An efficient deep learning approach for detecting lung disease from chest X-ray images using transfer learning and ensemble modeling

by

Mostofa Kamal Sagor
17301106
Ishrat Jahan
17101458
Susmita Chowdhury
17101025
Rubayet Ansary
20241050

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mostofa Kamal Sagor	[illegible]
Mostofa Kamal Sagor 17301106	Ishrat Jahan 17101458
Susmita Chowdhury	Rubayet Ansary
Susmita Chowdhury 17101025	Rubayet Ansary 20241050

Approval

The thesis/project titled "An efficient deep learning approach for detecting lung disease from chest X-ray images using transfer learning and ensemble modeling" submitted by

1. Mostofa Kamal Sagor (17301106)
2. Ishrat Jahan (17101458)
3. Susmita Chowdhury (17101025)
4. Rubayet Ansary (20241050)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 15, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Md. Ashraful Alam, PhD
Assistant Professor
CSE Department
BRAC University

Program Coordinator:
(Member)

[illegible]

Md. Golam Rabiul, PhD
Associate Professor
CSE Department
BRAC University

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Mahbubul Alam Majumdar, PhD
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Among the most convenient bacteriological assessments for the diagnosis and treatment with several health complications is the chest X-Ray. The World Health Organization (WHO) estimates, for instance, that pneumonic plague induces between 250,000 to 500,000 fatalities annually. Pneumonia and flu are serious challenges towards global health as well as being a source of significant death rates globally. [1]. In X-Ray imaging, it is a common technique to standardize the extracted image reconstruction with usual uniform disciplines taken before the study. Unfortunately, there has been relatively little study on several separate lung disease monitoring, including X-Ray picture analysis and poorly labelled repositories. Our paper suggests an effective approach for the detection of lung disease trained on automated chest X-ray images that could encourage radiologists in their moral choice. Besides, with a weighted binary classifier, a particular technique is also deployed that will optimally leverage the weighted predictions from optimal deep neural networks such as InceptionV3, VGG16 and ResNet50. In addition to the existing, transfer learning, along with more rigorous academic training and testing sets, is used to fine-tune deep neural networks to achieve higher internal processes. In comparison, 88.14 percent test accuracy was obtained with the final proposed weighted binary classifier, where other models give us about 76.91 percent average accuracy. For a brief recurring diagnosis, the legally prescribed procedure may also be used which may increase the course of the same condition for physicians. For a prompt diagnosis of pneumonia, the suggested approach should be used and can improve the diagnosis process for health practitioners.

Keywords: lung diseases; chest X-ray images; convolution neural network (CNN); deep learning; transfer learning; diagnostics facilitated by electronics

Acknowledgement

First of all, praise to the Almighty Allah for whom we have been able to finish our Thesis work in due time without any major interruptions.
That said, we would like to express the deepest appreciation to our respected faculty and supervisor, Dr. Md. Ashraful Alam, for his support and continuous supervision which led us to complete our project. Also, we are very much thankful to our co-supervisor, Md. Tanzim Reza, who has willingly helped us out a lot with his utmost abilities in this thesis.

Table of Contents

Declaration **i**

Approval **ii**

Abstract **iii**

Acknowledgment **iv**

Table of Contents **v**

List of Figures **vii**

List of Tables **ix**

Nomenclature **ix**

1 Introduction **1**

1.1 Motivation 1
1.2 Aims and Objectives 1
1.3 Research Methodology 1
1.4 Thesis Orientation 2

2 Methodology **3**

2.1 Related Work 3
2.2 Our Proposed Model 4
2.2.1 Convolutional Layer 5
2.2.2 Activation Functions 5
2.2.3 Average Pooling Layer 7
2.2.4 Flatten Layer 7
2.2.5 Dropout Layer 7
2.2.6 Fully Connected Layer 8
2.2.7 VGG16 8
2.2.8 Inception-v3 9
2.2.9 ResNet-50 9
2.2.10 Sequential Model Representation of Inception-v3, VGG16 and ResNet50 14
2.2.11 Ensemble Modeling 14
2.3 Workflow Diagram 16
2.3.1 Transfer Learning Phase 17
2.3.2 Training and Cross-Validation Phase 17

2.3.3 Testing Phase 18

3 Implementation **19**
3.1 Source of Datasets 19
3.2 Data Samples 19
3.3 Data Visualizations 20
3.4 Dataset Classification 21
3.4.1 Training Set 21
3.4.2 Validation set 21
3.4.3 Testing Set 21
3.5 Data Pre-processing 21
3.5.1 Resize Images 21
3.5.2 Normalization and Scaling Images 22
3.5.3 Data Augmentation 22

4 Results and Discussion **24**
4.1 Individual Model 24
4.2 Before Transfer Learning into "ENSEMBLE BETA" 24
4.3 After Transfer Learning into "ENSEMBLE BETA" 25
4.4 Learning Curve 25
4.5 Comparison with Other Models 26

5 Conclusion and Future work **28**

Bibliography **31**

List of Figures

2.1 Ensemble Model using InceptionV3, VGG16 and ResNet50. Using the features extracted from these three models we leveraged fully connected layers and averaging layer for the final prediction 4
2.2 An example of convolution operation done by the convolutional layer on an input image of dimension 5×5 with a kernel of dimension 3×3. The area where the kernel is applied and the result of the dot multiplication is shown in red color in each step of the figures 5
2.3 ReLU activation function plot 6
2.4 The matrix on the left is the input upon which average pooling is being applied. The one on the right is the resultant down-sampled matrix. In this example, the filter dimension is 2×2 and stride is 2. The average value of each colored area is being taken and the output values are also color coded to represent the area from where it was taken 7
2.5 Fully Connected Layer 8
2.6 The detailed architecture of VGG16 with the number of kernels in convolutional layers and activation functions used in each layer is shown in this figure. The output dimension of each layer is also shown here 9
2.7 The detailed architecture of inception-v3. The figures in a, b, c are taken from [24]. The layers named **Inception1**, **Inception2** and **Inception3** refers to the networks shown in a, b, c 10
2.8 Residual Learning 11
2.9 11
2.9 12
2.9 This figure shows the detailed architecture of ResNet-50 model with each stage of the model shown in sub-figures a, b, c, d, e. Stages 1, 2, 3, 4 and 5 are then combined in sub-figure f as the main model . . 13
2.10 Illustration of sequential models 14
2.11 Basic structure of an ensemble model 14
2.12 Illustration of the output layers of the ensemble model hierarchy . . . 15
2.13 Workflow diagram for our model training and evaluation 16
2.14 Transfer learning phase 17
2.15 Training and cross validation phase 17
2.16 Testing phase 18

3.1 The first row of images are X-Ray images of patients whose lungs are in normal condition (not pneumonia affected). The second row of images are the X-Ray images of lungs of pneumonia patients 19
3.2 Illustration of data balance using bar chart of "Labels(Normal, Abnormal)" column of Chest X-Ray8 dataset 20
3.3 Illustration of correlation mapping of different attributes or features with the label or class of Chest X-Ray8 dataset 20
3.4 Pre-processing phases of the dataset 22

4.1 Validation accuracy curve of Ensemble Beta 25
4.2 Validation loss curve of Ensemble Beta 26
4.3 Illustration of different models versus our model using a bar chart . . 27

List of Tables

2.1 Probability calculation in softmax function 7
2.2 Predicted output label of the 16-sample input chest X-Ray images . . 18

4.1 Validation accuracy and loss in the experimented individual model . . 24
4.2 Validation accuracy and loss before transfer learning into the ensemble model after 40 epochs 24
4.3 Validation accuracy and loss after transfer learning into the ensemble model after 40 epochs 25
4.4 Comparison of others model versus our model 26

Chapter 1

Introduction

1.1 Motivation

Lung diseases are a pretty common health problem though out the world. Usually, chest X-Rays are used to detect various lung diseases. A trained radiologist looks for certain signs in chest X-Ray images that may indicate that one or several diseases have infected the lung. But this is a slow process. Some of the patients are also misdiagnosed due to the radiologists' mistakes. Several lung diseases also cause permanent damage to the lungs. Also, coronavirus disease is now the most common and most concerning lung disease that is affecting millions of people around the world. To tackle this problem, we propose a technique that leverages a deep transfer learning algorithm and ensemble approach that improves upon the existing models for automatically recognizing the existence of lung diseases from chest X-Ray images.

1.2 Aims and Objectives

We want to build a deep learning model leveraging various convolutional neural network models and using the technique of transfer learning and ensembling. The state-of-the-art model achieved an accuracy of 78.73%. Our objective is to build a model that can beat this accuracy.

1.3 Research Methodology

Firstly, we have observed several models individually and we have attempted with Inception-v3, VGG16, VGG19, ResNet-50 and ResNet-101 [2]. We later attached the convolutional layers and dense layers to the output layer or last layer of those three models ("mixed10", "block pool5", "avg pool" for Inception-v3, VGG16 and ResNet-50 respectively). We pre-processed the X-Ray image data into a well-defined form of $224 \times 224 \times 3$ in the initial stage. The last layer of the selected models (Inception-v3, VGG16 and ResNet-50) was then selected as a transfer layer and a sequential model was initiated by adding convolutional layers, flatten layer, dropout layer, fully connected layer and a two neuron output layer for classification.

1.4 Thesis Orientation

At the beginning of chapter 2, we have talked about previous work done in this area. Then we provide an overview of our algorithm. Then, we describe each component/step of our model/algorithm. Finally, we discuss how all of these components come together to build the final model. In chapter 3, we describe the implementation side of our thesis. After illustrating how our datasets are distributed, we go on to describe the pre-processing techniques applied to the used dataset. In chapter 4, we present the results achieved using our model and compare them with previous models.

Chapter 2

Methodology

2.1 Related Work

Around 15 percent of cases with COVID-19 are serious. In a hospital, that means they may need to be treated with oxygen. Approximately 5% of individuals have serious infections and require a ventilator. There is also a disorder called acute respiratory distress syndrome in people who get pneumonia (ARDS). It's an illness that develops suddenly and causes issues with breathing. Extreme inflammation in your lungs triggers the latest coronavirus. It affects the cells and tissue in your lungs that line the air sacs. These sacs are where you process the oxygen you breathe and transmit it to your blood. Tissue breaks off and clogs your lungs because of the damage. It will thicken the walls of the bags, making it very difficult for you to breathe [3].

In recent times, mainly CNN-based algorithms have been used to solve medical image classification related problems. SegNet [4], U-Net [5], AlexNet [6], GoogLeNet [7], VGGNet-16 [8], ChestNet [9], CardiacNet [10] and ResNet-50 [11] are a some of the more well-known models for medical image related classification problems. For determining optimum network hyper-parameters, models like evolutionary-based algorithms [12], BPNN [13] (a multi-layer supervised feed-forward neural network), CpNN [13] (an unsupervised simple neural network with two layers) and reinforcement learning have been developed. For conducting lung nodule detection [14] and pulmonary tuberculosis classification, these algorithms are regularly used.

