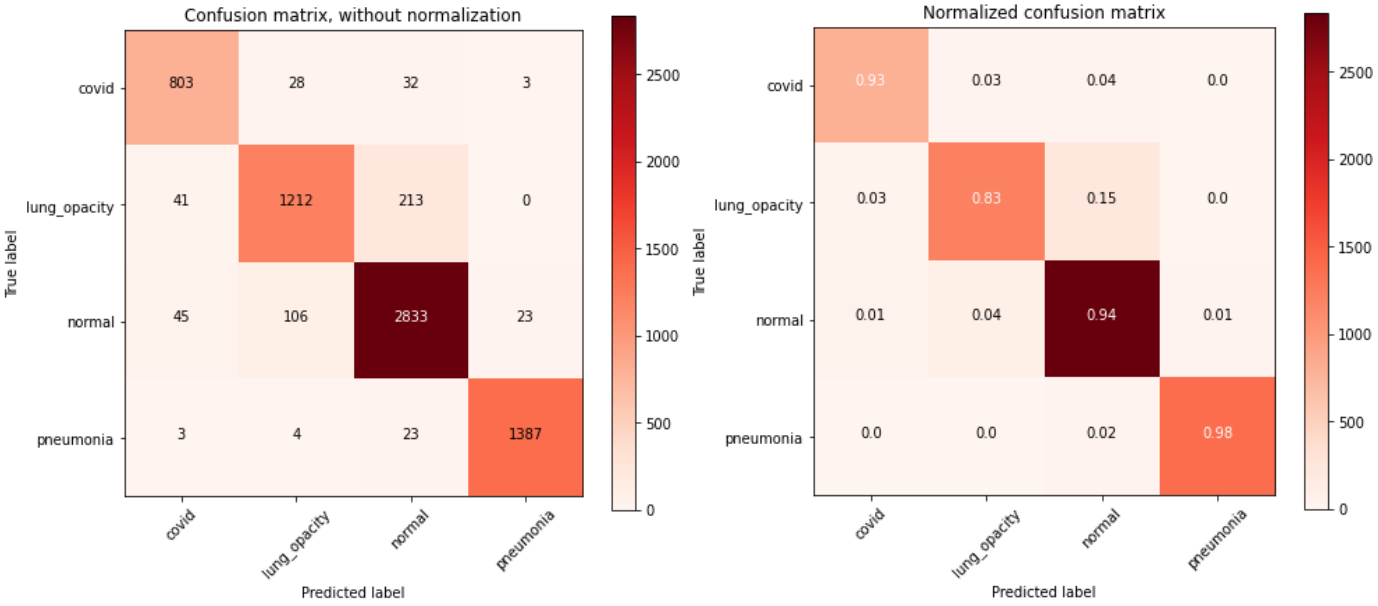
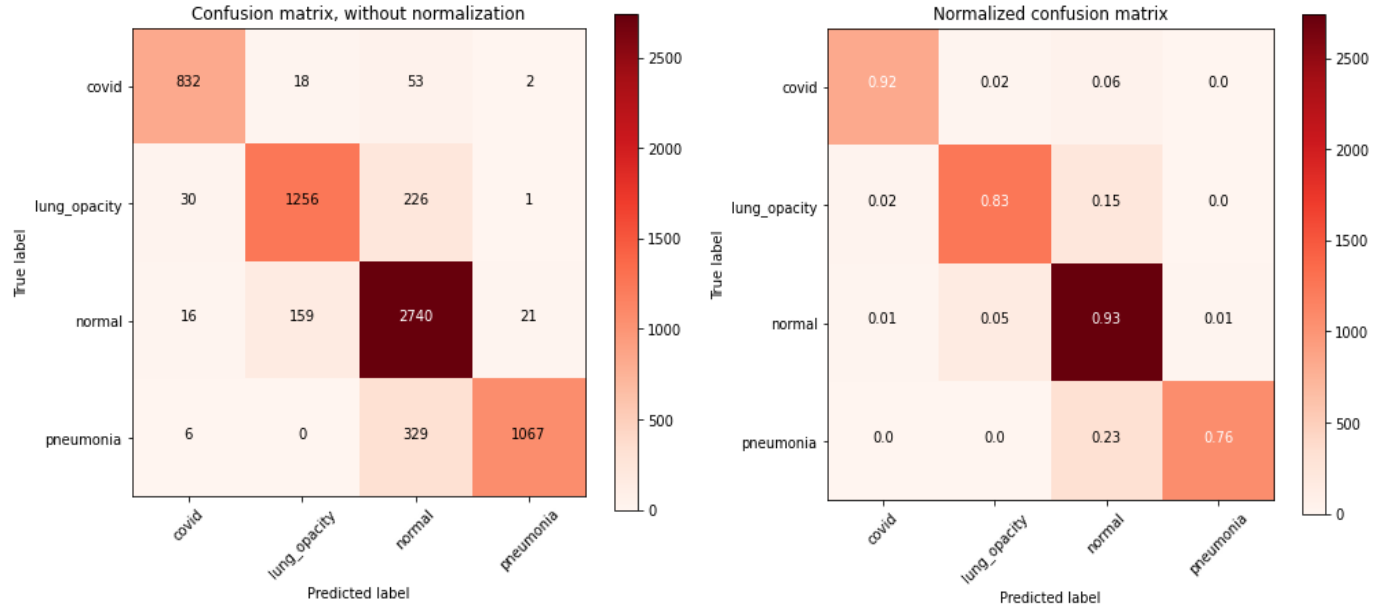


