SYSTEM SPECIFICATIONS

LITERATURE SURVEY

ANALYSIS

DESIGN

IMPLEMENTATION AND RESULTS ANALYSIS

TESTING AND VALIDATIONS

CONCLUSION AND FUTURE ENHANCEMENTS

REFERENCES

INTRODUCTION

1. INTRODUCTION

The project titled “Chest X-ray based Disease Prediction System using CNN” is a system used to predict diseases from the Chest X-rays of the patients.

1.1 INTRODUCTION

There are many diseases that can be predicted from the Chest X-rays of human beings. These diseases include Emphysema, Pneumonia, Fibrosis, Cardiomegaly etc. The timely diagnosis of these diseases is very important. If these diseases are not predicted in time, then the severity of these diseases will increase and it will endanger the patient.

This project can be useful for the prediction of diseases from the Chest X-rays of the patients in just a few clicks. It is a very efficient and accurate way to get to know about the diagnosis of the diseases without any human intervention. It is very reliable and free of cost. It can predict up to 14 types of diseases from the Chest X-rays of the patients.

1.2 MOTIVATION

The motivation for doing this project is primarily an interest in undertaking a challenging project in an interesting area of research. The opportunity to learn about a new area of computing not covered in lectures is appealing. With the excessive growth in the need for the diagnosis of diseases, Chest X-ray based Disease Prediction System using CNN will be useful in the diagnosis of some of the diseases.

1.3 PROBLEM DEFINITION

Dispredo (Disease Predictor) is a website which is used in order to view the disease predictions by uploading the Chest X-ray image of the patient. This website is an efficient way to get to know about the diagnosis of the diseases. This website is fast and reliable in it’s own way. The good thing about this website is that it a cost free solution and gives the predictions in a matter of seconds without any human intervention. It can predict up to 14 types of diseases from the Chest X-rays of the patients.

1.4 OBJECTIVE OF THE PROJECT

The main objective of this project is to develop a system that can predict 14 types of diseases from the Chest X-rays of the patients without the need of any human intervention. The diseases that can be predicted in the project are:

- Atelectasis
- Consolidation
- Infiltration
- Pneumothorax
- Edema
- Emphysema
- Fibrosis
- Effusion
- Pneumonia
- Pleura Thickening
- Cardiomegaly
- Nodule
- Mass
- Hernia

1.5 LIMITATIONS OF THE PROJECT

- Can only predict 14 types of diseases from the Chest X-rays
- Cannot predict diseases from X-rays of body parts other than the Chest

1.6 ORGANIZATION OF THE REPORT

The first chapter deals with the introduction of the Disease Prediction project, motivation for developing this project, problem definition, objective of the project. The second chapter deals with the system specifications required for developing the project. It includes hardware and software specifications. The third chapter gives you the preview about literature survey which includes the information about existing system, disadvantages of existing system

and proposed system. The fourth chapter is for analysis of the project which includes data and requirement analysis, module organization. The fifth chapter deals with the design of the project which includes data flow diagrams and UML diagrams. The sixth chapter deals with the implementation of the project which includes module definition and result analysis of the project. The seventh chapter tells about the testing and validations. Finally eighth chapter deals with the conclusion and future enhancement.

2. SYSTEM SPECIFICATIONS

2.1 SOFTWARE SPECIFICATIONS

- Operating System: Windows 10
- Browser: Google Chrome
- Languages: Python 3.7, HTML
- Libraries/Modules: TensorFlow, OpenCV, Flask

2.2 HARDWARE SPECIFICATIONS

- Processor: i5 2.5GHz
- RAM: 4GB and above
- Hard Disk: 80GB and above

3. LITERATURE SURVEY

3.1 INTRODUCTION

The following research papers have been studied for the purpose of this project:

- Intelligent Pneumonia Identification from Chest X-Rays by WASIF KHAN , NAZAR ZAKI , and LUQMAN ALI
- Diagnosis of Pneumonia from Chest X-Ray Images using Deep Learning by Enes AYAN, Halil Murat ÜNVER
- CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning by Pranav Rajpurkar, 1 Jeremy Irvin
- CXNet-m1: Anomaly Detection on Chest X-Rays With Image-Based Deep Learning by SHUAIJING XU, HAO WU, AND RONGFANG BIE
- Diagnosis of Chest Diseases in X-Ray images using Deep Convolutional Neural Network by Arjun Choudhary, Abhishek Hazra, Prakash Choudhary

The key findings obtained from these papers are:

- Understanding the usage of Convolutional Neural Networks (CNN) for the purpose of prediction and usage of different types of layers like Convolutional Layer, Pooling Layer, Activation Layer, Dropout Layer and the Classifier for the creation of the model.
- Understanding the usage of data augmentation technique and dropout method in order to prevent the overfitting of the model.
- Understanding the usage of Activation functions like ReLU and sigmoid in the different layers of the model.
- Understanding the usage of sin-loss function which achieves better accuracy rate, recall rate, F1-score, and AUC value when compared to other loss functions.
- Understanding the usage of Kaggle Library for the purpose of creation and training of the Neural Networks.

3.2 EXISTING SYSTEM

The existing system uses Convolutional Neural Network (CNN) for the purpose of the prediction of the diseases and uses Multi-label classification method. The existing system takes the image as input and gives the predictions of the diseases as the output.

3.3 DISADVANTAGES OF THE EXISTING SYSTEM

The disadvantages of the existing system are:

- The accuracy of the existing system is low
- The model of the existing system has a lot of parameters and is huge in size and is not suitable for integrating into websites
- The existing system does not have a proper user interface

3.4 PROPOSED SYSTEM

The main intention of the proposed system is to design and develop a website which can be used for the diagnosis of various diseases from the Chest X-rays of the patients. The proposed system uses MobileNetV1 architecture. MobileNet is an efficient and portable CNN architecture that is used in real world applications. MobileNets primarily use depthwise separable convolutions in place of the standard convolutions used in earlier architectures to build lighter models. MobileNets introduce two new global hyperparameters(width multiplier and resolution multiplier) that allow model developers to trade off latency or accuracy for speed and low size depending on their requirements.

The models are integrated into a website which will provide the users with a user-friendly interface.

3.5 ADVANTAGES OF PROPOSED SYSTEM

The advantages of the proposed system are:

- High Accuracy
- No human intervention required for the prediction of the diseases
- Free of cost
- Fast

4. ANALYSIS

4.1 INTRODUCTION

An extensive analysis has been done to identify the best architecture to be used for the standard Convolutional Neural Network (CNN). There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet
- MobileNet

After the careful analysis, the MobileNetV1 architecture was chosen for the project. MobileNet is an efficient and portable CNN architecture that is used in real world applications. MobileNets primarily use depthwise separable convolutions in place of the standard convolutions used in earlier architectures to build lighter models. MobileNets introduce two new global hyperparameters(width multiplier and resolution multiplier) that allow model developers to trade off latency or accuracy for speed and low size depending on their requirements.

MobileNets are built on depthwise seperable convolution layers. Each depthwise seperable convolution layer consists of a depthwise convolution and a pointwise convolution. Counting depthwise and pointwise convolutions as separate layers, a MobileNet has 28 layers. A standard MobileNet has 4.2 million parameters which can be further reduced by tuning the width multiplier hyperparameter appropriately.

4.2 MODULE ORGANIZATION

There are two main modules in the project. The first one is the Disease Prediction module and the second one is the User Interface module.

The User Interface module consists of two main buttons. The first button is used for choosing the image to be uploaded by the user. The second button is used to submit the image uploaded by the user. The User Interface module also consists of Image Displayer which displays the image uploaded by the user. There is a heading called DIAGNOSIS under which the disease predictions from the image will be displayed.

In the Disease Prediction module, we first load all the 14 disease prediction models. After that, when the image is submitted by the user, it is received by this module. After receiving the image, predictions are performed on the image with the help of all the 14 types of models. And the predictions will be sent to the User Interface module.

4.3 FEASIBILITY STUDY

4.3.1 Financial Feasibility

As the project is a website, because of this, there will be an associated hosting cost. The bandwidth required generally is very low. The system will follow the freeware software standards. No cost will be charged from the users. From these, it is clear that the project is financially feasible.

4.3.2 Technical Feasibility

The project is a website. And CNN models have been used for prediction of the diseases. The main tools and technologies that are associated with the project are:

- HTML
- CSS
- Python
- Tensorflow
- Flask

Each of these tools and technologies are freely available and the technical skills required are manageable. From this, we know that the project is technically feasible.

4.3.3 Resource Feasibility

The Resources required for the completion of the project are:

- Programming Device (Laptop)
- Programming Tools (Freely available)
- Programming Individuals (available)

So, it is clear that the project is feasible as far as resources are concerned.

5. DESIGN

5.1 INTRODUCTION

The most creative and challenging face of the system development is System Design. It provides the understanding and procedural details necessary for the logical and physical stages of development. In designing a new system, the system analyst must have a clear understanding of the objectives, which the design is aiming to fulfill. The first step is to determine how the output is to be produced and in what format. Second, input data and master files have to be designed to meet the requirements of the proposed output. The operational phases are handled through program construction and testing.