In addition, the majority vote of a jury of experts acted as a benchmark on the confirmation collection of Chest Radiographs Classification. The reliability of CheXNet on the validation range was contrasted with the level of performance of 9 medical experts using the AU-ROC as the measuring instrument. The average time to go through and classify the around 400 images in the validation set was noticeably longer for the radiologists than for the automated model. The main problem of this study was that both CheXNet and radiologists were not allowed to use patients' previous data and this experiment was limited to a dataset from a single institution [15].

However, in recent studies [16], multi-layer, probabilistic, learning vector quantization, and generalized regression neural networks have been used for diagnosis of chest and lung diseases. The diagnosis of lung diseases such as TB, pneumonia, etc. using chest radiographs in [17] was implemented using a neural network for grouping after pre-processing images using normalization. The research works described in this paragraph had been used effectively in classifying diseases but their performance was not up to par with the contemporary deep learning models.

2.2 Our Proposed Model

When we were trying to develop the model, the first focus was how to predict from more than one model and take the average probability to predict the final output label or class. As a consequence, we end up developing an ensemble model of three well known convolutional neural networks. First of all, we have observed several models individually and we have attempted with Inception-v3, VGG16, VGG19, ResNet-50 and ResNet-101. However, we decided to select only three based on their performance over the Pneumonia dataset while training. Because of resource limitation, we could not select more than three models for performing ensemble operation. Later on, we have added a flatten layer, a dropout layer and 2 dense layers as well after the last convolutional layer of those three models (for InceptionV3, VGG16 and ResNet50, they are "mixed10", "block_pool5", "avg_pool" respectively). Lastly, we have taken the output from these sequential models as the input of an averaging layer and considered the output layer of that averaging layer as our desired classification categories.

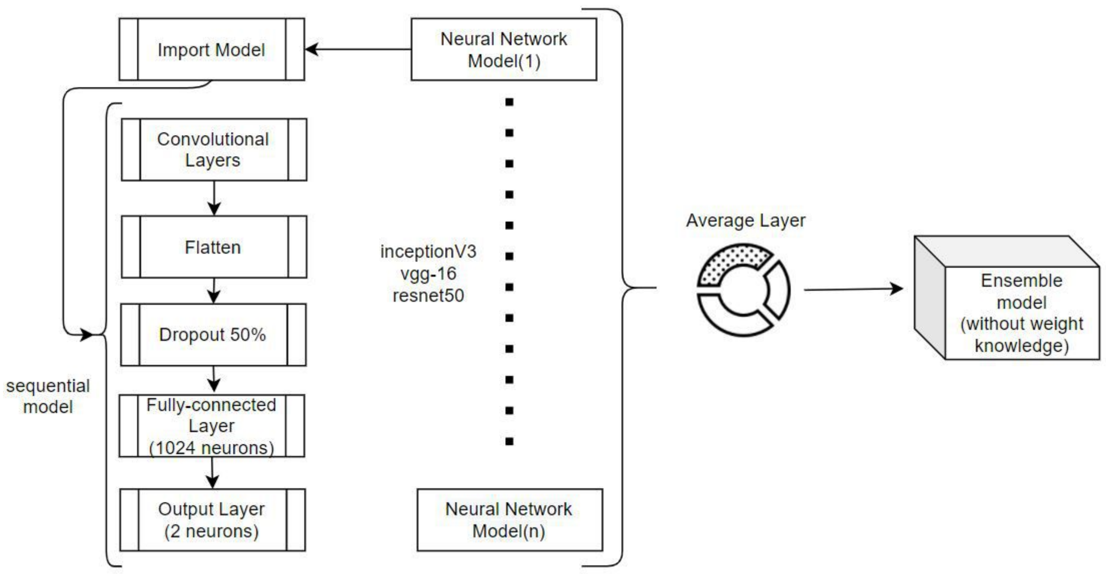


Figure 2.1: Ensemble Model using InceptionV3, VGG16 and ResNet50. Using the features extracted from these three models we leveraged fully connected layers and averaging layer for the final prediction

We will first describe the various components and layers of our training model before moving onto the main part of our algorithm. Each of these layers has different parameters that can be optimized.

2.2.1 Convolutional Layer

A convolutional layer is the major building block of any Convolutional Neural Network. A convolutional layer consists of several filters/kernels of a specific dimension and the whole network tries to learn the value of these filters. The filter is applied by sliding it across the input image and performing dot multiplication (Figure 2.2). The output values of the dot multiplication depend on the shapes that appear on the image. The filter size is usually much smaller than an input image. Thus, the network tries to learn filters that will respond to a specific region of the input enabling it to detect shapes and patterns in a specific image area. Convolutional layers are heavily used in medical image analysis related problems [18], [19].

$$\begin{array}{|c|c|c|c|c|}\hline 5&3&4&2&1\\\hline 10&8&7&5&5\\\hline 5&5&5&5&5\\\hline 3&2&1&1&5\\\hline 4&6&5&1&0\\\hline\end{array} \circledast \begin{array}{|c|c|c|}\hline 2&-1&1\\\hline 3&0&1\\\hline -2&-1&0\\\hline\end{array} = \begin{array}{|c|c|}\hline 33& \\\hline & \\\hline\end{array}$$

(a)

$$\begin{array}{|c|c|c|c|c|}\hline 5&3&4&2&1\\\hline 10&8&7&5&5\\\hline 5&5&5&5&5\\\hline 3&2&1&1&5\\\hline 4&6&5&1&0\\\hline\end{array} \circledast \begin{array}{|c|c|c|}\hline 2&-1&1\\\hline 3&0&1\\\hline -2&-1&0\\\hline\end{array} = \begin{array}{|c|c|}\hline 33&18\\\hline & \\\hline\end{array}$$

(b)

$$\begin{array}{|c|c|c|c|c|}\hline 5&3&4&2&1\\\hline 10&8&7&5&5\\\hline 5&5&5&5&5\\\hline 3&2&1&1&5\\\hline 4&6&5&1&0\\\hline\end{array} \circledast \begin{array}{|c|c|c|}\hline 2&-1&1\\\hline 3&0&1\\\hline -2&-1&0\\\hline\end{array} = \begin{array}{|c|c|}\hline 33&18\\\hline 6& \\\hline\end{array}$$

(c)

$$\begin{array}{|c|c|c|c|c|}\hline 5&3&4&2&1\\\hline 10&8&7&5&5\\\hline 5&5&5&5&5\\\hline 3&2&1&1&5\\\hline 4&6&5&1&0\\\hline\end{array} \circledast \begin{array}{|c|c|c|}\hline 2&-1&1\\\hline 3&0&1\\\hline -2&-1&0\\\hline\end{array} = \begin{array}{|c|c|}\hline 33&18\\\hline 6&7\\\hline\end{array}$$

(d)

Figure 2.2: An example of convolution operation done by the convolutional layer on an input image of dimension 5×5 with a kernel of dimension 3×3. The area where the kernel is applied and the result of the dot multiplication is shown in red color in each step of the figures

2.2.2 Activation Functions

An activation function [20] is used to decide whether a neuron should be activated or not. The output values of a network layer can be anything between $-\infty$ to ∞.

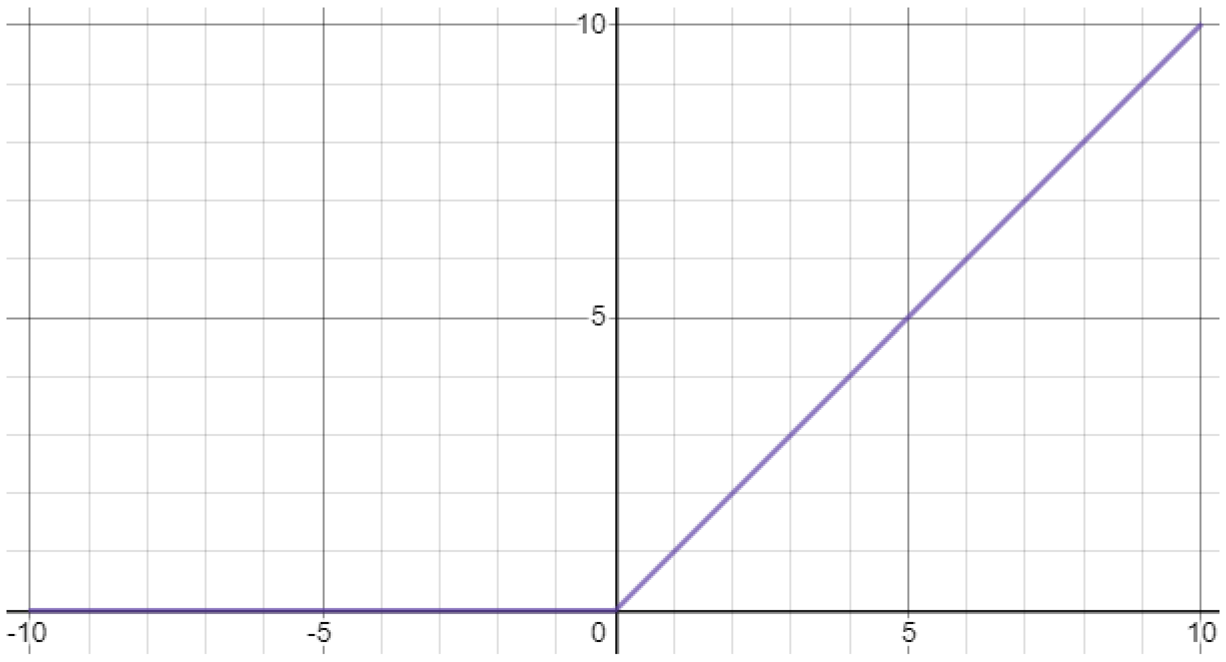


Figure 2.3: ReLU activation function plot

So, we may need to bound the values within a limit before sending it to the next layer. This is a job for activation functions. We describe the activation functions used in our method below-

Rectified Linear Unit (ReLU)

Rectified linear unit [21] is widely used among all other activation functions. It is a non-linear function that tries to reduce the activation by converting the negative inputs to zero and thus it becomes easier to train the data of the model. The ReLU function does not activate all the neurons simultaneously which gives it an advantage over other activation functions. In mathematical form, the function looks like-

$$f(x) = max(0, x)$$

The ReLU equation tells us that:

- If the input x is less than or equal to 0, set output equal to 0
- If the input is greater than 0, set output equal to input

A function plot of this function can be seen in figure 2.3.