Figure 17 : Confusion matrix of all deep learning models using 80:20 train test split

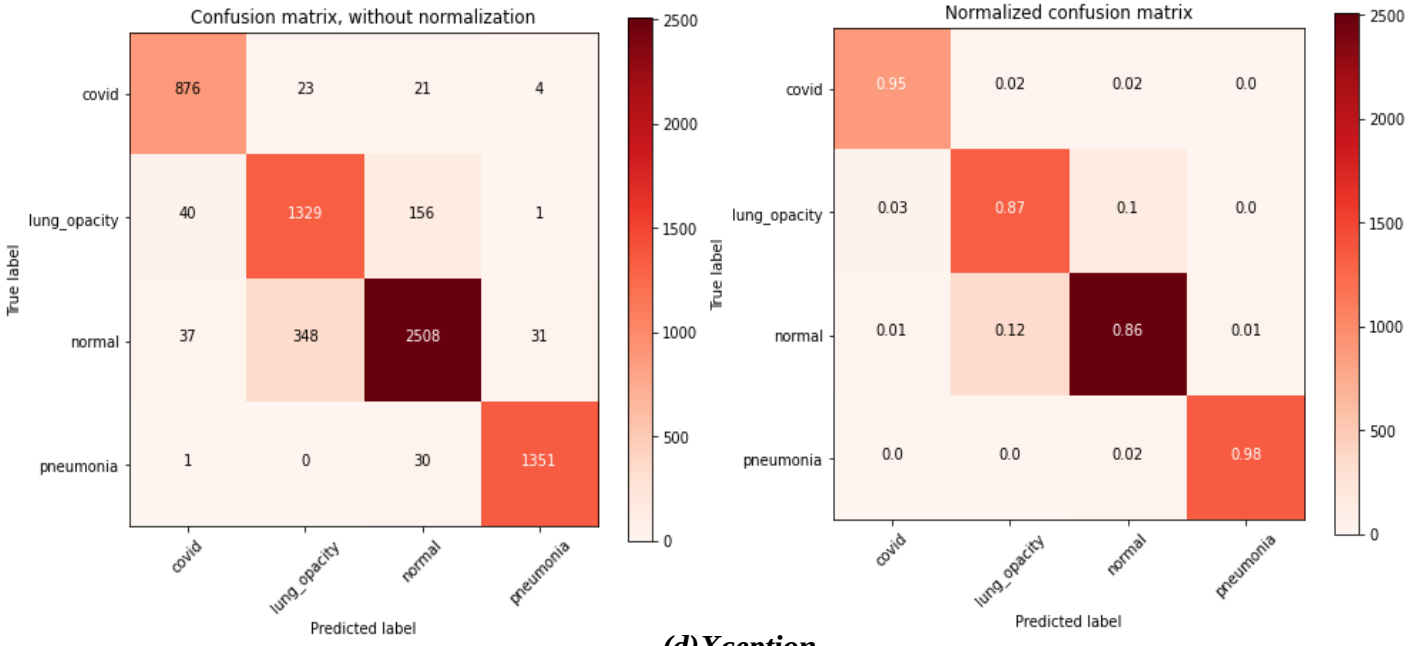
(a) ResNet50



(b) VGG19



(c) EfficientNetB0



(d)Xception