5.2 UML DIAGRAMS

UML stands for Unified Modelling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices. Improved integration between structural models like class diagrams and behaviour models like activity diagrams. Added the ability to define a hierarchy and decompose a software system into components and sub-components. The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

5.2.1 Use Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e., use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

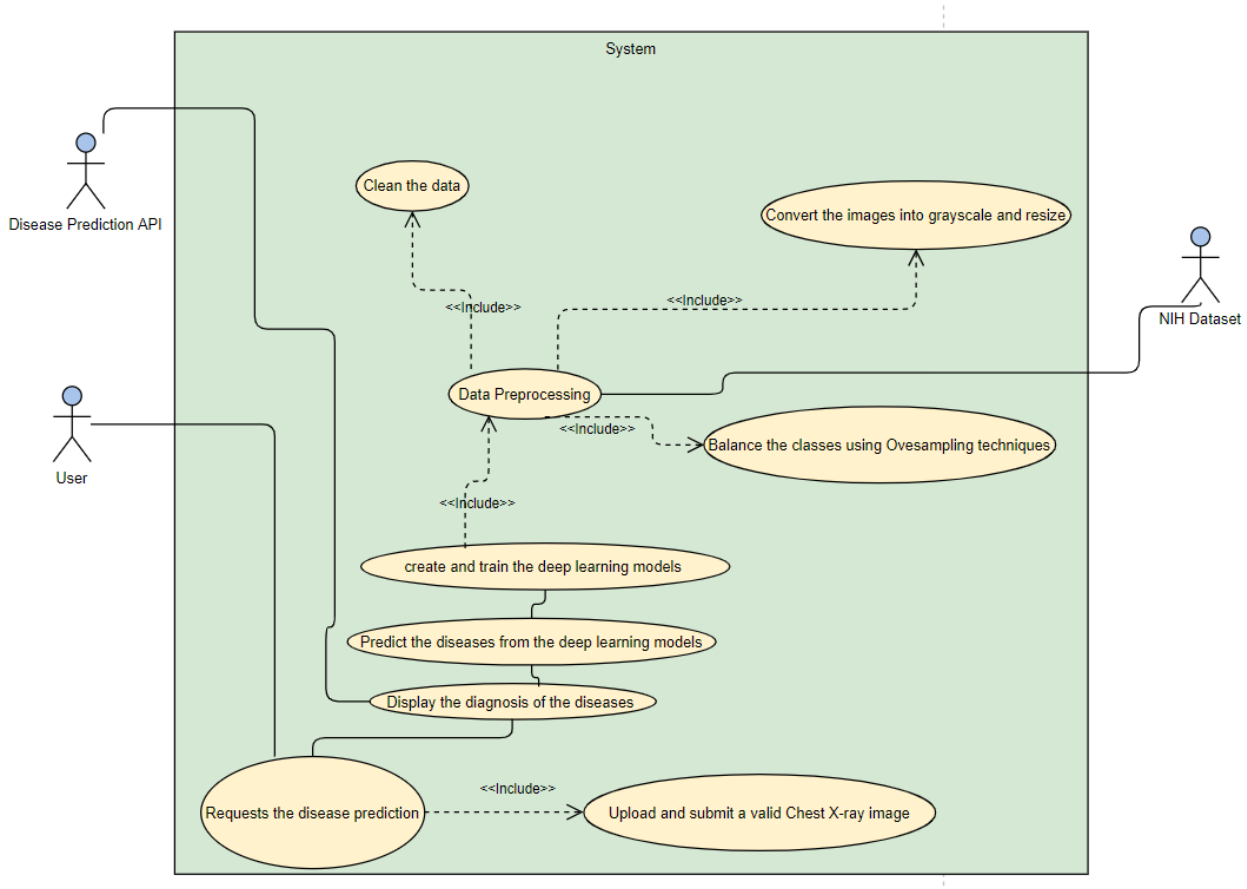

Figure 5.2.1.1 Use Case Diagram

5.2.2 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

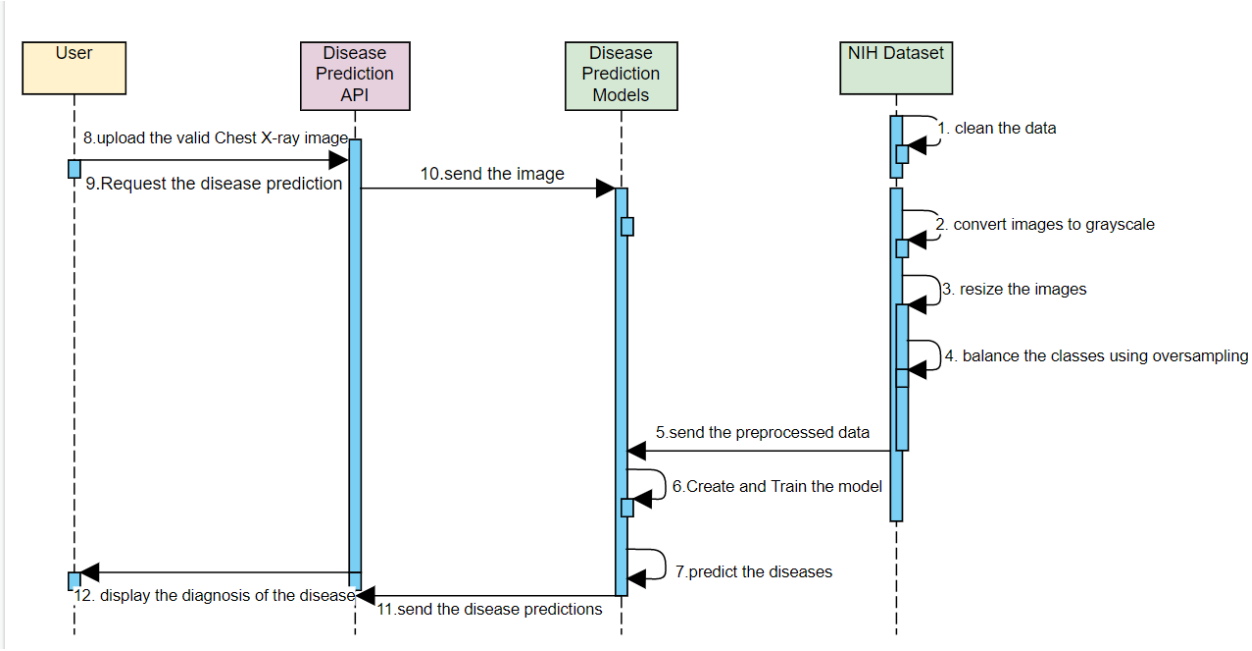

Figure 5.2.2.1 Sequence Diagram

5.3 DATA FLOW DIAGRAM

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams.

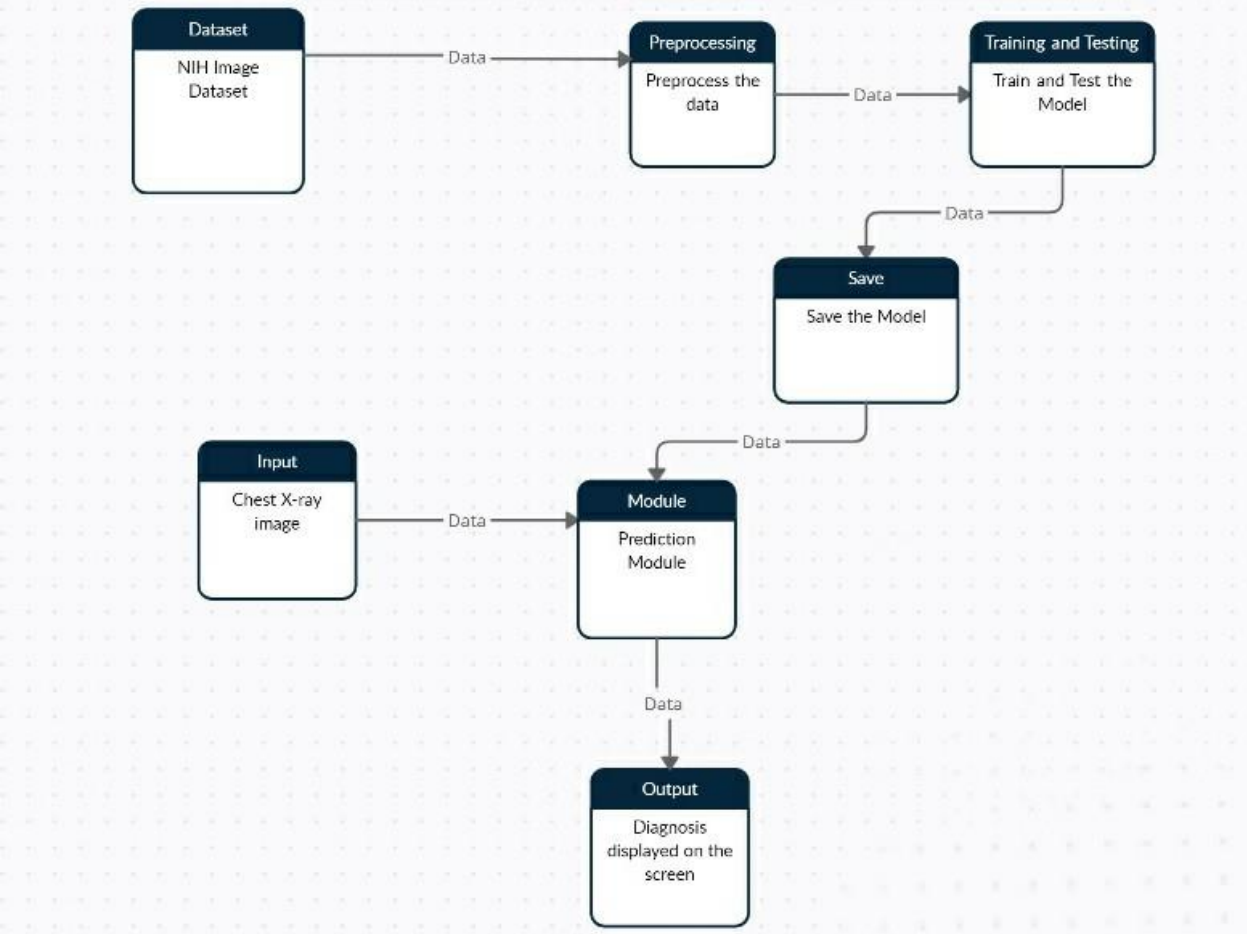

Figure 5.3.1 Data Flow Diagram

5.4 MODULE DESIGN AND ORGANIZATION

There are two main modules in the project. The first one is the Disease Prediction module and the second one is the User Interface module.