Softmax

The softmax function [22] is used to transform the outputs to probability values so that the sum of the outputs is equal to 1. This is usually used with the final output layer in a classification problem. It basically converts each prediction to probability values for each class. The softmax function looks like-

$$f(x) = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}$$

Here x_i is the output value of i-th neuron given $i = 1, 2, 3 \ldots, k$ and the bigger the value of x_i, the greater the probability we will get for that instance. An example of softmax function for our network output [0,1] is shown below-
Here, $denominator = e^0 + e^1 = 1 + 2.71828 = 3.71828$ From table 2.1, we can say that

Chest X-Ray	**x**	e^x	**Probability**
Normal	0	1	0.27
Pneumonia	1	2.71828	0.73

Table 2.1: Probability calculation in softmax function

he network is 73% confident that the X-Ray image is classified under Pneumonia.

2.2.3 Average Pooling Layer

A pooling layer [23] is usually used after a convolutional layer. This layer down-samples the output of a convolutional layer. This helps in reducing the number of parameters to learn and controlling over-fitting. An average pooling operation takes the average value from a specific pixel region while down-sampling. The size of the pixel region is specified and is applied to the input matrix with a specific stride. Figure 2.4 shows an example operation of average pooling layer.

10	6	7	3
5	4	1	2
8	8	12	1
9	1	5	4

6.25	3.25
6.5	5.5

Figure 2.4: The matrix on the left is the input upon which average pooling is being applied. The one on the right is the resultant down-sampled matrix. In this example, the filter dimension is 2×2 and stride is 2. The average value of each colored area is being taken and the output values are also color coded to represent the area from where it was taken

2.2.4 Flatten Layer

Usually, the output from the convolutional layers and max-pooling layers are multi-dimensional. But, in most of the networks, the final few layers are fully connected. The input to the fully connected layers must be one dimensional. So, we use a flatten layer to convert the multi-dimensional data to one-dimensional data by flattening it.

2.2.5 Dropout Layer

Dropout layers are mainly used between fully connected layers to reduce over-fitting. At each back-propagation phase, it does not consider some of the connections of hidden layers (by randomly setting their output to 0) and so it forces the other neurons of the hidden layers to learn about the data patterns in a more generalized manner.

2.2.6 Fully Connected Layer

It is convenient to use fully connected layers in the last few layers of a convolutional neural network. A neuron of a layer like this sends its output to all the neurons of the next layer, hence the name fully connected layer. The convolutional layers provide a feature space and the fully connected layers try to learn a pattern from that feature space. The idea behind the fully connected layer can be seen in figure 2.5.

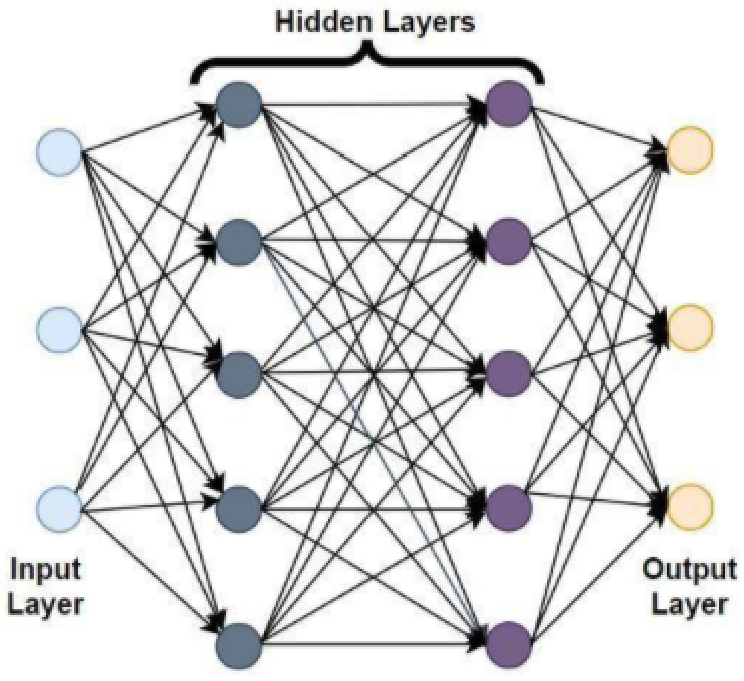


Figure 2.5: Fully Connected Layer

A fully-connected layer is composed where all neurons of the previous layer integrate information of each neuron of that layer. The flattening layer and the first dense layer are the entirely associated layers in our research methodology. Here, the inputs via feature analysis are extracted and weights are performed to determine the feature vectors or classification in the first hidden layers of 1024 neurons.

2.2.7 VGG16

VGG16 is a convolutional neural network model that was proposed in [8]. VGG16 mainly consists of several convolutional layers with ReLU activation function, max-pooling layers, fully connected layers with ReLU activation function and softmax activation function in the final output layer.
The input to this network is $224 \times 224 \times 3$ RGB image. This image is passed through several convolutional layers with kernels of dimension 3×3. The filters in max-pooling layers are of dimension 2×2 and are applied with a stride of 2. The architecture of this network is shown in figure 2.6.

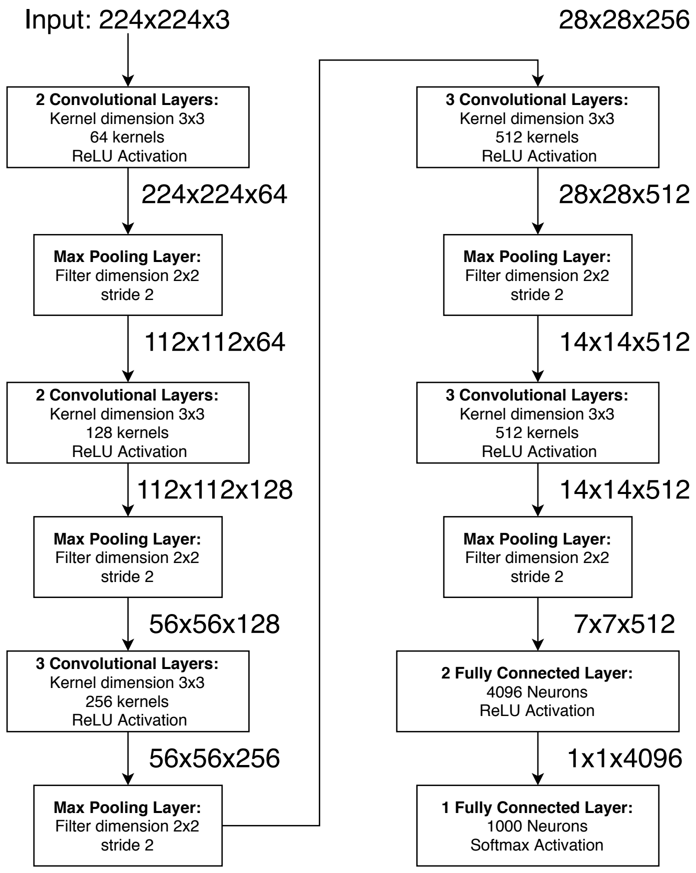


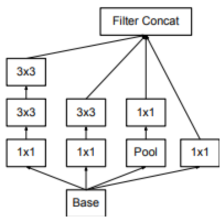
Figure 2.6: The detailed architecture of VGG16 with the number of kernels in convolutional layers and activation functions used in each layer is shown in this figure. The output dimension of each layer is also shown here

2.2.8 Inception-v3

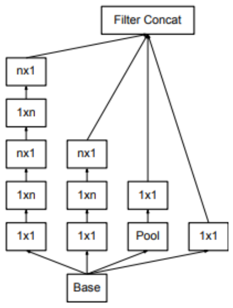
Before inception, most CNN models were just deeply stacked convolutional layers. However, inception is a complex network that leverages a lot of tricks to increase performance. There are several versions of inception that iteratively improves upon the previous one. First two versions of inception was introduced in [7] and [24] called inception-v1 and inception-v2. Inception-v3 was introduced along with inception-v2 in [24]. The inception network uses kernels of multiple dimensions in the same convolutional layer. This makes the network wider rather than deeper, thus reducing the probability of over-fitting. The architecture of this network is shown in figure 2.7.

2.2.9 ResNet-50

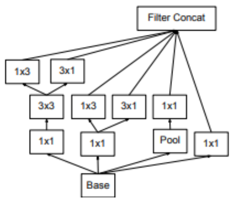
ResNet-50 [11] allows us to train extremely deep neural networks successfully because it eliminates the vanishing gradient problem. In this model, the concept of skip connections was introduced first (concept depicted in figure 2.8). In a model like this, the output of a layer can be passed to a distant layer along with the immediate layer. This ensures that all the layers perform at the same capacity. There are 5 stages in this model, each stage consisting of various combinations of convolution layers. The architecture of this model is shown in figure 2.9.



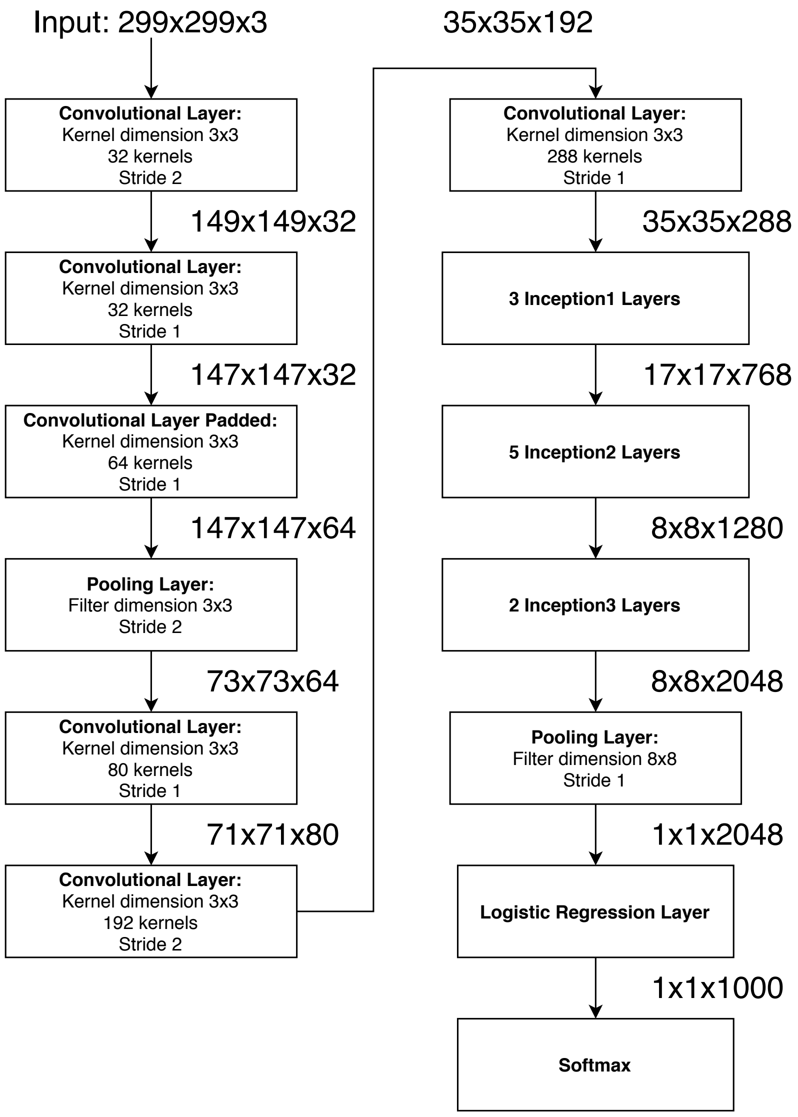
(a) Inception1



(b) Inception2



(c) Inception3


Input: 299x299x3
Convolutional Layer: Kernel dimension 3x3 32 kernels Stride 2
149x149x32
Convolutional Layer: Kernel dimension 3x3 32 kernels Stride 1
147x147x32
Convolutional Layer Padded: Kernel dimension 3x3 64 kernels Stride 1
147x147x64
Pooling Layer: Filter dimension 3x3 Stride 2
73x73x64
Convolutional Layer: Kernel dimension 3x3 80 kernels Stride 1
71x71x80
Convolutional Layer: Kernel dimension 3x3 192 kernels Stride 2
35x35x192
Convolutional Layer: Kernel dimension 3x3 288 kernels Stride 1
35x35x288
3 Inception1 Layers
17x17x768
5 Inception2 Layers
8x8x1280
2 Inception3 Layers
8x8x2048
Pooling Layer: Filter dimension 8x8 Stride 1
1x1x2048
Logistic Regression Layer
1x1x1000
Softmax