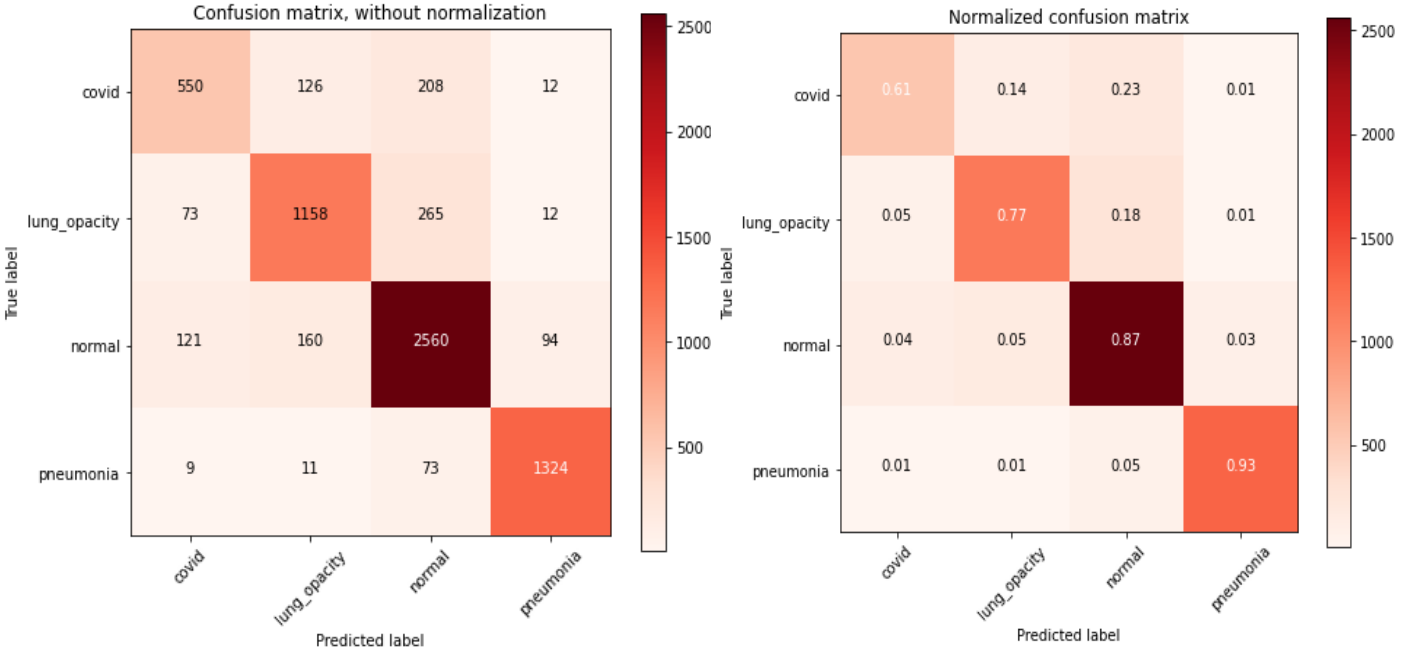
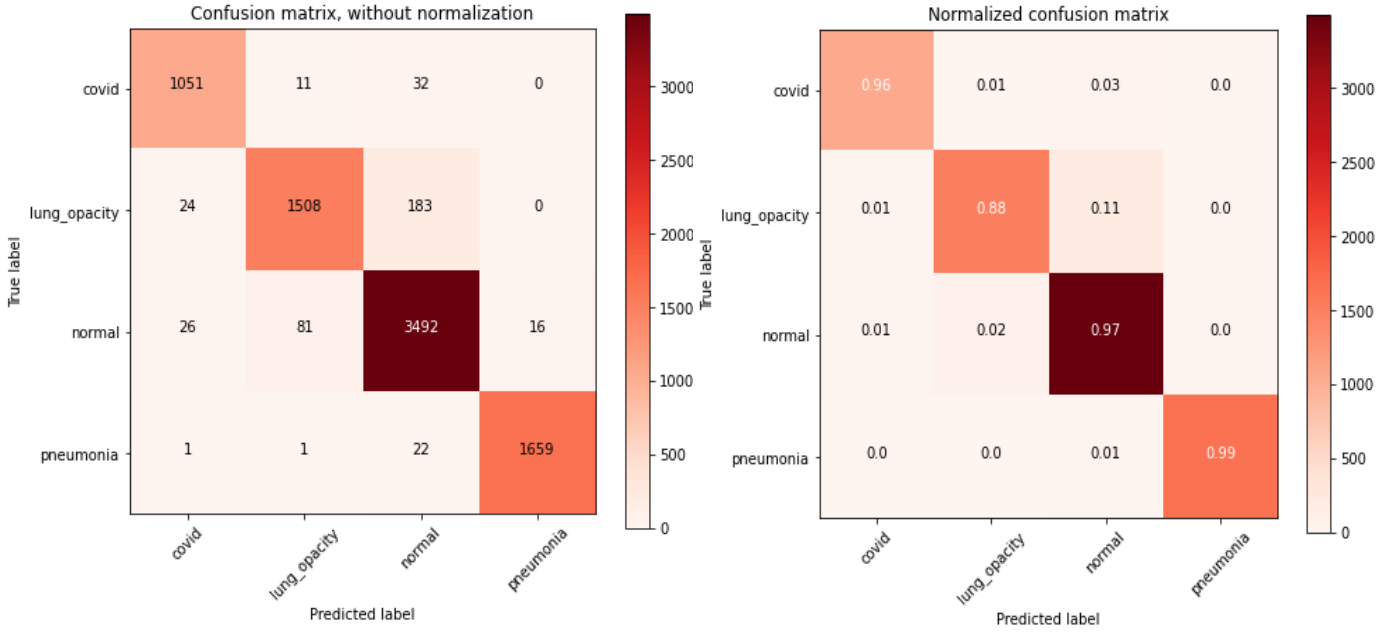
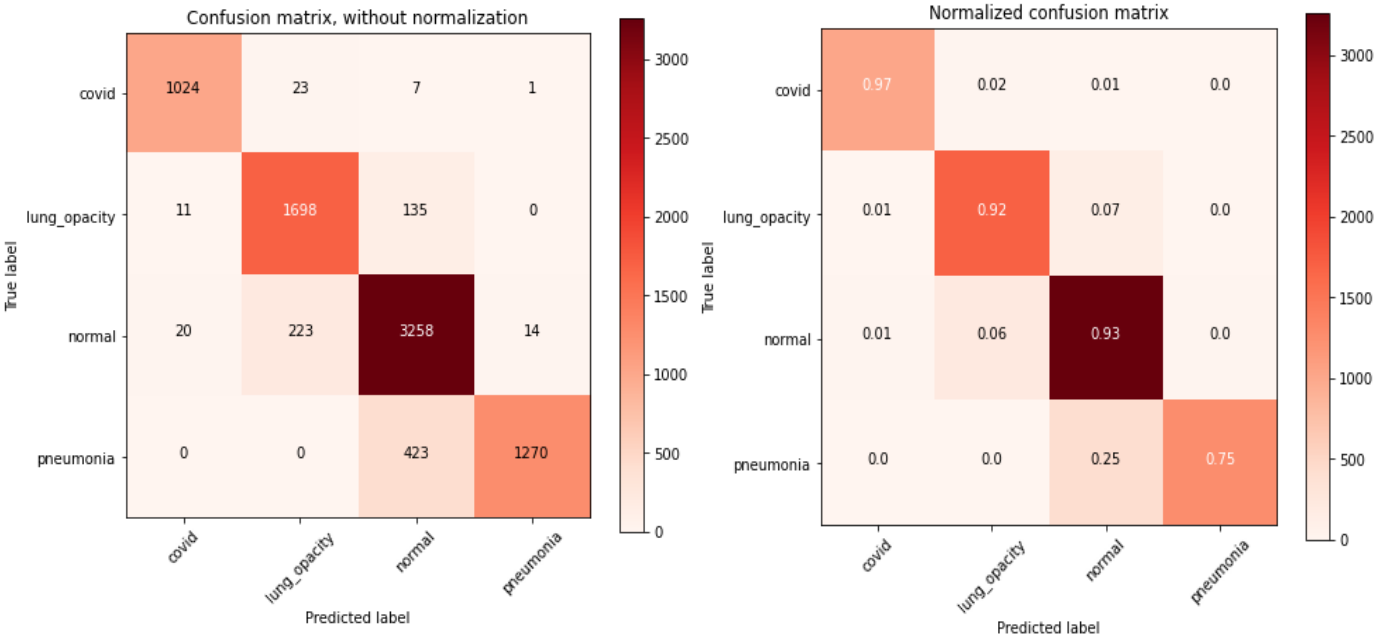