The User Interface module consists of two main buttons. The first button is used for choosing the image to be uploaded by the user. The second button is used to submit the image uploaded by the user. The User Interface module also consists of Image Displayer which displays the image uploaded by the user. There is a heading called DIAGNOSIS under which the disease predictions from the image will be displayed.

In the Disease Prediction module, we first load all the 14 disease prediction models. After that, when the image is submitted by the user, it is received by this module. After receiving the image, predictions are performed on the image with the help of all the 14 types of models. And the predictions will be sent to the User Interface module.

6. IMPLEMENTATION AND RESULT ANALYSIS

6.1 INTRODUCTION

The implementation details of the project will be discussed in this chapter. All the steps that were followed in the development of this project like Data Preprocessing, Design and Training of the model will be discussed in detail.

6.2 DESCRIPTION OF KEY PARAMETERS AND FUNCTIONS

The key parameters and functions that have been used in the project are:

MobileNets

MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks.

Dropout

Dropout is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons. regularization is way to prevent over-fitting. dropout refers to ignoring units (i.e., neurons) during the training phase of certain set of neurons which is chosen at random. At each training stage, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.

Global Average Pooling

Global Average Pooling is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task in the last layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer.

Dense

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

Sigmoid

Sigmoid function is sig(x) = 1/(1+exp(-x)). Sigmoid function produces similar results to step function in that the output is between 0 and 1. The curve crosses 0.5 at z=0, which we can set up rules for the activation function, such as: If the sigmoid neuron's output is larger than or equal to 0.5, it outputs 1; if the output is smaller than 0.5, it outputs 0. It is generally used in the last layer of the model.

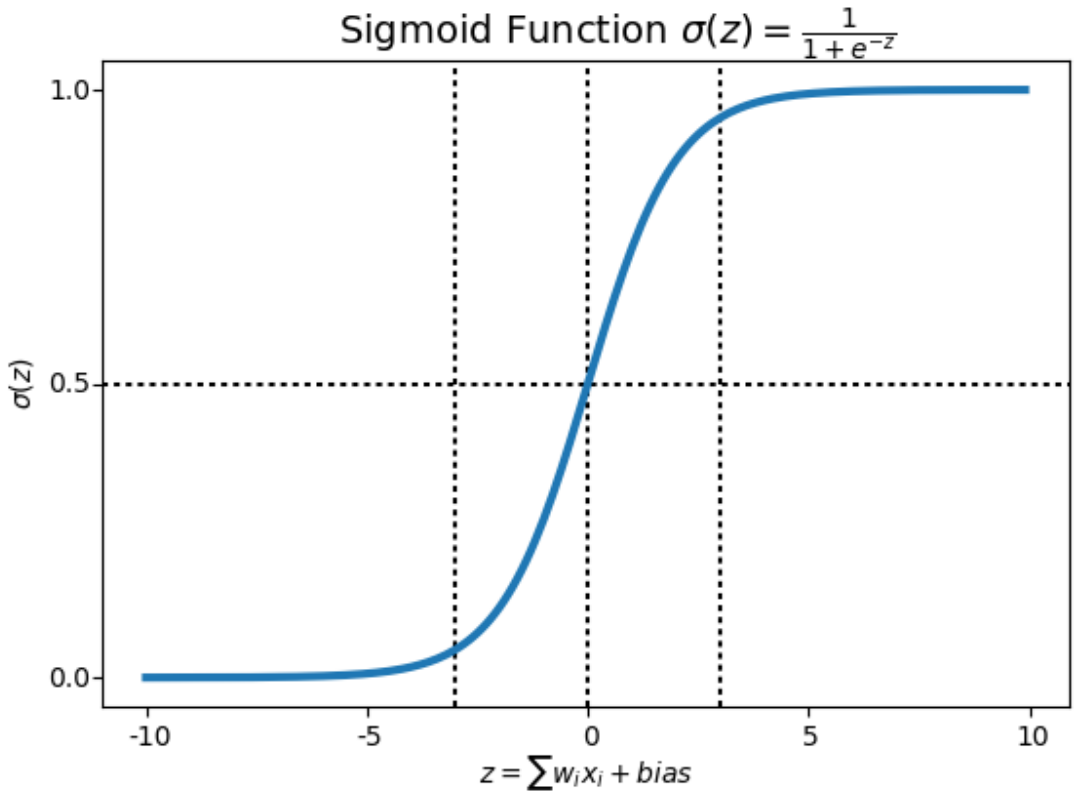

Figure 6.2.1 Sigmoid Function

ReLU

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. The formula is given by a = max(0,z)

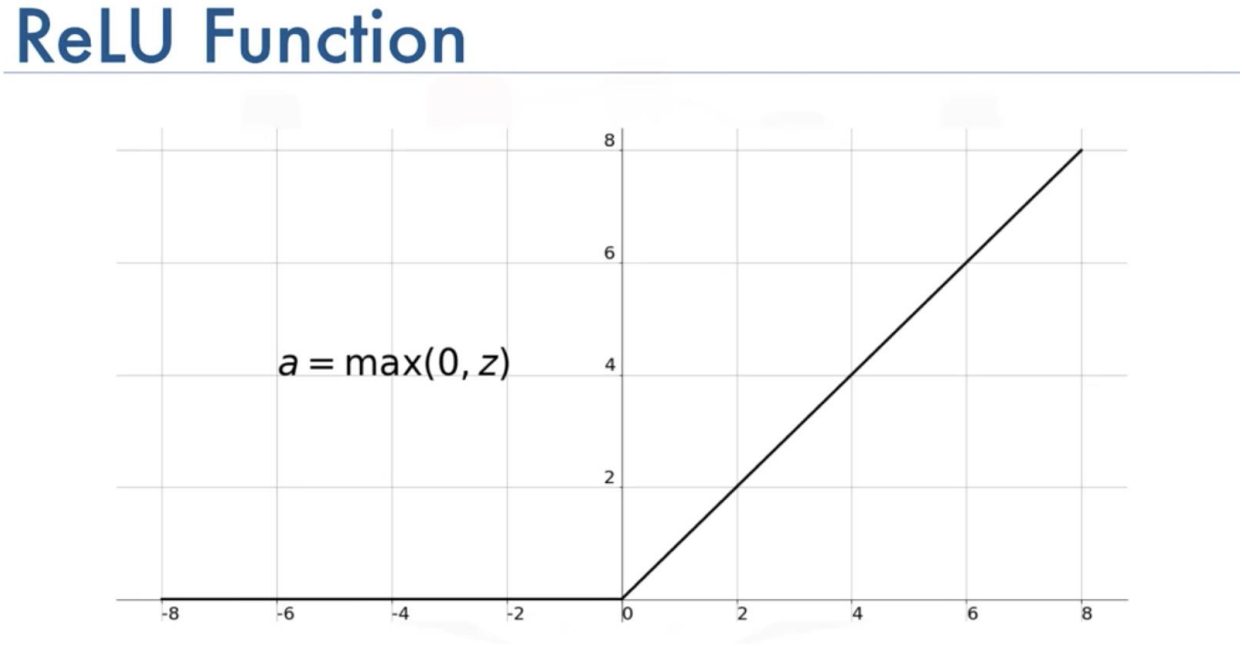

Figure 6.2.2 ReLU Function

Binary CrossEntropy

Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value. Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.

6.3 METHOD OF IMPLEMENTATION

6.3.1 Implementation Steps

6.3.1.1 Choosing and Collecting the Dataset

A dataset can be viewed as a collection of *data objects*, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. The first step of this project is the dataset selection. The project requires a large number of Chest X-ray images with proper labelling. Chest X-ray exams are one of the most frequent and cost-effective medical imaging examinations available. However, clinical diagnosis of a chest X-ray can be challenging and sometimes more difficult than diagnosis via chest CT imaging. The lack of large publicly available datasets with annotations means it is still very difficult, if not impossible, to achieve clinically relevant computer-aided detection and diagnosis (CAD) in real world medical sites with chest X-rays. One major hurdle in creating large X-ray image datasets is the lack resources for labeling so many images. Prior to the release of NIH Chest X-ray Dataset, Openi was the largest publicly available source of chest X-ray images with 4,143 images available.

The NIH Chest X-ray Dataset is comprised of 112,120 X-ray images with disease labels from 30,805 unique patients. To create these labels, the authors used Natural Language Processing to text-mine disease classifications from the associated radiological reports. The labels are expected to be >90% accurate and suitable for weakly-supervised learning.

There are 15 classes (14 diseases, and one for "No findings"). Images can be classified as "No findings" or one or more disease classes:

- Atelectasis
- Consolidation
- Infiltration
- Pneumothorax
- Edema
- Emphysema
- Fibrosis
- Effusion

- Pneumonia
- Pleural thickening
- Cardiomegaly
- Nodule Mass
- Hernia

The dataset was collected from the Kaggle Website.

6.3.1.2 Data Preprocessing

Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

It is very much usual to have missing values in your dataset. It may have happened during data collection, or maybe due to some data validation rule, but regardless missing values must be taken into consideration.

Eliminate rows with missing data technique is a simple and sometimes effective strategy. Fails if many objects have missing values. If a feature has mostly missing values, then that feature itself can also be eliminated.

Estimate missing values is good if only a reasonable percentage of values are missing, then we can also run simple interpolation methods to fill in those values. However, most common method of dealing with missing values is by filling them in with the mean, median or mode value of the respective feature.

In the project, the Eliminate rows with missing data technique was used. The labels of the Chest X-ray images were present in a csv file. The labels were converted using one hot encoding technique. The target label of each image was a vector of size 14 where each individual value was a binary value i.e., either 0 or 1. 0 means that the particular disease was absent. Whereas 1 means that the particular disease was present.

The images were converted to grayscale and they were resized to a size of width 100 and height 100. The images as well as the labels were stored in the form of numpy arrays. After this, the data was separated based on the classes.