(d)

Figure 2.7: The detailed architecture of inception-v3. The figures in a, b, c are taken from [24]. The layers named **Inception1**, **Inception2** and **Inception3** refers to the networks shown in a, b, c

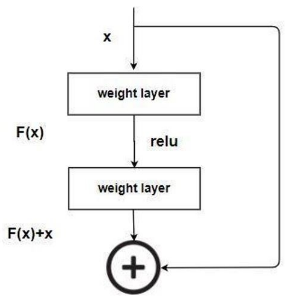
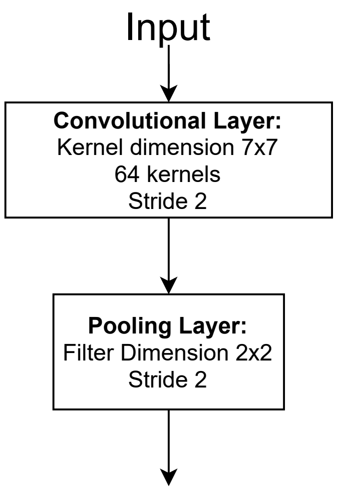
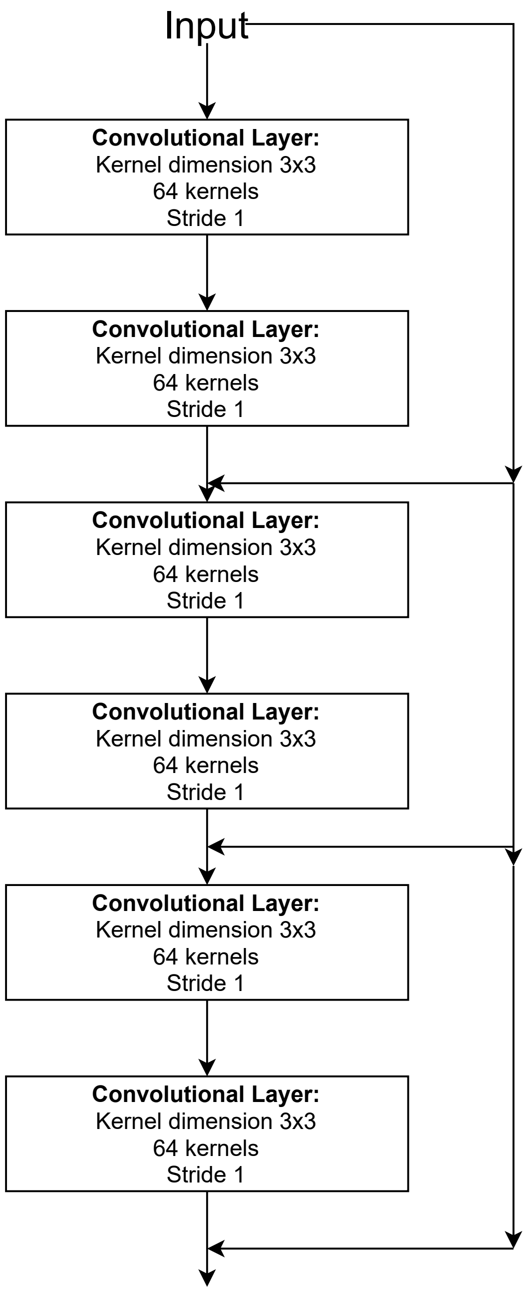


Figure 2.8: Residual Learning

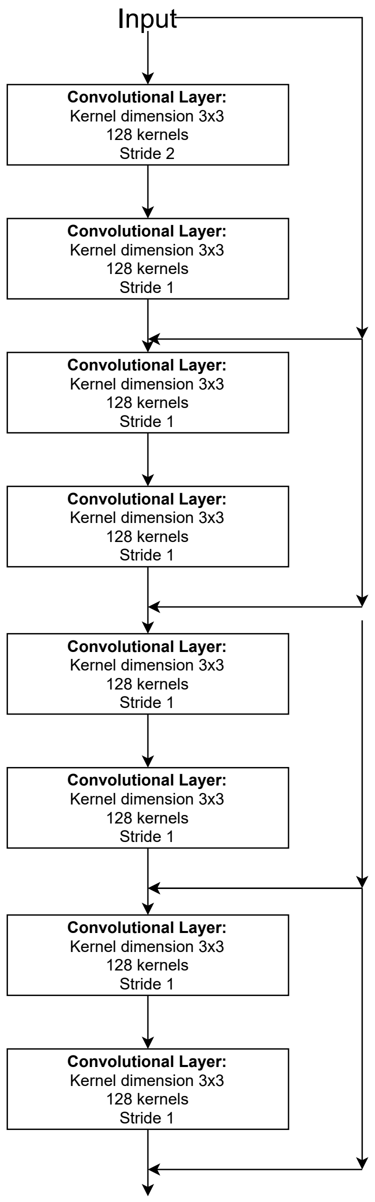


(a) Stage 1

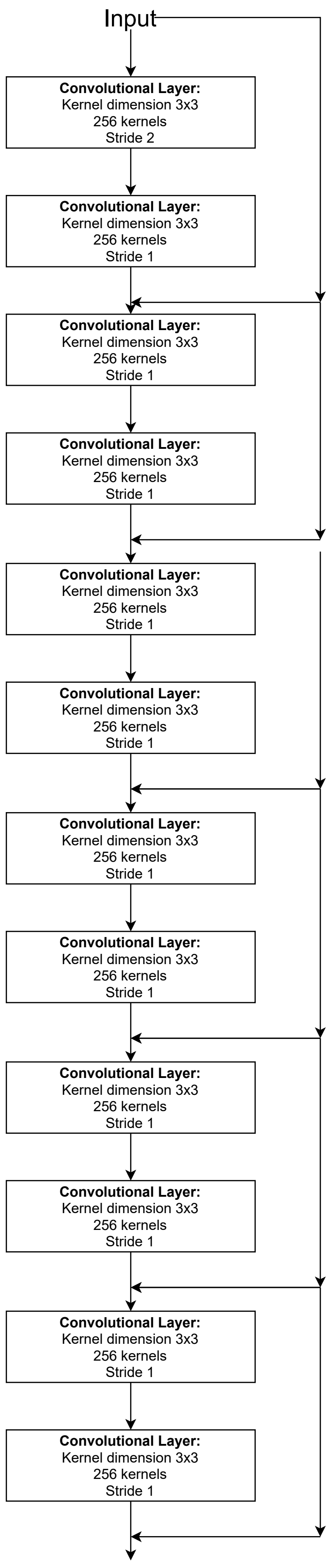


(b) Stage 2

Figure 2.9

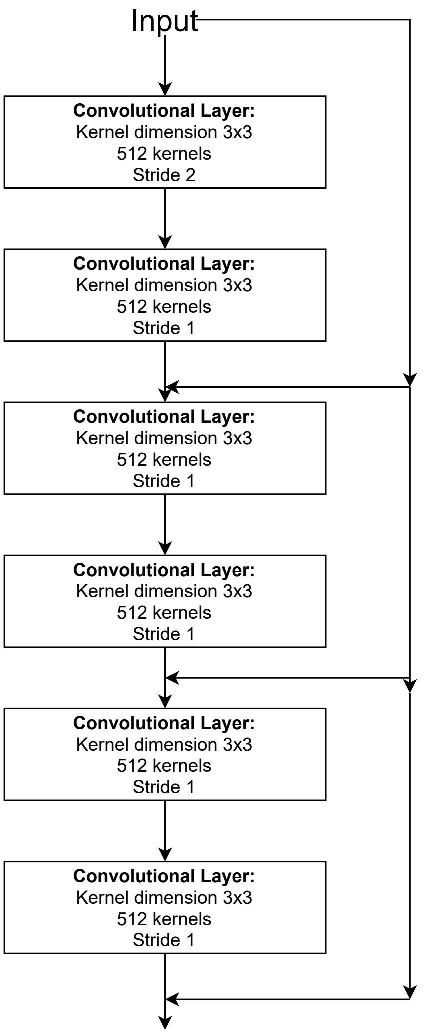


(c) Stage 3

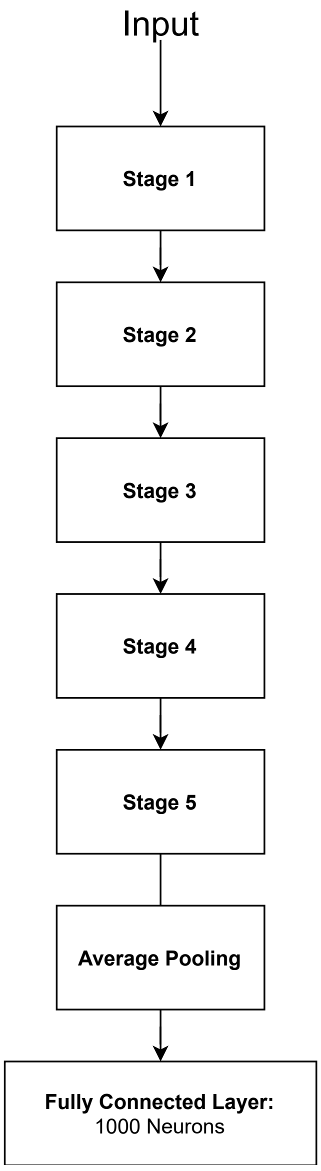


(d) Stage 4

Figure 2.9


Input
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 2
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 1
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 1
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 1
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 1
Convolutional Layer: Kernel dimension 3x3 512 kernels Stride 1

(e) Stage 5



(f) Main Model

Figure 2.9: This figure shows the detailed architecture of ResNet-50 model with each stage of the model shown in sub-figures a, b, c, d, e. Stages 1, 2, 3, 4 and 5 are then combined in sub-figure f as the main model

2.2.10 Sequential Model Representation of Inception-v3, VGG16 and ResNet50

A sequential model representation of the models that we used is shown in figure 2.10. This figure was generated using Keras [25] library.