Figure 18 : Confusion matrix of all deep learning models using 75:25 train test split

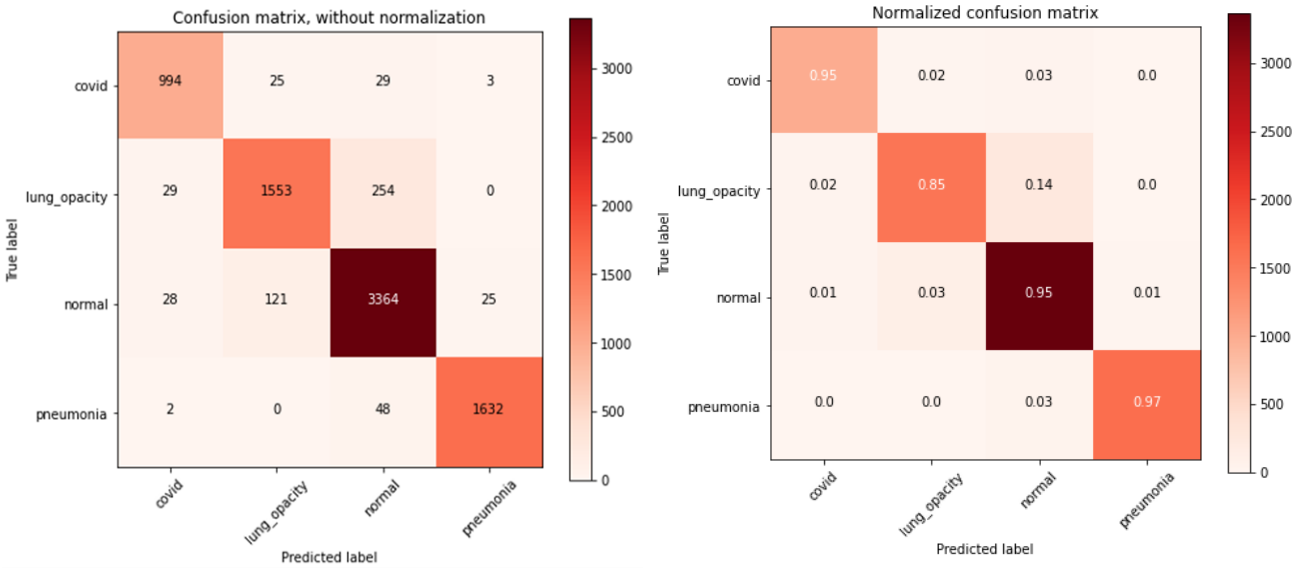
(a) ResNet50



(b)VGG19



(c)EfficientNetB0



(d)Xception

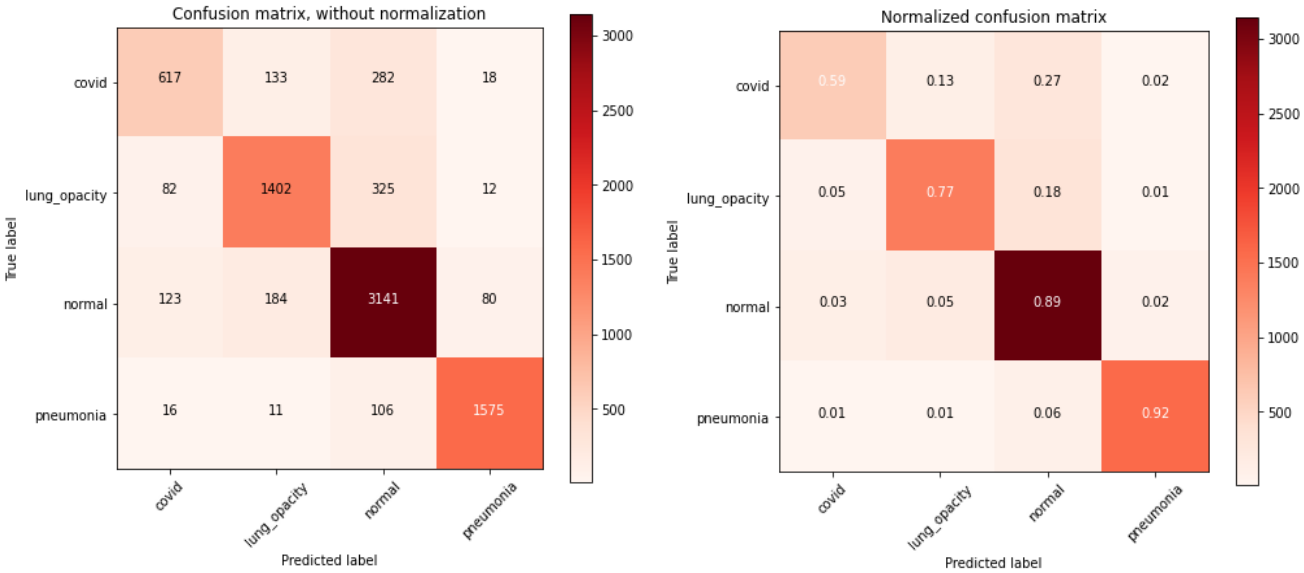


Figure 19 : Confusion matrix of all deep learning models using 70:30 train test split

Chapter 5

Conclusions and Future Work

5.1 Conclusions

As coronavirus already spread in many countries worldwide, early and accurate detection of this disease becomes very much necessary because of the shortage of medical facilities faced by almost every country in the world. So, at the end of this thesis paper, we have proposed a fully automated COVID-19 detection method using a deep neural network that eliminates the need to undergo the RT-PCR testing process but instead uses the chest X-ray images for classification which are more commonly available.

Four pre-trained deep learning models ResNet50, VGG19, EfficeintNetB0, and Xception are fine-tuned and later retrained to perform the classification of four different chest X-ray classes. It can be observed from Table 7 that all the architectures do a fairly good job in recognizing the classification of different chest infections. However, architecture 1(ResNet50) and architecture 2(VGG19) shine among other architectures with best training accuracy scores of around 96% and 97% and test accuracy scores of around 95% and 96%.

As seen in Table 4,5,6,7, all the architecture performs slightly better in the 70:30 train test split. Each of the performance parameters (accuracy, precision, recall, and F1 score) value is higher in the 70:30 train test split rather than in the 80:20 and 75:25 train test split.

It is evident from Table 4, 5 and 6 that architecture 1(ResNet50), architecture 2(VGG19), and architecture 3(EfficientNetB0) perform better in case of COVID-19 detection out of all chest infections in our multiclass image classification problem. Precision, recall, and F1 score all these performance parameter values are higher in the case of the COVID class.

5.2 Future Research Directions

There are a few areas where further improvement can be made thereby achieving better accuracy for this multiclass classification problem.

- In the future this work can be extended for a larger database with more than four classes to be classified.
- Although fine-tuning is applied in each of the architecture, improvement can be made by adding some new layers or modifying the layers of the model.
- To improve the computational time other lightweight deep learning models can be used. Apart from this, some optimization techniques can be used to select optimum features for classification to improve performance.