The target classes corresponding to the images were observed to be heavily imbalanced.

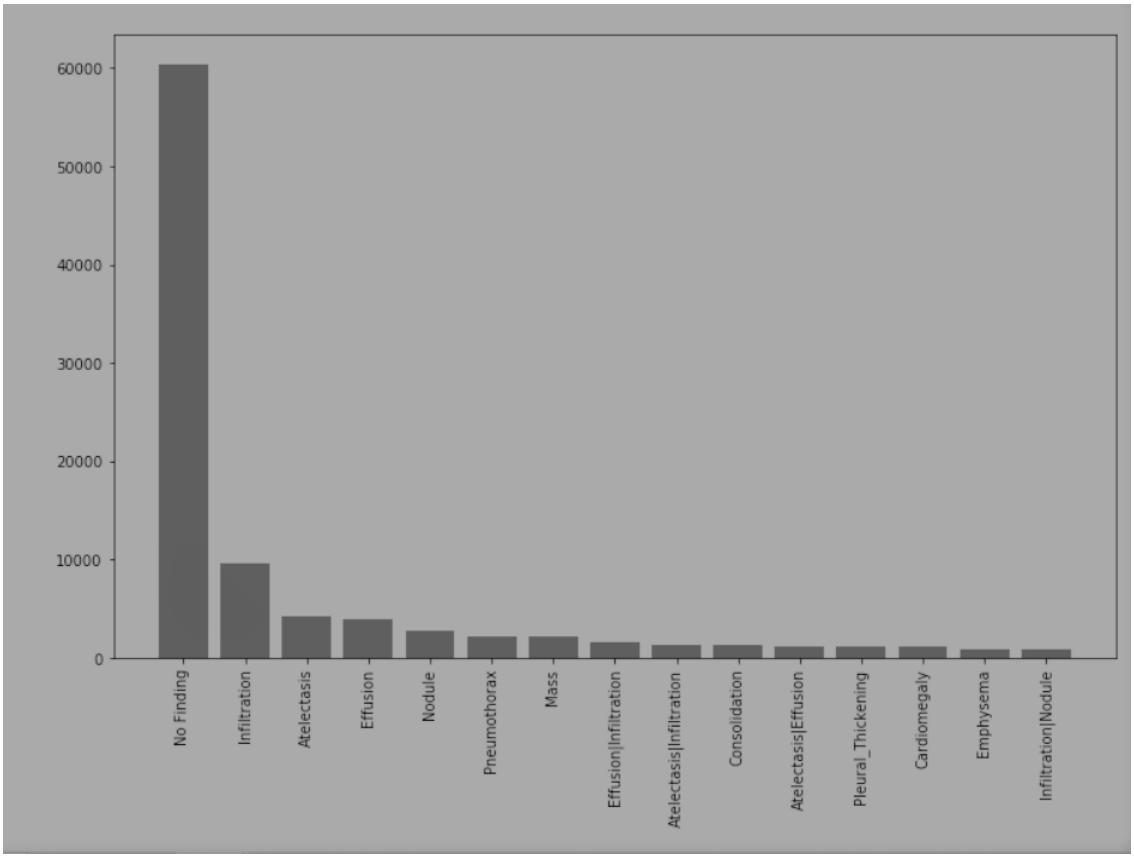

Figure 6.3.1.2.1 Frequency of various classes

If the imbalance is not resolved, then the trained model would be biased towards the class with the highest frequency. In classification problems (binary and multiclass), datasets are often imbalanced which means that one class has a higher number of samples than others. This will lead to bias during the training of the model, the class containing a higher number of samples will be preferred more over the classes containing a lower number of samples. Having bias will, in turn, increase the true-negative and false-positive rates. Hence to overcome this bias of the

model we need to make the dataset balanced containing an approximately equal number of samples in all the classes.

So, this imbalanced was eliminated using SMOTETomek technique. Undersampling technique samples down or reduces the samples of the class containing more data equivalent to the class containing the least samples. Suppose class A has 900 samples and class B has 100 samples, then the imbalance ratio is 9:1. Using the undersampling technique we keep class B as 100 samples and from class A we randomly select 100 samples out of 900. Then the ratio becomes 1:1 and we can say it's balanced. From the imblearn library, we have the under_sampling module which contains various libraries to achieve undersampling. Undersampling is also referred to as downsampling as it reduces the number of samples. This method should only be used for large datasets as otherwise there's a huge loss of data, which is not good for the model. Losing out on data is not appropriate as it could hold important information regarding the dataset.

Oversampling is just the opposite of undersampling. Here the class containing less data is made equivalent to the class containing more data. This is done by adding more data to the least sample containing class. Let's take the same example of undersampling, then, in this case, class A will remain 900 and class B will also be 900 (which was previously 100). Hence the ratio will be 1:1 and it'll be balanced. The imblearn library contains an over_sampling module which contains various libraries to achieve oversampling. Oversampling is also referred to as upsampling as it increases the number of samples. This method should primarily be used in the small or medium-sized dataset. It is better than undersampling as there is no loss of data instead more data is added, which can prove to be good for the model.

SMOTE means Synthetic Minority Oversampling Technique. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b. This procedure can be used to create as many synthetic examples for the minority class as are required. It first uses random undersampling to trim the number of examples in the majority class, then use SMOTE to oversample the minority class to balance the class distribution.

Tomek Links is one of a modification from Condensed Nearest Neighbors (CNN, not to be confused with Convolutional Neural Network) undersampling technique that is developed by Tomek (1976). Unlike the CNN method that are only randomly select the samples with its k nearest neighbors from the majority class that wants to be removed, the Tomek Links method uses the rule to selects the pair of observation (say, a and b) that are fulfilled these properties:

- The observation a's nearest neighbor is b.
- The observation b's nearest neighbor is a.
- Observation a and b belong to a different class. That is, a and b belong to the minority and majority class (or vice versa), respectively.

SMOTETomek is somewhere upsampling and downsampling. SMOTETomek is a hybrid method which is a mixture of the above two methods, it uses an under-sampling method (Tomek) with an oversampling method (SMOTE). This is present within imblearn.combine module. After using this method, the final dataset of the each class had equal number of images belonging to the disease class and the no disease class.

6.3.1.3 Model Creation and Training

The model that was chosen for the purpose of prediction of diseases in this project is CNN (Convolutional Neural Network).

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

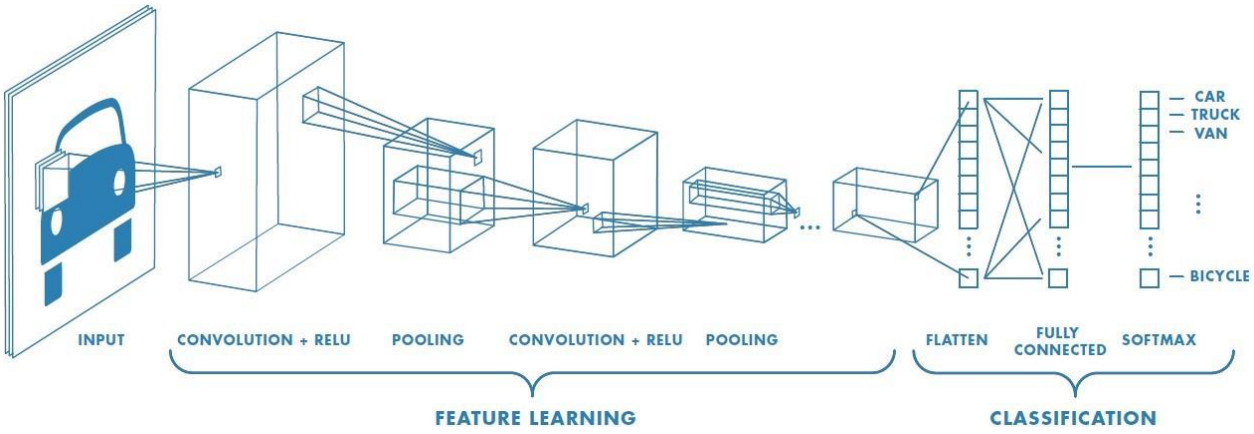

Figure 6.3.1.3.1 CNN

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet
- MobileNet

In this project, the MobileNetV1 architecture has been used. MobileNet is an efficient and portable CNN architecture that is used in real world applications. MobileNets primarily use depthwise separable convolutions in place of the standard convolutions used in earlier architectures to build lighter models. MobileNets introduce two new global hyperparameters(width multiplier and resolution multiplier) that allow model developers to trade off latency or accuracy for speed and low size depending on their requirements.

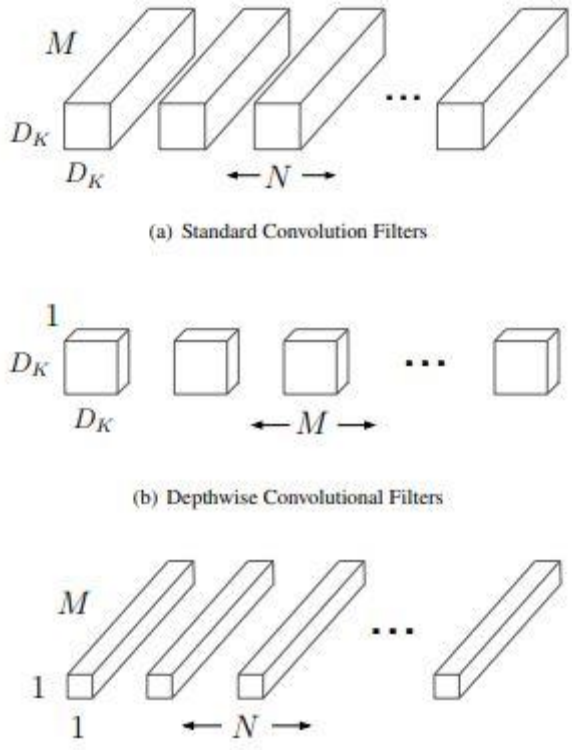

Figure 6.3.1.3.2 MobileNet

MobileNets are built on depthwise seperable convolution layers. Each depthwise seperable convolution layer consists of a depthwise convolution and a pointwise convolution. Counting depthwise and pointwise convolutions as separate layers, a MobileNet has 28 layers. A standard MobileNet has 4.2 million parameters which can be further reduced by tuning the width multiplier hyperparameter appropriately.