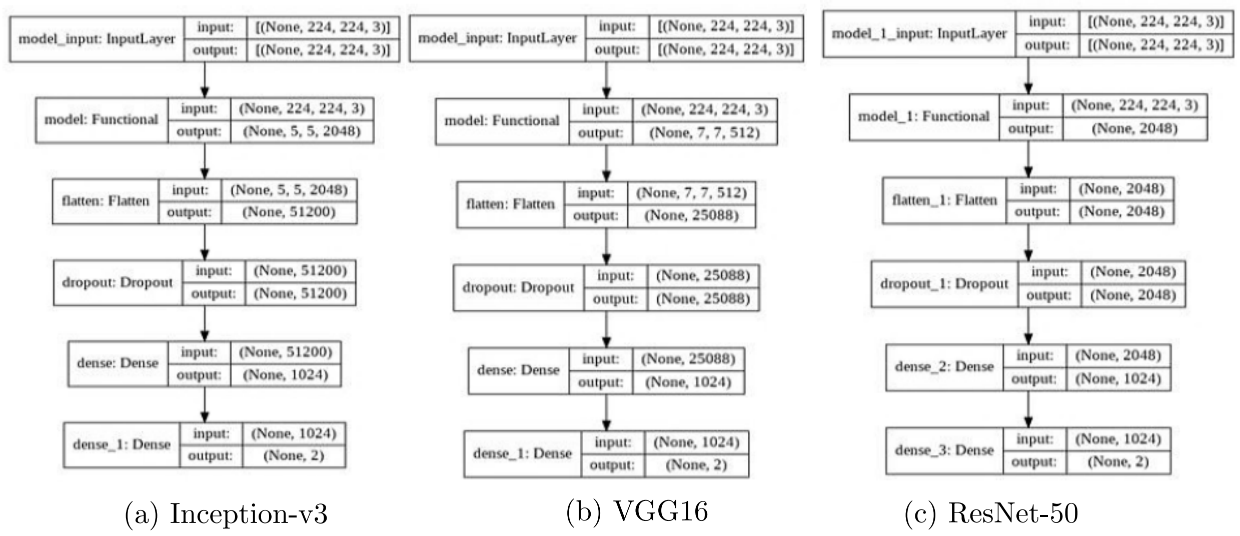

model_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
model: Functional | input: (None, 224, 224, 3) | output: (None, 5, 5, 2048)
flatten: Flatten | input: (None, 5, 5, 2048) | output: (None, 51200)
dropout: Dropout | input: (None, 51200) | output: (None, 51200)
dense: Dense | input: (None, 51200) | output: (None, 1024)
dense_1: Dense | input: (None, 1024) | output: (None, 2)
(a) Inception-v3
model_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
model: Functional | input: (None, 224, 224, 3) | output: (None, 7, 7, 512)
flatten: Flatten | input: (None, 7, 7, 512) | output: (None, 25088)
dropout: Dropout | input: (None, 25088) | output: (None, 25088)
dense: Dense | input: (None, 25088) | output: (None, 1024)
dense_1: Dense | input: (None, 1024) | output: (None, 2)
(b) VGG16
model_1_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
model_1: Functional | input: (None, 224, 224, 3) | output: (None, 2048)
flatten_1: Flatten | input: (None, 2048) | output: (None, 2048)
dropout_1: Dropout | input: (None, 2048) | output: (None, 2048)
dense_2: Dense | input: (None, 2048) | output: (None, 1024)
dense_3: Dense | input: (None, 1024) | output: (None, 2)
(c) ResNet-50

Figure 2.10: Illustration of sequential models

2.2.11 Ensemble Modeling

Ensemble modeling [26], [27] is a hierarchical process of taking a collection of multiple diversified models for making predictions. Here, the inputs for each model stay the same in shape and also the output shape stays the same to get adjusted in the averaging layer. Most importantly, this type of modeling reduces error or false negative and false positive predictions. However, there is a term and condition to set up the ensemble mode and that is, the models that are being used to get ensembled have to be different in architecture independently. The concept of ensembling is illustrated in figure 2.11.

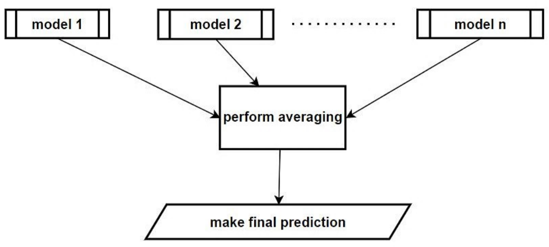


Figure 2.11: Basic structure of an ensemble model

In a nutshell, an ensemble model is a single model that comprises multiple models and predicts the instance with better feature analysis and better accuracy.

Averaging or Average Layer

In this layer, we have taken the output from the three models as the input and the output of this layer is the average of those three inputs. From the average output probability, our model has predicted two classes or labels.

Output Layer

In this layer, the "ENSEMBLE BETA" finally predicts the relative probability of the two predefined classes or labels or neurons. As a result, we can predict the actual class of that input X-Ray at the very early step of our architecture by comparing the two predicted probability based on feature analysis and averaging.

Let's say, for example, if the output layer provides the following NumPy array:

$$prediction = [0.33, 0.67]$$

We can see that the probability at index 0 is less than the probability at index 1 of the prediction matrix. So, the predicted class will be Pneumonia or labeled as 1. An illustration of this layer is depicted in figure 2.12 (Figure generated using Keras).

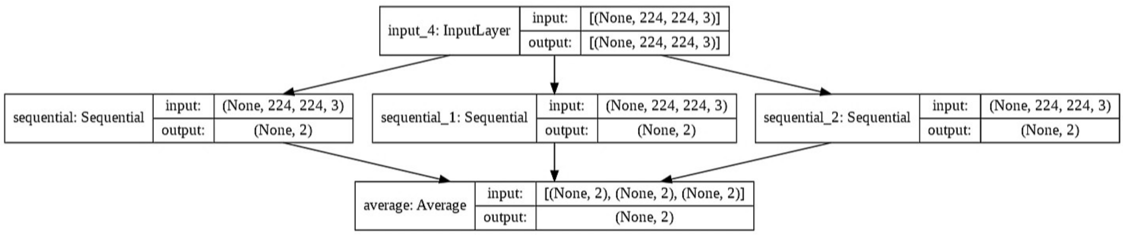

input_4: InputLayer
input: [(None, 224, 224, 3)]
output: [(None, 224, 224, 3)]
sequential: Sequential
input: (None, 224, 224, 3)
output: (None, 2)
sequential_1: Sequential
input: (None, 224, 224, 3)
output: (None, 2)
sequential_2: Sequential
input: (None, 224, 224, 3)
output: (None, 2)
average: Average
input: [(None, 2), (None, 2), (None, 2)]
output: (None, 2)

Figure 2.12: Illustration of the output layers of the ensemble model hierarchy

2.3 Workflow Diagram

The working diagram (figure 2.13) is an overview of the distinct stages we have taken to train and validate our prototype. We separated our flow of work into three fundamental segments, named after the phase of transfer learning, the phase of training-cross-validation and the phase of testing.

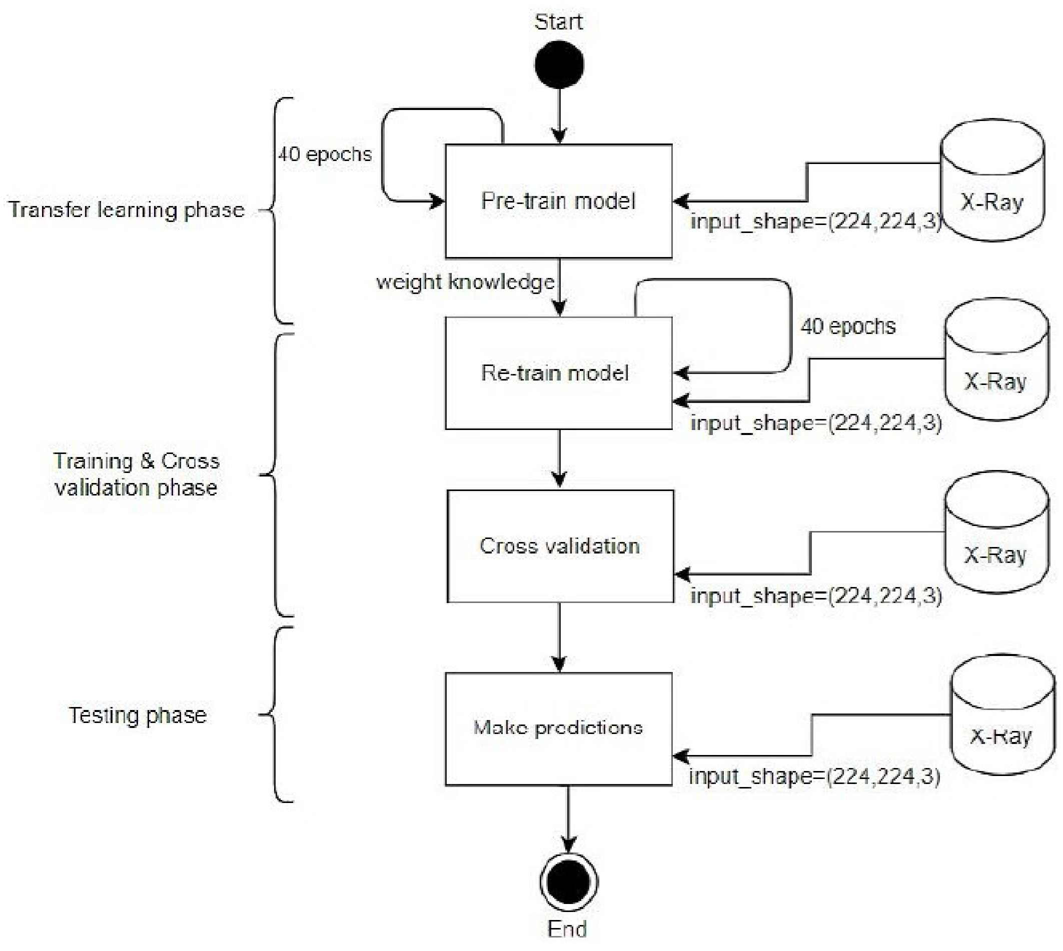


Figure 2.13: Workflow diagram for our model training and evaluation

Here, in each phase, we have used a different subset of chest X-Ray images as input and have performed particular actions. In our first phase of transfer learning, we have built an ensemble model and trained it for 40 epochs and we have also settled weight parameters as "imagenet". Secondly, we have re-trained our pre-trained model and also checked the validation of our model's training accuracy, whether it is under-fitting or over-fitting. Lastly, we have done predictions on an unknown label subset of chest X-Ray images to our model and evaluated it on behalf of the predicted label.