References

[1]Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Robyn L. Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng, CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning,

[2]Areej A Wahab Ahmed Musleh, Ashraf Yunis Maghari.Covid-19 detection in X-ray images using CNN algorithm

[3]Wang L, Wong A. COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images. arXiv preprint arXiv:2003.09871. 2020 Mar 22.

[4]Mobiny A, Cicalese PA, Zare S, Yuan P, Abavisani M, Wu CC, Ahuja J, de Groot PM, Van Nguyen H. Radiologist-Level COVID-19 Detection Using CT Scans with Detail-Oriented Capsule Networks. arXiv preprint arXiv:2004.07407. 2020 Apr 16.

[5]Asif Iqbal Khana,∗, Junaid Latief Shahb , Mohammad Mudasir Bhat. CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images

[6]Amit Kumar Jaiswal, Prayag Tiwari, Vipin Kumar Rathi, Jia Qian, Hari Mohan Pandey, Victor Hugo C. Albuquerque. COVIDPEN: A Novel COVID-19 Detection Model using Chest X-Rays and CT Scans

[7]Farooq M, Hafeez A. Covid-resnet: A deep learning framework for screening of covid19 from radiographs. arXiv preprint arXiv:2003.14395. 2020 Mar 31.

[8]Mahmud T, Rahman MA, Fattah SA. CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization. Computers in Biology and Medicine. 2020 Jun 20:103869.

[9] Umut Ozkaya1 , Saban Ozturk2 , Mucahid Barstugan1, Coronavirus (COVID-19) Classification using Deep Features Fusion and Ranking Technique

[10]Rohit Kundu 1, Hritam Basak 1, Pawan Kumar Singh 2, Ali Ahmadian 3,4*,Massimiliano Ferrara 4 & Ram Sarkar 5 Fuzzy rank-based fusion of CNN models using Gompertz function for screening COVID-19 CT-scans

[11]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Microsoft Research Deep Residual Learning for Image Recognition

[12]Narin A, Kaya C, Pamuk Z. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. arXiv preprint arXiv:2003.10849. 2020 Mar 24.

[13] Zhang J, Xie Y, Li Y, Shen C, Xia Y. Covid-19 screening on chest x-ray images using deep learning-based anomaly detection. arXiv preprint arXiv:2003.12338. 2020 Mar 27.

[14]Brunese L, Mercaldo F, Reginelli A, Santone A. Explainable Deep Learning for Pulmonary Disease and Coronavirus COVID-19 Detection from X-rays. Computer Methods and Programs in Biomedicine. 2020 Jun 20:105608

[15]Hemdan EE, Shouman MA, Karar ME. Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. arXiv preprint arXiv:2003.11055. 2020 Mar 24

[16]. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. InProceedings of the IEEE conference on computer vision and pattern recognition 2015.

[17]Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional Networks.

[18]Karim M, Döhmen T, Rebholz-Schuhmann D, Decker S, Cochez M, Beyan O. Deepcovidexplainer: Explainable covid-19 predictions based on chest x-ray images. arXiv preprint arXiv:2004.04582. 2020 Apr 9.

[19]Ezzat D, Ella HA. GSA-DenseNet121-COVID-19: a hybrid deep learning architecture for the diagnosis of COVID-19 disease based on gravitational search optimization algorithm. arXiv preprint arXiv:2004.05084. 2020 Apr 9.

[20]Ferdin Joe John Joseph, Sarayut Nonsiri, Annop Monsakul.Keras and TensorFlow: A Hands-On Experience

[21]M. Rampurawala, “Classification with TensorFlow and Dense Neural Networks”, Accessed on:May.14,2020.[Online].Available:https://heartbeat.fritz.ai/classification-with-tensorflow-and-dense-neural-networks-8299327a818a

[22]https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset

[23]https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

[24]Ejaz Khan 1, Muhammad Zia Ur Rehman 2 , Fawad Ahmed 3, Faisal Abdulaziz Alfouzan 4 ,Nouf M. Alzahrani 5 and Jawad Ahmad 6,”Chest X-ray Classification for the Detection of COVID-19 Using Deep Learning Techniques”