Depthwise separable convolution is a depthwise convolution followed by a pointwise convolution as follows:

- Depthwise convolution is the channel-wise DK×DK spatial convolution. Suppose in the figure above, we have 5 channels, then we will have 5 DK×DK spatial convolution.
- Pointwise convolution actually is the 1×1 convolution to change the dimension.

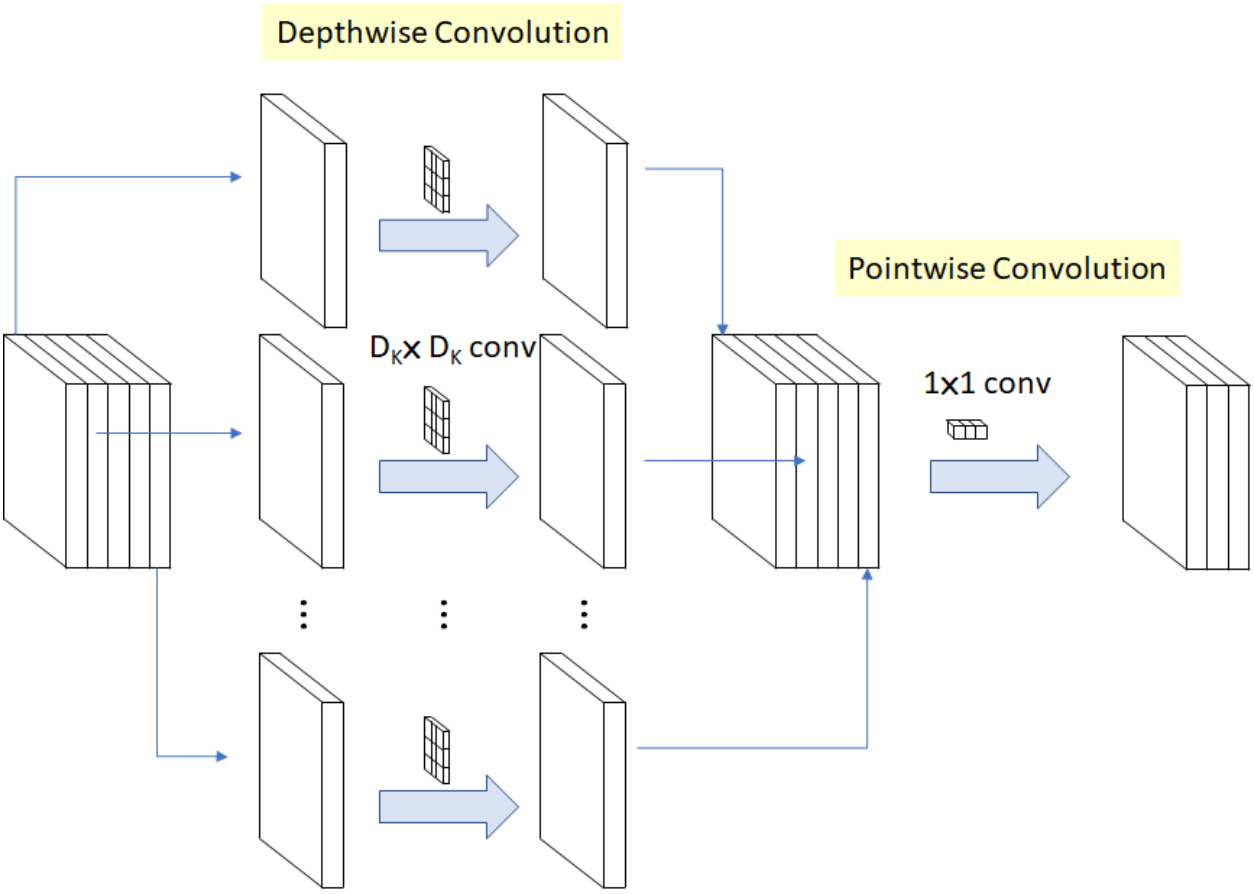

Figure 6.3.1.3.3 Depthwise and Pointwise Convolution

The MobileNetV1 architecture was chosen for the project because of the following reasons:

- It is fast
- It is small in size and less parameters when compared to normal CNN
- It is suitable for Mobile and Embedded devices and can be easily integrated into websites.

6.3.1.4 Performance Metrics

The metrics that have been used in the project to evaluate the performance of the model are:

- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrix

Accuracy is the ratio of number of correct predictions to the total number of input samples. It works well only if there are equal number of samples belonging to each class.

Precision is the number of correct positive results divided by the number of positive results predicted by the classifier.

Recall is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model.

Confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa.

6.3.2 Source Code

Data PreProcessing

```
import numpy as np

import pandas as pd

import os

dise=["Atelectasis",

"Consolidation",

"Infiltration",

"Pneumothorax",

"Edema",

"Emphysema",

"Fibrosis",

"Effusion",

"Pneumonia",

"Pleural_thickening",

"Cardiomegaly",

"Nodule","Mass",

"Hernia"]

for i in range(len(dise)):

    dise[i]=dise[i].lower()

y_map={}
```

```
for i in range(len(dise)):
    y_map[dise[i]]=i
x_y={}
cols=data.keys()
z=[0 for i in range(len(dise))]
for i in range(112120):
    a,b='Image Index','Finding Labels'
    for j in range(len(dise)):
        z[j]=0
    for j in data[b][i].split("|"):
        if j!="No Finding":
            z[y_map[j.lower()]]=1
    x_y[data[a][i]]=tuple(z)
x_data=[]
y_data=[]
import cv2
import matplotlib.pyplot as plt
path="../input/data/"
c=0
for k in os.listdir(path):
    if "images" in k:
```

```
    a=path+k+"/images/"
    for j in os.listdir(a):
        im=cv2.resize(cv2.imread(a+j,0),(100,100))
        #x_data.append(im)
        cv2.imwrite("./images/"+str(c)+".jpg",im)
        y_data.append(x_y[j])
        if c%1000==0:
            print(c)
        c+=1


import tensorflow as tf
from tensorflow import keras
import cv2
import numpy as np
import random
from imblearn.combine import SMOTETomek
z=np.load("D:/Projects/Major Project/data/newdata/negative.npy")
y_all=np.load("D:\Projects\Major Project\data\y.npy")
l=len(y_all)
for k in range(1,13):
    q=[]
```

```
    for i in range(l):
        if y_all[i][k]==1:
            x=cv2.imread("D:/Projects/Major Project/data/data/"+str(i)+".jpg",0)/255.0
            x=x.reshape(100,100,1)
            q.append(x)
    q=np.array(q)
    x=np.append(z,q,axis=0)
    x=x.reshape(len(x),100*100)
    y=np.array([0 for i in range(12000)]+[1 for i in range(len(q))])
    x_new,y_new=SMOTETomek(random_state=42).fit_resample(x,y)
    x_1=x_new.reshape(len(x_new),100,100,1)
    np.save("D:/Projects/Major Project/data/newdata/"+str(k)+"/x.npy",x_1)
    np.save("D:/Projects/Major Project/data/newdata/"+str(k)+"/y.npy",y_new)
    print("----------"+str(k)+"------")
```

Model Creation and Training

```
import tensorflow as tf
import numpy as np
from tensorflow import keras
import cv2
import os
```

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D, Dropout,GlobalAveragePooling2D

from tensorflow.keras.applications.mobilenet import MobileNet

from tensorflow.keras.callbacks import ModelCheckpoint

with tf.device("GPU:0"):

    for i in range(14):

        x=np.load("D:/Projects/Major Project/data/newdata/"+str(i)+"/x.npy")

        y=np.load("D:/Projects/Major Project/data/newdata/"+str(i)+"/y.npy")

        print(y,len(y))

        base_mobilenet_model = MobileNet(input_shape =  (100,100,1),

                                include_top = False, weights = None)

        x_train,x_test=x[6000:],x[:6000]

        y_train,y_test=y[6000:],y[:6000]

        checkpoint = ModelCheckpoint("D:/Projects/Major Project/new_models/"+str(i)+"/"+str(i)+"({val_accuracy:.2f}).h5",

                                monitor="val_accuracy",mode='max', verbose=1,save_best_only=True)

        model = Sequential()

        model.add(base_mobilenet_model)

        model.add(GlobalAveragePooling2D())

        model.add(Dropout(0.2))

        model.add(Dense(128))
```

```
model.add(Dropout(0.2))