2.3.1 Transfer Learning Phase

Transfer learning (TL) is a concept of machine learning (ML), where the model gets trained with an initial weight knowledge and then the learned weights are applied to solve the relatable problem [28].

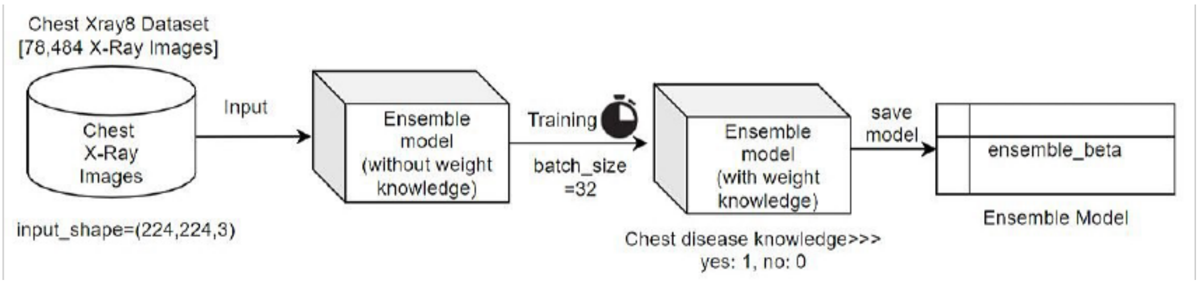


Figure 2.14: Transfer learning phase

First of all, we have trained our ensemble model with the Chest-Xray8 dataset and let the model learn about two classes, one is a normal patient class and another one is an abnormal or diseased patient class. Afterwards, we have saved the model with the gained knowledge of weights and have named it "ENSEMBLE BETA". The idea is shown in figure 2.14.

2.3.2 Training and Cross-Validation Phase

Secondly, we have loaded our "ENSEMBLE BETA" model and re-trained it with a Pneumonia dataset, which is known as fine-tuning [28] and along with that, we have also cross-checked the validation accuracy as well as validation loss, whether the accuracy is increasing or not and the loss is decreasing or not. Later on, depending on our analysis, we have generated the graphs of validation accuracy and validation loss that have been discussed in chapter 4 of result and discussion. This phase is depicted in figure 2.15.

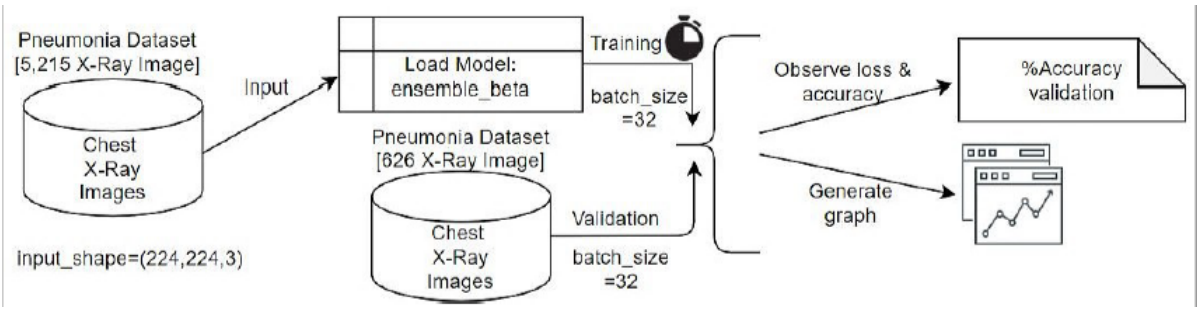


Figure 2.15: Training and cross validation phase

2.3.3 Testing Phase

This is our final step and in this step we have used 16 X-Ray images of the chest that are totally unknown to our model and have predicted their label. Table 2.2 shows the result of the predictions. Figure 2.16 shows the concept behind this phase.

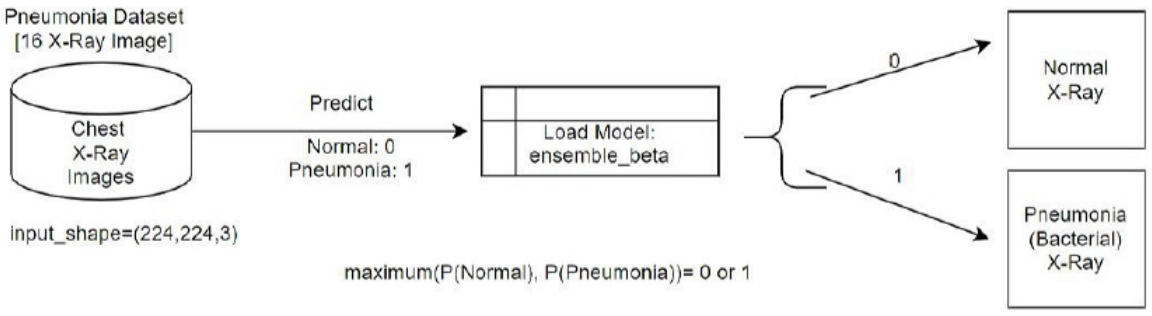


Figure 2.16: Testing phase

Actual Image Names	Predicted by "ENSEMBLE_BETA"	Category
NORMAL2-IM-1427-0001	0	Normal
NORMAL2-IM-1430-0001	0	Normal
NORMAL2-IM-1431-0001	0	Normal
NORMAL2-IM-1436-0001	0	Normal
NORMAL2-IM-1437-0001	1	Pneumonia
NORMAL2-IM-1438-0001	0	Normal
NORMAL2-IM-1440-0001	0	Normal
NORMAL2-IM-1442-0001	0	Normal
person1946_bacteria_4874	0	Normal
person1946_bacteria_4875	1	Pneumonia
person1947_bacteria_4876	1	Pneumonia
person1949_bacteria_4880	1	Pneumonia
person1950_bacteria_4881	1	Pneumonia
person1951_bacteria_4882	1	Pneumonia
person1952_bacteria_4883	1	Pneumonia
person1954_bacteria_4886	1	Pneumonia

Table 2.2: Predicted output label of the 16-sample input chest X-Ray images

From table 2.2, we can see that our model has classified the input images into either 0 or 1, naming Normal and Pneumonia respectively.

Chapter 3

Implementation

3.1 Source of Datasets

- **ChestX-Ray8 dataset:** Constructed in [10]
- **Pneumonia dataset:** Kaggle [29]

3.2 Data Samples

Sample X-Ray images are shown in figure 3.1.

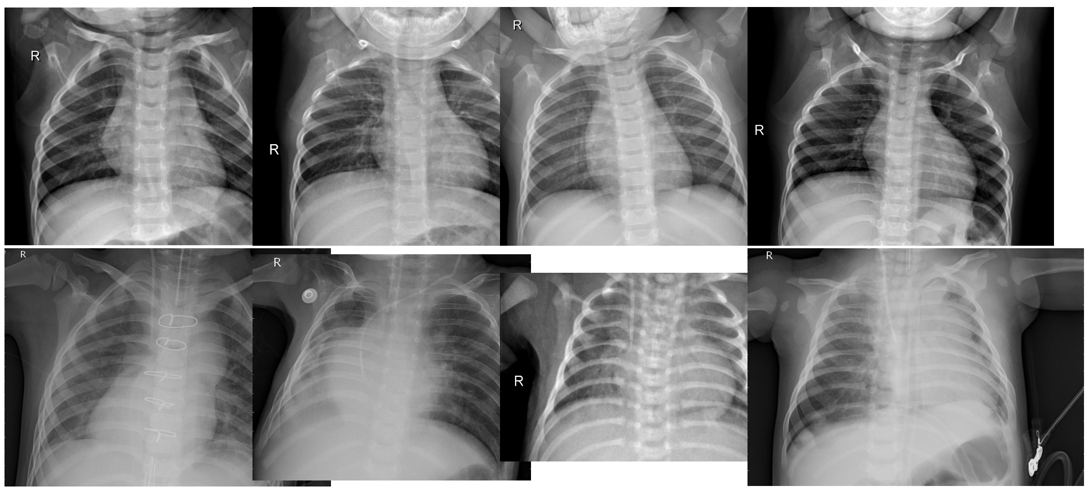


Figure 3.1: The first row of images are X-Ray images of patients whose lungs are in normal condition (not pneumonia affected). The second row of images are the X-Ray images of lungs of pneumonia patients

3.3 Data Visualizations

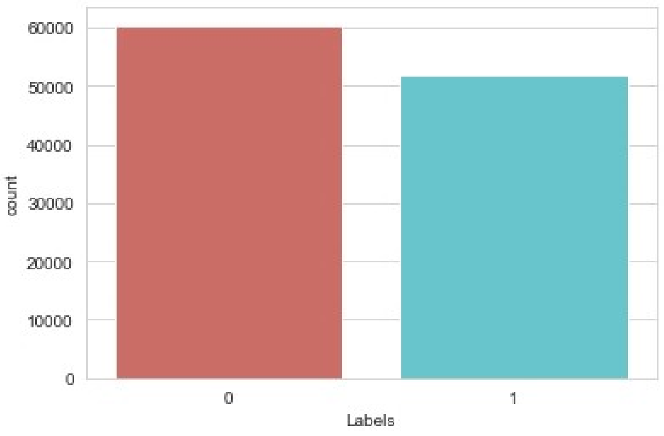


Figure 3.2: Illustration of data balance using bar chart of “Labels(Normal, Abnormal)” column of Chest X-Ray8 dataset