"""model.add(Dense(128))

model.add(Dropout(0.2))"""

model.add(Dense(1, activation = 'sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy',

                metrics = ['accuracy'])

print("\n\n\n-------Model "+str(i)+"--------\n\n\n")

model.fit(x_train,y_train,

                epochs = 20,

                validation_data=(x_test,y_test),batch_size=32,callbacks=[checkpoint])
```

training output:

-------Model 0--------

Epoch 1/20

515/515 [==============================] - 72s 134ms/step - loss: 0.8148 - accuracy: 0.5766 - val_loss: 0.7251 - val_accuracy: 0.5055

Epoch 00001: val_accuracy improved from -inf to 0.50550, saving model to D:/Projects/Major Project/new_models/0\0(0.51).h5

Epoch 2/20

515/515 [==============================] - 78s 152ms/step - loss: 0.6017 - accuracy: 0.6886 - val_loss: 0.5694 - val_accuracy: 0.7157

Epoch 00002: val_accuracy improved from 0.50550 to 0.71567, saving model to D:/Projects/Major Project/new_models/0\0(0.72).h5

Epoch 3/20

515/515 [==============================] - 73s 141ms/step - loss: 0.5483 - accuracy: 0.7255 - val_loss: 0.5987 - val_accuracy: 0.6887

Epoch 00003: val_accuracy did not improve from 0.71567

Epoch 4/20

515/515 [==============================] - 79s 154ms/step - loss: 0.5273 - accuracy: 0.7415 - val_loss: 0.5658 - val_accuracy: 0.7102

Epoch 00004: val_accuracy did not improve from 0.71567

Epoch 5/20

515/515 [==============================] - 71s 139ms/step - loss: 0.5038 - accuracy: 0.7570 - val_loss: 0.6267 - val_accuracy: 0.6700

Epoch 00005: val_accuracy did not improve from 0.71567

Epoch 6/20

515/515 [==============================] - 70s 136ms/step - loss: 0.5003 - accuracy: 0.7631 - val_loss: 0.5542 - val_accuracy: 0.7225

Epoch 00006: val_accuracy improved from 0.71567 to 0.72250, saving model to D:/Projects/Major Project/new_models/0\0(0.72).h5

Epoch 7/20

515/515 [==============================] - 58s 113ms/step - loss: 0.4910 - accuracy: 0.7640 - val_loss: 0.7276 - val_accuracy: 0.6397

Epoch 00007: val_accuracy did not improve from 0.72250

Epoch 8/20

515/515 [==============================] - 20s 38ms/step - loss: 0.4810 - accuracy: 0.7759 - val_loss: 0.6145 - val_accuracy: 0.6787

Epoch 00008: val_accuracy did not improve from 0.72250

Epoch 9/20

515/515 [==============================] - 20s 39ms/step - loss: 0.4676 - accuracy: 0.7801 - val_loss: 0.7383 - val_accuracy: 0.6498

Epoch 00009: val_accuracy did not improve from 0.72250

Epoch 10/20

515/515 [==============================] - 20s 38ms/step - loss: 0.4627 - accuracy: 0.7861 - val_loss: 0.7141 - val_accuracy: 0.7045

Epoch 00010: val_accuracy did not improve from 0.72250

Epoch 11/20

515/515 [==============================] - 20s 38ms/step - loss: 0.4463 - accuracy: 0.7950 - val_loss: 0.6387 - val_accuracy: 0.7172

Epoch 00011: val_accuracy did not improve from 0.72250

Epoch 12/20

515/515 [==============================] - 20s 38ms/step - loss: 0.4360 - accuracy: 0.7997 - val_loss: 1.0180 - val_accuracy: 0.6437

Epoch 00012: val_accuracy did not improve from 0.72250

Epoch 13/20

515/515 [==============================] - 20s 38ms/step - loss: 0.4316 - accuracy: 0.8024 - val_loss: 0.5877 - val_accuracy: 0.7285

Epoch 00013: val_accuracy improved from 0.72250 to 0.72850, saving model to D:/Projects/Major Project/new_models/0\0(0.73).h5

Epoch 14/20

515/515 [==============================] - 19s 38ms/step - loss: 0.4070 - accuracy: 0.8181 - val_loss: 0.5379 - val_accuracy: 0.7450

Epoch 00014: val_accuracy improved from 0.72850 to 0.74500, saving model to D:/Projects/Major Project/new_models/0\0(0.75).h5

Epoch 15/20

515/515 [==============================] - 20s 38ms/step - loss: 0.3872 - accuracy: 0.8267 - val_loss: 0.5937 - val_accuracy: 0.7282

Epoch 00015: val_accuracy did not improve from 0.74500

Epoch 16/20

515/515 [==============================] - 20s 38ms/step - loss: 0.3533 - accuracy: 0.8498 - val_loss: 0.7373 - val_accuracy: 0.6663

Epoch 00016: val_accuracy did not improve from 0.74500

Epoch 17/20

515/515 [==============================] - 20s 38ms/step - loss: 0.3416 - accuracy: 0.8522 - val_loss: 0.8331 - val_accuracy: 0.6838

Epoch 00017: val_accuracy did not improve from 0.74500

Epoch 18/20

515/515 [==============================] - 20s 38ms/step - loss: 0.3129 - accuracy: 0.8689 - val_loss: 0.7497 - val_accuracy: 0.7040

Epoch 00018: val_accuracy did not improve from 0.74500

Epoch 19/20

515/515 [==============================] - 20s 38ms/step - loss: 0.2868 - accuracy: 0.8774 - val_loss: 0.7926 - val_accuracy: 0.6862

Epoch 00019: val_accuracy did not improve from 0.74500

Epoch 20/20

515/515 [==============================] - 20s 38ms/step - loss: 0.2489 - accuracy: 0.8999 - val_loss: 0.7030 - val_accuracy: 0.6967

Epoch 00020: val_accuracy did not improve from 0.74500

[0 0 1 ... 1 0 1] 23956

-------Model 1--------

Epoch 1/20

562/562 [==============================] - 24s 40ms/step - loss: 0.7318 - accuracy: 0.6630 - val_loss: 0.6989 - val_accuracy: 0.5053

Epoch 00001: val_accuracy improved from -inf to 0.50533, saving model to D:/Projects/Major Project/new_models/1\1(0.51).h5

Epoch 2/20

562/562 [==============================] - 21s 38ms/step - loss: 0.4772 - accuracy: 0.7813 - val_loss: 0.5785 - val_accuracy: 0.7140

Epoch 00002: val_accuracy improved from 0.50533 to 0.71400, saving model to D:/Projects/Major Project/new_models/1\1(0.71).h5

Epoch 3/20

562/562 [==============================] - 21s 38ms/step - loss: 0.4244 - accuracy: 0.8120 - val_loss: 0.4611 - val_accuracy: 0.7863

Epoch 00003: val_accuracy improved from 0.71400 to 0.78633, saving model to D:/Projects/Major Project/new_models/1\1(0.79).h5

Epoch 4/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3965 - accuracy: 0.8265 - val_loss: 0.4549 - val_accuracy: 0.7883

Epoch 00004: val_accuracy improved from 0.78633 to 0.78833, saving model to D:/Projects/Major Project/new_models/1\1(0.79).h5

Epoch 5/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3740 - accuracy: 0.8365 - val_loss: 0.5971 - val_accuracy: 0.7025

Epoch 00005: val_accuracy did not improve from 0.78833

Epoch 6/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3559 - accuracy: 0.8452 - val_loss: 0.4869 - val_accuracy: 0.7868

Epoch 00006: val_accuracy did not improve from 0.78833

Epoch 7/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3440 - accuracy: 0.8547 - val_loss: 0.6272 - val_accuracy: 0.7583

Epoch 00007: val_accuracy did not improve from 0.78833

Epoch 8/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3269 - accuracy: 0.8658 - val_loss: 0.6659 - val_accuracy: 0.6890

Epoch 00008: val_accuracy did not improve from 0.78833

Epoch 9/20

562/562 [==============================] - 21s 38ms/step - loss: 0.3186 - accuracy: 0.8670 - val_loss: 0.5474 - val_accuracy: 0.7698

Epoch 00009: val_accuracy did not improve from 0.78833

Epoch 10/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2932 - accuracy: 0.8826 - val_loss: 0.4484 - val_accuracy: 0.7905

Epoch 00010: val_accuracy improved from 0.78833 to 0.79050, saving model to D:/Projects/Major Project/new_models/1\1(0.79).h5

Epoch 11/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2587 - accuracy: 0.8950 - val_loss: 0.5764 - val_accuracy: 0.7760

Epoch 00011: val_accuracy did not improve from 0.79050

Epoch 12/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2592 - accuracy: 0.8962 - val_loss: 0.5126 - val_accuracy: 0.8105

Epoch 00012: val_accuracy improved from 0.79050 to 0.81050, saving model to D:/Projects/Major Project/new_models/1\1(0.81).h5

Epoch 13/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2095 - accuracy: 0.9185 - val_loss: 0.4154 - val_accuracy: 0.8238

Epoch 00013: val_accuracy improved from 0.81050 to 0.82383, saving model to D:/Projects/Major Project/new_models/1\1(0.82).h5

Epoch 14/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2073 - accuracy: 0.9184 - val_loss: 0.4400 - val_accuracy: 0.8367

Epoch 00014: val_accuracy improved from 0.82383 to 0.83667, saving model to D:/Projects/Major Project/new_models/1\1(0.84).h5

Epoch 15/20

562/562 [==============================] - 21s 38ms/step - loss: 0.2354 - accuracy: 0.9084 - val_loss: 0.4999 - val_accuracy: 0.8160

Epoch 00015: val_accuracy did not improve from 0.83667

Epoch 16/20

562/562 [==============================] - 21s 38ms/step - loss: 0.1687 - accuracy: 0.9327 - val_loss: 0.4666 - val_accuracy: 0.8293

Epoch 00016: val_accuracy did not improve from 0.83667

Epoch 17/20

562/562 [==============================] - 21s 37ms/step - loss: 0.1661 - accuracy: 0.9361 - val_loss: 0.4778 - val_accuracy: 0.8278

Epoch 00017: val_accuracy did not improve from 0.83667

Epoch 18/20

562/562 [==============================] - 21s 37ms/step - loss: 0.2313 - accuracy: 0.9079 - val_loss: 0.5431 - val_accuracy: 0.8053

Epoch 00018: val_accuracy did not improve from 0.83667

Epoch 19/20

562/562 [==============================] - 21s 37ms/step - loss: 0.1059 - accuracy: 0.9622 - val_loss: 0.6124 - val_accuracy: 0.8262

Epoch 00019: val_accuracy did not improve from 0.83667

Epoch 20/20