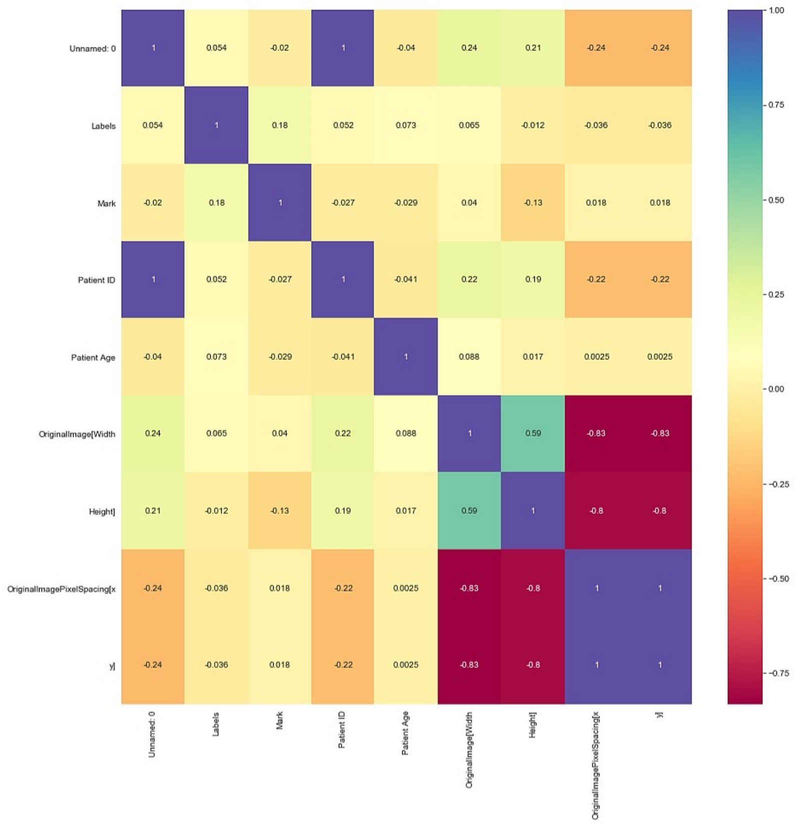


Figure 3.3: Illustration of correlation mapping of different attributes or features with the label or class of Chest X-Ray8 dataset

Using some standard histograms and correlation mapping using numpy and seaborn libraries, we have tried to visualize the dataset of chest-xray8. In parallel, we could see the balance of that same population in various attributes by analyzing the histograms. Last and not least, we have noted the attributes that would be most closely correlated.

3.4 Dataset Classification

3.4.1 Training Set

The phase during which classified example data with the results or feedback labels are given to the deep supervised learning framework.

3.4.2 Validation set

The set of instances used and quite often referred to as the learning set across training.

3.4.3 Testing Set

In certain situations, for "real-world" testing, a final group of instances is used for the algorithm model iterates to improve significantly with the validation set, it may learn key features of the training set. With an "unseen" test collection, good output increases the assurance that the algorithm would provide corrective feedback in the physical world.

3.5 Data Pre-processing

3.5.1 Resize Images

The prime objective of our transfer learning process is to make a diagnosis of pneumonia disease from X-Ray images with the most positive results. For this reason, as shown, we have grouped some prototypes and independently trained them. The dataset accommodates X-Ray images from patients where some of the patients are diagnosed with pneumonia and some patients are not. Since we will be using a pre-trained ResNet-50 model, while training the network, each image will be re-sized into a fixed size of 224×224. For this purpose, we will be using Scikit Image [30], TensorFlow [31], and Caffe frameworks [32]. The ImageDataGenerator [33], [34] class in Keras is used for scaling pixel values in our image dataset before modeling. This class combines our image dataset during formation, validation, or assessment, then, while requested, it will restore images to the algorithm through batches and apply the scaling operations as required. When modeling with neural networks, this provides a powerful and reasonable methodology for scaling image data. The ImageDataGenerator handles the percentage of pixels scaling methods as well as a variety of different feature selection methods. A reference to leveling is allowed by the ImageDataGenerator class which mostly operates the mean calculated on the training dataset as feature-wise centering. Statistics calibrated before regression on the training sample are needed here.

3.5.2 Normalization and Scaling Images

Principal component analysis (PCA) [35], [36] is used to compute the eigen flat fields of a set of flat fields. Each X-Ray projection is then normalized using a linear combination of the most important eigen flat fields. From experiments, we know that the proposed dynamic flat field correction results in a significant reduction of systematic errors in projection intensity normalization in comparison with the conventional flat field correction. For this purpose, we will be using the ImageDataGenerator class of Keras. Scaling information to the extent of 0-1 is known as normalization. We can get this by constructing the re-scale argument to a ratio by which each pixel can be multiplied to attain the wanted range.

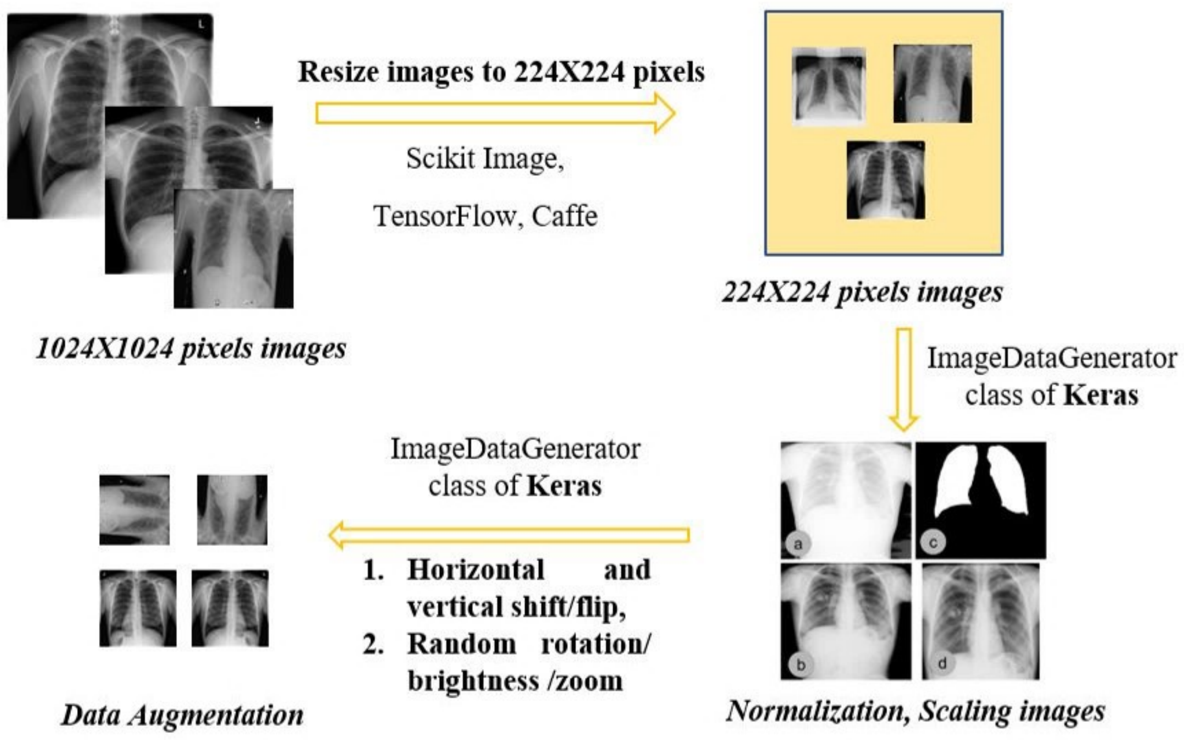

Resize images to 224X224 pixels
Scikit Image,
TensorFlow, Caffe
1024X1024 pixels images
224X224 pixels images
ImageDataGenerator class of Keras
ImageDataGenerator class of Keras
1. Horizontal and vertical shift/flip,
2. Random rotation/ brightness /zoom
Data Augmentation
Normalization, Scaling images
a
b
c
d

Figure 3.4: Pre-processing phases of the dataset

3.5.3 Data Augmentation

We augment our data by applying a set of random transformations to the images for increasing our model performance. We apply rotation (90°, 180°and 270°) and translation to the images and also horizontally and vertically flip the images. For this purpose, the ImageDataGenerator class of Keras will be used.
The Keras [25] deep learning neural network library can fit models using image data augmentation via the ImageDataGenerator class. Image data augmentation is needed to enlarge the training dataset to upgrade the achievement and capability of the model to generalize. The dataset images are not used directly. As an alternative, only augmented images are supplied to the model. Random augmentation of

images allows both modified and near copy of the images to be produced and used for training. The validation dataset and the test dataset are specified by a Data Generator. Most of the time, a separate ImageDataGenerator instance is used that may have the same pixel scaling configuration as the ImageDataGenerator instance used for the training dataset, but not data augmentation. The reason is data augmentation is only used as a technique for artificially enlarging the training dataset in contemplation of making better model performance on an unsegmented dataset. An image shifting means moving all pixels of the image in one direction, it can be both horizontally or vertically, but the dimensions of the images are the same. The width_shift_range and height_shift_range arguments to the ImageDataGenerator constructor direct the amount of horizontal and vertical shift sequentially.

Chapter 4

Results and Discussion

4.1 Individual Model

Dataset	Keras model	val_categorical_accuracy	val_loss
Pneumonia	Inception-v3	76.69%	1.9598
	VGG16	82.99%	0.9277
	VGG19	73.74%	1.5533
	ResNet50	81.03%	0.5689
	ResNet101	65.62%	1.6825

Table 4.1: Validation accuracy and loss in the experimented individual model

4.2 Before Transfer Learning into "ENSEMBLE BETA"

Dataset	Phase of the model	val_categorical_accuracy	val_loss
Pneumonia	Ensemble model without weight knowledge	78.45%	1.5383

Table 4.2: Validation accuracy and loss before transfer learning into the ensemble model after 40 epochs

4.3 After Transfer Learning into "ENSEMBLE BETA"

Dataset	**Phase of the model**	**val_categorical_accuracy**	**val_loss**
ChestX-Ray8	Ensemble model without weight knowledge	61.05%	1.234
Pneumonia	Ensemble model with weight knowledge	88.14%	0.5033

Table 4.3: Validation accuracy and loss after transfer learning into the ensemble model after 40 epochs

4.4 Learning Curve

In machine learning, learning curves are most often used and are frequently a plot that exhibits iterations or time or history on the x-axis and the consistency of learning or classification on the y-axis. It helps to test as well as evaluate the model at the beginning of training. We included a tqdm callback to see the percentage of training and minimize the progress callback to decrease the number of learning when another ratio halted progressing with a patience value of 3. For 40 periods of history or oscillations, we have trained and validated our model and established the following prediction performance and validation loss graph.

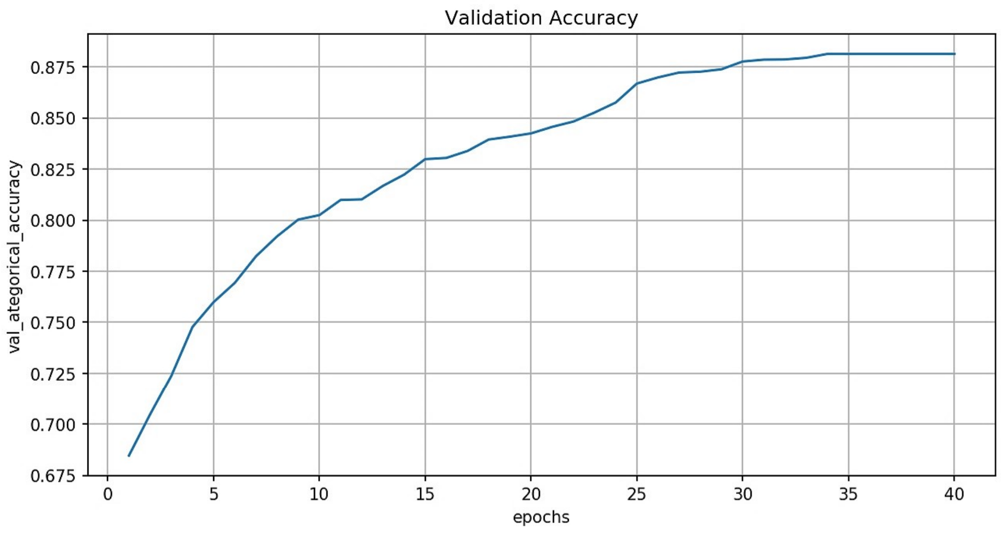


Figure 4.1: Validation accuracy curve of Ensemble Beta

From figure 4.1, we can observe that the validation accuracy has an increasing slope and it stops increasing and becomes constant at the very end of training and gets fixed with an accuracy of 88.14%.

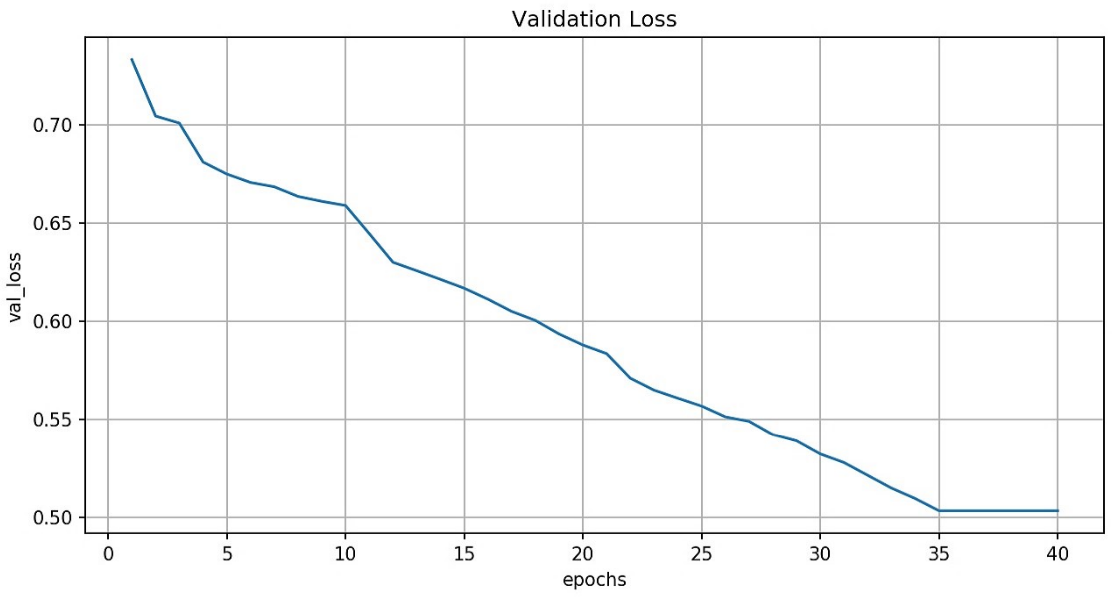


Figure 4.2: Validation loss curve of Ensemble Beta

On the other hand, from the figure 4.2, we can observe that the validation loss is decreasing having a negative slope and it illustrates the efficiency of our model ENSEMBLE BETA.

4.5 Comparison with Other Models

We have compared our model with the other existing models that we have covered during over literature reviewing and have listed the accuracy in table 4.4.

Model	**val_categorical_accuracy**
CheXNet [37]	76.80%
CNN with Lightened Image on Increased Contrast	75.65%
CNN with Lightened Image on Increased Contrast with ResNet	78.73%
Our model (Ensemble_Beta)	88.14%

Table 4.4: Comparison of others model versus our model

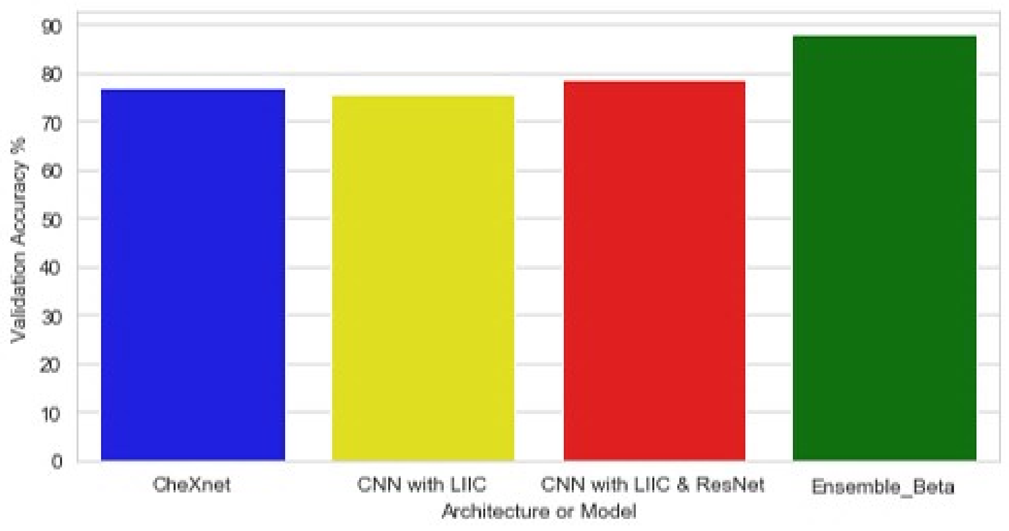


Figure 4.3: Illustration of different models versus our model using a bar chart

Since it has two positive aspects, our ensemble model with transfer learning is efficient. First of all, ensemble model gives higher accuracy as, it reduces the over-fitting and during our model testing we have seen an increasing sloped curve for validation accuracy and decreasing sloped curve for validation loss. Secondly, ensemble technique helps us to reduce the bias and variance error by maintaining a trade-off in between these two parameters and this helps to learn less noisy data while training.As a result higher precision was obtained comparing with other existing models.

Chapter 5

Conclusion and Future work

We analyzed the existing CNN lung disease classification techniques, their correlations and also stated in-depth our planned ”ENSEMBLE BETA” structure in terms of its proposed architecture, transfer learning stage, classification model, implementation stage, predictive validity and effectiveness in this research paper. Furthermore, this research has future demands, as now we are going through Corona virus pandemic and now the lung diseases are getting prioritized to be detected in the early stage. As a consequence, health professionals will be capable of classifying the normal against the disordered X-Ray and would have reasonable steps on time. Conversely, we encountered some challenges even during the implementation phase when training the dataset with a sufficiently smaller like 16 or greater like $64, 128, \ldots, k$ as batch size during the implementation phase. Moreover, we might use GoogleNet, AlexNet and other powerful machine learning models to ensemble and check whether the experiment gives better performance or not and we will use a better configuration PC set up with a good amount of GPU. In conclusion, this research project can be strengthened by classifying different lung diseases consequently in the future, rather than just predicting pneumonia from standard X-Ray images.

Bibliography

[1] W. H. Organization *et al.*, "Influenza. who fact sheet no 211," *http://www. who. int/mediacentre/factsheets/2003/fs211/en/*, 2003.

[2] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[3] *Coronavirus and pneumonia*, https://www.webmd.com/lung/covid-and-pneumonia/.

[4] L. Yao, E. Poblenz, D. Dagunts, B. Covington, D. Bernard, and K. Lyman, "Learning to diagnose from scratch by exploiting dependencies among labels," *arXiv preprint arXiv:1710.10501*, 2017.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] H. Wang and Y. Xia, "Chestnet: A deep neural network for classification of thoracic diseases on chest radiography," *arXiv preprint arXiv:1807.03058*, 2018.

[10] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2097–2106.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao, and R. M. Summers, “Interleaved text/image deep mining on a very large-scale radiology database,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1090–1099.

[13] R. H. Abiyev and M. K. S. Ma’aitah, “Deep convolutional neural networks for chest diseases detection,” *Journal of healthcare engineering*, vol. 2018, 2018.

[14] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao, and R. M. Summers, “Interleaved text/image deep mining on a large-scale radiology database for automated image interpretation,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3729–3759, 2016.

[15] P. Rajpurkar, J. Irvin, R. L. Ball, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. P. Langlotz, *et al.*, “Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists,” *PLoS medicine*, vol. 15, no. 11, e1002686, 2018.

[16] O. Er, N. Yumusak, and F. Temurtas, “Chest diseases diagnosis using artificial neural networks,” *Expert Systems with Applications*, vol. 37, no. 12, pp. 7648–7655, 2010.

[17] S. Khobragade, A. Tiwari, C. Patil, and V. Narke, “Automatic detection of major lung diseases using chest radiographs and classification by feed-forward artificial neural network,” in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, IEEE, 2016, pp. 1–5.

[18] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, “Medical image analysis using convolutional neural networks: A review,” *Journal of medical systems*, vol. 42, no. 11, p. 226, 2018.

[19] M. I. Razzak, S. Naz, and A. Zaib, “Deep learning for medical image processing: Overview, challenges and the future,” in *Classification in BioApps*, Springer, 2018, pp. 323–350.

[20] S. Sharma, “Activation functions in neural networks,” *Towards Data Science*, vol. 6, 2017.

[21] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding deep neural networks with rectified linear units,” *arXiv preprint arXiv:1611.01491*, 2016.

[22] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, “A high-speed and low-complexity architecture for softmax function in deep learning,” in *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, IEEE, 2018, pp. 223–226.

[23] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, IEEE, 2017, pp. 1–6.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[25] F. Chollet *et al.* (2015). “Keras,” [Online]. Available: https://github.com/fchollet/keras.

[26] T. G. Dietterich *et al.*, “Ensemble learning,” *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[27] R. Polikar, “Ensemble learning,” in *Ensemble machine learning*, Springer, 2012, pp. 1–34.

[28] M. Kurtz, *How transfer learning can make machine learning more efficient*, https://thenewstack.io/how-transfer-learning-can-make-machine-learning-more-efficient/.

[29] P. Mooney, *Chest x-ray images (pneumonia)*, https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.

[33] A. Gulli and S. Pal, *Deep learning with Keras.* Packt Publishing Ltd, 2017.

[34] N. K. Manaswi, “Understanding and working with keras,” in *Deep Learning with Applications Using Python*, Springer, 2018, pp. 31–43.

[35] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[36] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[37] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, *et al.*, “Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning,” *arXiv preprint arXiv:1711.05225*, 2017.