562/562 [==============================] - 21s 37ms/step - loss: 0.0952 - accuracy: 0.9664 - val_loss: 0.6882 - val_accuracy: 0.7897

Epoch 00020: val_accuracy did not improve from 0.83667

[0 1 1 ... 0 0 1] 39262

-------Model 2--------

Epoch 1/20

1040/1040 [==============================] - 40s 37ms/step - loss: 0.7117 - accuracy: 0.6275 - val_loss: 0.5822 - val_accuracy: 0.7023

Epoch 00001: val_accuracy improved from -inf to 0.70233, saving model to D:/Projects/Major Project/new_models/2\2(0.70).h5

Epoch 2/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5662 - accuracy: 0.7090 - val_loss: 0.5722 - val_accuracy: 0.7197

Epoch 00002: val_accuracy improved from 0.70233 to 0.71967, saving model to D:/Projects/Major Project/new_models/2\2(0.72).h5

Epoch 3/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5479 - accuracy: 0.7250 - val_loss: 0.5584 - val_accuracy: 0.7098

Epoch 00003: val_accuracy did not improve from 0.71967

Epoch 4/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5295 - accuracy: 0.7353 - val_loss: 0.6022 - val_accuracy: 0.6845

Epoch 00004: val_accuracy did not improve from 0.71967

Epoch 5/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5203 - accuracy: 0.7427 - val_loss: 0.6634 - val_accuracy: 0.7040

Epoch 00005: val_accuracy did not improve from 0.71967

Epoch 6/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5136 - accuracy: 0.7483 - val_loss: 0.5723 - val_accuracy: 0.7158

Epoch 00006: val_accuracy did not improve from 0.71967

Epoch 7/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.5031 - accuracy: 0.7581 - val_loss: 0.5208 - val_accuracy: 0.7508

Epoch 00007: val_accuracy improved from 0.71967 to 0.75083, saving model to D:/Projects/Major Project/new_models/2\2(0.75).h5

Epoch 8/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.4858 - accuracy: 0.7704 - val_loss: 0.5307 - val_accuracy: 0.7432

Epoch 00008: val_accuracy did not improve from 0.75083

Epoch 9/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.4703 - accuracy: 0.7775 - val_loss: 0.7863 - val_accuracy: 0.6713

Epoch 00009: val_accuracy did not improve from 0.75083

Epoch 10/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.4495 - accuracy: 0.7902 - val_loss: 0.5244 - val_accuracy: 0.7328

Epoch 00010: val_accuracy did not improve from 0.75083

Epoch 11/20

1040/1040 [==============================] - 38s 37ms/step - loss: 0.4264 - accuracy: 0.8065 - val_loss: 0.5314 - val_accuracy: 0.7513

Epoch 00011: val_accuracy improved from 0.75083 to 0.75133, saving model to D:/Projects/Major Project/new_models/2\2(0.75).h5

Epoch 12/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.4013 - accuracy: 0.8181 - val_loss: 0.5709 - val_accuracy: 0.7540

Epoch 00012: val_accuracy improved from 0.75133 to 0.75400, saving model to D:/Projects/Major Project/new_models/2\2(0.75).h5

Epoch 13/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.3683 - accuracy: 0.8367 - val_loss: 0.5486 - val_accuracy: 0.7578

Epoch 00013: val_accuracy improved from 0.75400 to 0.75783, saving model to D:/Projects/Major Project/new_models/2\2(0.76).h5

Epoch 14/20

1040/1040 [==============================] - 37s 36ms/step - loss: 0.3364 - accuracy: 0.8550 - val_loss: 0.5495 - val_accuracy: 0.7498

Epoch 00014: val_accuracy did not improve from 0.75783

Epoch 15/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.3050 - accuracy: 0.8684 - val_loss: 0.5902 - val_accuracy: 0.7535

Epoch 00015: val_accuracy did not improve from 0.75783

Epoch 16/20

1040/1040 [==============================] - 37s 36ms/step - loss: 0.2686 - accuracy: 0.8867 - val_loss: 0.5631 - val_accuracy: 0.7605

Epoch 00016: val_accuracy improved from 0.75783 to 0.76050, saving model to D:/Projects/Major Project/new_models/2\2(0.76).h5

Epoch 17/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.2321 - accuracy: 0.9061 - val_loss: 0.5785 - val_accuracy: 0.7613

Epoch 00017: val_accuracy improved from 0.76050 to 0.76133, saving model to D:/Projects/Major Project/new_models/2\2(0.76).h5

Epoch 18/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.2171 - accuracy: 0.9104 - val_loss: 0.6269 - val_accuracy: 0.7708

Epoch 00018: val_accuracy improved from 0.76133 to 0.77083, saving model to D:/Projects/Major Project/new_models/2\2(0.77).h5

Epoch 19/20

1040/1040 [==============================] - 38s 36ms/step - loss: 0.1934 - accuracy: 0.9222 - val_loss: 0.5828 - val_accuracy: 0.7705

Epoch 00019: val_accuracy did not improve from 0.77083

Epoch 20/20

1040/1040 [==============================] - 38s 37ms/step - loss: 0.1590 - accuracy: 0.9358 - val_loss: 0.7883 - val_accuracy: 0.7553

Epoch 00020: val_accuracy did not improve from 0.77083

[1 0 1 ... 1 0 0] 23996

-------Model 4--------

Epoch 1/20

563/563 [==============================] - 23s 38ms/step - loss: 0.5961 - accuracy: 0.7694 - val_loss: 0.8070 - val_accuracy: 0.5017

Epoch 00001: val_accuracy improved from -inf to 0.50167, saving model to D:/Projects/Major Project/new_models/4\4(0.50).h5

Epoch 2/20

563/563 [==============================] - 21s 38ms/step - loss: 0.2938 - accuracy: 0.8810 - val_loss: 0.3481 - val_accuracy: 0.8467

Epoch 00002: val_accuracy improved from 0.50167 to 0.84667, saving model to D:/Projects/Major Project/new_models/4\4(0.85).h5

Epoch 3/20

563/563 [==============================] - 21s 37ms/step - loss: 0.2377 - accuracy: 0.9084 - val_loss: 0.3629 - val_accuracy: 0.8425

Epoch 00003: val_accuracy did not improve from 0.84667

Epoch 4/20

563/563 [==============================] - 21s 38ms/step - loss: 0.2299 - accuracy: 0.9098 - val_loss: 0.3810 - val_accuracy: 0.8448

Epoch 00004: val_accuracy did not improve from 0.84667

Epoch 5/20

563/563 [==============================] - 21s 38ms/step - loss: 0.2104 - accuracy: 0.9187 - val_loss: 0.3459 - val_accuracy: 0.8755

Epoch 00005: val_accuracy improved from 0.84667 to 0.87550, saving model to D:/Projects/Major Project/new_models/4\4(0.88).h5

Epoch 6/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1693 - accuracy: 0.9353 - val_loss: 0.2419 - val_accuracy: 0.9047

Epoch 00006: val_accuracy improved from 0.87550 to 0.90467, saving model to D:/Projects/Major Project/new_models/4\4(0.90).h5

Epoch 7/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1659 - accuracy: 0.9351 - val_loss: 0.4727 - val_accuracy: 0.8390

Epoch 00007: val_accuracy did not improve from 0.90467

Epoch 8/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1613 - accuracy: 0.9390 - val_loss: 0.3599 - val_accuracy: 0.8812

Epoch 00008: val_accuracy did not improve from 0.90467

Epoch 9/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1367 - accuracy: 0.9491 - val_loss: 0.3011 - val_accuracy: 0.8672

Epoch 00009: val_accuracy did not improve from 0.90467

Epoch 10/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1319 - accuracy: 0.9499 - val_loss: 0.2123 - val_accuracy: 0.9220

Epoch 00010: val_accuracy improved from 0.90467 to 0.92200, saving model to D:/Projects/Major Project/new_models/4\4(0.92).h5

Epoch 11/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1279 - accuracy: 0.9530 - val_loss: 0.4930 - val_accuracy: 0.8448

Epoch 00011: val_accuracy did not improve from 0.92200

Epoch 12/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1144 - accuracy: 0.9558 - val_loss: 0.2364 - val_accuracy: 0.9068

Epoch 00012: val_accuracy did not improve from 0.92200

Epoch 13/20

563/563 [==============================] - 21s 37ms/step - loss: 0.1023 - accuracy: 0.9616 - val_loss: 0.2983 - val_accuracy: 0.8818

Epoch 00013: val_accuracy did not improve from 0.92200

Epoch 14/20

563/563 [==============================] - 21s 37ms/step - loss: 0.0925 - accuracy: 0.9650 - val_loss: 0.4403 - val_accuracy: 0.8643

Epoch 00014: val_accuracy did not improve from 0.92200

Epoch 15/20

563/563 [==============================] - 21s 37ms/step - loss: 0.0773 - accuracy: 0.9717 - val_loss: 0.3448 - val_accuracy: 0.9202

Epoch 00015: val_accuracy did not improve from 0.92200

Epoch 16/20

563/563 [==============================] - 21s 37ms/step - loss: 0.0698 - accuracy: 0.9728 - val_loss: 0.2740 - val_accuracy: 0.9183

Epoch 00016: val_accuracy did not improve from 0.92200

Epoch 17/20

563/563 [==============================] - 21s 38ms/step - loss: 0.0630 - accuracy: 0.9773 - val_loss: 0.2426 - val_accuracy: 0.9265

Epoch 00017: val_accuracy improved from 0.92200 to 0.92650, saving model to D:/Projects/Major Project/new_models/4\4(0.93).h5

Epoch 18/20

563/563 [==============================] - 21s 38ms/step - loss: 0.0620 - accuracy: 0.9774 - val_loss: 0.2868 - val_accuracy: 0.9107

Epoch 00018: val_accuracy did not improve from 0.92650

Epoch 19/20

563/563 [==============================] - 21s 37ms/step - loss: 0.0489 - accuracy: 0.9822 - val_loss: 0.1959 - val_accuracy: 0.9347

Epoch 00019: val_accuracy improved from 0.92650 to 0.93467, saving model to D:/Projects/Major Project/new_models/4\4(0.93).h5

Epoch 20/20

563/563 [==============================] - 21s 38ms/step - loss: 0.0477 - accuracy: 0.9835 - val_loss: 0.2385 - val_accuracy: 0.9270

Epoch 00020: val_accuracy did not improve from 0.93467

Performance Evaluation of the Models:

```
import tensorflow as tf

from tensorflow import keras

import numpy as np

import os

import cv2
```

```
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix, accuracy_score

diseases=["Atelectasis",

"Consolidation",

"Infiltration",

"Pneumothorax",

"Edema",

"Emphysema",

"Fibrosis",

"Effusion",

"Pneumonia",

"Pleural_thickening",

"Cardiomegaly",

"Nodule","Mass",

"Hernia"]

data_path="D:/Projects/Major Project/data/newdata/"

for i in range(14):

    x=np.load(data_path+str(i)+"/x.npy")

    y=np.load(data_path+str(i)+"/y.npy")

    perm=np.random.permutation(len(y))

    x=x[perm]
```

```
    y=y[perm]

    x_test,y_test=x[:6000],y[:6000]

    model = keras.models.load_model("Website/models/"+str(i)+"/"+str(i)+".h5")

    y_pred=model.predict(x_test)

    y_pred=np.array([0 if i<0.5 else 1 for i in y_pred])


    print("--------Metrics of the Model that predicts the disease "+diseases[i]+" are--------")

    print("Precision Score is ",precision_score(y_test, y_pred ))

    print("Recall Score is ",recall_score(y_test, y_pred ))

    print("F1 score is ", f1_score(y_test, y_pred ))

    print("Accuracy score is ",accuracy_score(y_test,y_pred))

    print("Confusion Matrix is \n",confusion_matrix(y_test,y_pred))

    print("----------------------------")
```

Output:

```
--------Metrics of the Model that predicts the disease Atelectasis are--------

Precision Score is  0.794088989931796

Recall Score is  0.8339017735334243

F1 score is  0.8135085676260189

Accuracy score is  0.8131666666666667

Confusion Matrix is

 [[2434  634]
```

[487 2445]]

--------Metrics of the Model that predicts the disease Consolidation are--------

Precision Score is 0.9017080244924267

Recall Score is 0.9137818419333769

F1 score is 0.907704785077048

Accuracy score is 0.9051666666666667

Confusion Matrix is

[[2633 305]

[264 2798]]

--------Metrics of the Model that predicts the disease Infiltration are--------

Precision Score is 0.9146999325691166

Recall Score is 0.8886341303635769

F1 score is 0.9014786509386942

Accuracy score is 0.9011666666666667

Confusion Matrix is

[[2694 253]

[340 2713]]

--------Metrics of the Model that predicts the disease Pneumothorax are--------

Precision Score is 0.8922546012269938

Recall Score is 0.7718076285240464

F1 score is 0.8276720611773074

Accuracy score is 0.8385

Confusion Matrix is

[[2704 281]

[688 2327]]

--------Metrics of the Model that predicts the disease Edema are--------

Precision Score is 0.9544122435688701

Recall Score is 0.9815807099799062

F1 score is 0.9678058444774641

Accuracy score is 0.9675

Confusion Matrix is

[[2874 140]

[55 2931]]

--------Metrics of the Model that predicts the disease Emphysema are--------

Precision Score is 0.9631320224719101

Recall Score is 0.9192359249329759

F1 score is 0.9406721536351166

Accuracy score is 0.9423333333333334

Confusion Matrix is

[[2911 105]

[241 2743]]

--------Metrics of the Model that predicts the disease Fibrosis are--------

Precision Score is 0.955

Recall Score is 0.9553184394798266

F1 score is 0.9551591931988666

Accuracy score is 0.9551666666666667

Confusion Matrix is

[[2866 135]

[134 2865]]

--------Metrics of the Model that predicts the disease Effusion are--------

Precision Score is 0.8614072494669509

Recall Score is 0.8117883456128601

F1 score is 0.8358620689655173

Accuracy score is 0.8413333333333334

Confusion Matrix is

[[2624 390]

[562 2424]]

--------Metrics of the Model that predicts the disease Pneumonia are--------

Precision Score is 0.9634350888963435

Recall Score is 0.9614998326079679

F1 score is 0.9624664879356568

Accuracy score is 0.9626666666666667

Confusion Matrix is

[[2904 109]

[115 2872]]

--------Metrics of the Model that predicts the disease Pleural_thickening are--------

Precision Score is 0.9349485562562231

Recall Score is 0.9306243805748265

F1 score is 0.9327814569536423

Accuracy score is 0.9323333333333333

Confusion Matrix is

[[2777 196]

[210 2817]]

--------Metrics of the Model that predicts the disease Cardiomegaly are--------

Precision Score is 0.9502798814619691

Recall Score is 0.9636060100166944

F1 score is 0.956896551724138

Accuracy score is 0.9566666666666667

Confusion Matrix is

[[2854 151]

[109 2886]]

--------Metrics of the Model that predicts the disease Nodule are--------

Precision Score is 0.9121851599727706

Recall Score is 0.9002351360429963

F1 score is 0.9061707523245984

Accuracy score is 0.9075

Confusion Matrix is

[[2765 258]

[297 2680]]

--------Metrics of the Model that predicts the disease Mass are--------

Precision Score is 0.8942398489140698

Recall Score is 0.9266144814090019

F1 score is 0.9101393560788082

Accuracy score is 0.9065

Confusion Matrix is

[[2598 336]

[225 2841]]

--------Metrics of the Model that predicts the disease Hernia are--------

Precision Score is 0.9850845210473981

Recall Score is 0.987375415282392

F1 score is 0.9862286377965821

Accuracy score is 0.9861666666666666

Confusion Matrix is

[[2945 45]

[38 2972]]

Website API:

```
from PIL import Image

from flask import Flask, render_template, request, redirect

from numpy.lib.utils import source

from werkzeug.utils import secure_filename

import tensorflow as tf

from tensorflow import keras

import sys

import cv2

import numpy as np

import os

from keras.models import load_model

models=[]

for i in range(14):

    models.append(load_model("models/"+str(i)+"/"+str(i)+".h5"))

print('Load complete')

app = Flask(__name__)

diseases=["Atelectasis",

"Consolidation",

"Infiltration",

"Pneumothorax",
```

```
"Edema",
"Emphysema",
"Fibrosis",
"Effusion",
"Pneumonia",
"Pleural_thickening",
"Cardiomegaly",
"Nodule","Mass",
"Hernia"]
@app.route("/",methods=['POST','GET'])
def zz():
    if request.method=='POST':
        if request.files['image'].filename!='':
            with tf.device("CPU:0"):

                #basepath = os.path.dirname(__file__)
                file=request.files['image']
                file_path = os.path.join(
                        'static', 'images', secure_filename(file.filename))
                file.save(file_path)
                try:
```

```
        img = Image.open(file).convert('L')



        img = np.array(img)
        img=img.astype(np.float)
        img = cv2.resize(img,(100,100))
        #img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        img/=255.0


        img=np.reshape(img,(100,100,1))
        res=''
        for i in range(14):
            z=models[i](np.array([img]),training=False).numpy()[0][0]
            z=round(z*100,2)
            if z>=50:
                res+="<p>There is a high chance for the patient to have "+diseases[i]+"</p>"
        if res=='':
            res='<p>No diseases have been identified from the X-ray</p>'
        return render_template("index.html",source=file_path,result=res)
    except:
        return render_template("index.html",source='static/images/default.png',
```

```
                result='<p>Please only upload image files</p>')
        return render_template("index.html",source='static/images/default.png',g='')
    else:
        return render_template("index.html",source='static/images/default.png',g='')
if __name__=="__main__":
    app.run(debug=True)
```

base.html

```
<html>
    <head>

        <link rel="stylesheet" href="{{url_for('static',filename='styles.css')}}"
    </head>
    <body>
        {% block body %}
        {% endblock %}

    </body>
</html>
```

index.html

```
{% extends 'base.html' %}
```

```
{% block body %}
<h1 id='heading'>DISPREDO</h1>
<form method="POST",action="/upload" enctype="multipart/form-data">

    <div id='container'>
    <img src={{source}} width="300px" height="300px" class="im"></img><br><br><br>

    <input type="file" name="image" accept="image/*"  id='filebtn'><br><br><br>
    <button type="submit">Submit</button>
</div>
</form>
    <div id='main'>
        <h1 ><u>DIAGNOSIS</u></h1>
        {{result|safe}}
</div>
{% endblock %}
```

styles.css

```
body{
    background-color: black;
    color:white;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;


}
form
{
    margin-left: 5%; width: 30%;
}
#container
{
    text-align: center;
}
p
{
    border:1px solid green;
    border-collapse: collapse;
    font-size: 120%;
}
```

```
#heading
{

    font-size: 50px;
    width:fit-content;
    margin-right: auto;
    margin-left: auto;
    background-image: url("images/xray.jpg");
    background-size: 100% 100%;
    color: transparent;
    -webkit-background-clip: text;
    animation-name: anim;
    animation-timing-function: ease-in-out;
    animation-duration: 3s;
    animation-iteration-count: infinite;
}
@keyframes anim
{
    0%{
        background-position-x: 0%;
        background-position-y: 100%;
```

```
        background-size: 100% 100%;
    }
    50%
    {
        background-position-x: 100%;
        background-position-y: 0%;
        background-size: 50% 50%;


    }
    100%
    {
        background-position-x: 0%;
        background-position-y: 0%;
        background-size: 100% 100%;
    }
}
.im
{
    border: 3px solid green;


}
```

```
button, #filebtn
{
    border: 0px solid black;
    border-radius: 25px;
    background-color: chartreuse;
    color: black;
    font-size: large;
}

#main {
    margin-left: 60%;
    margin-top: -450px;
}
```

6.3.3 Output Screens

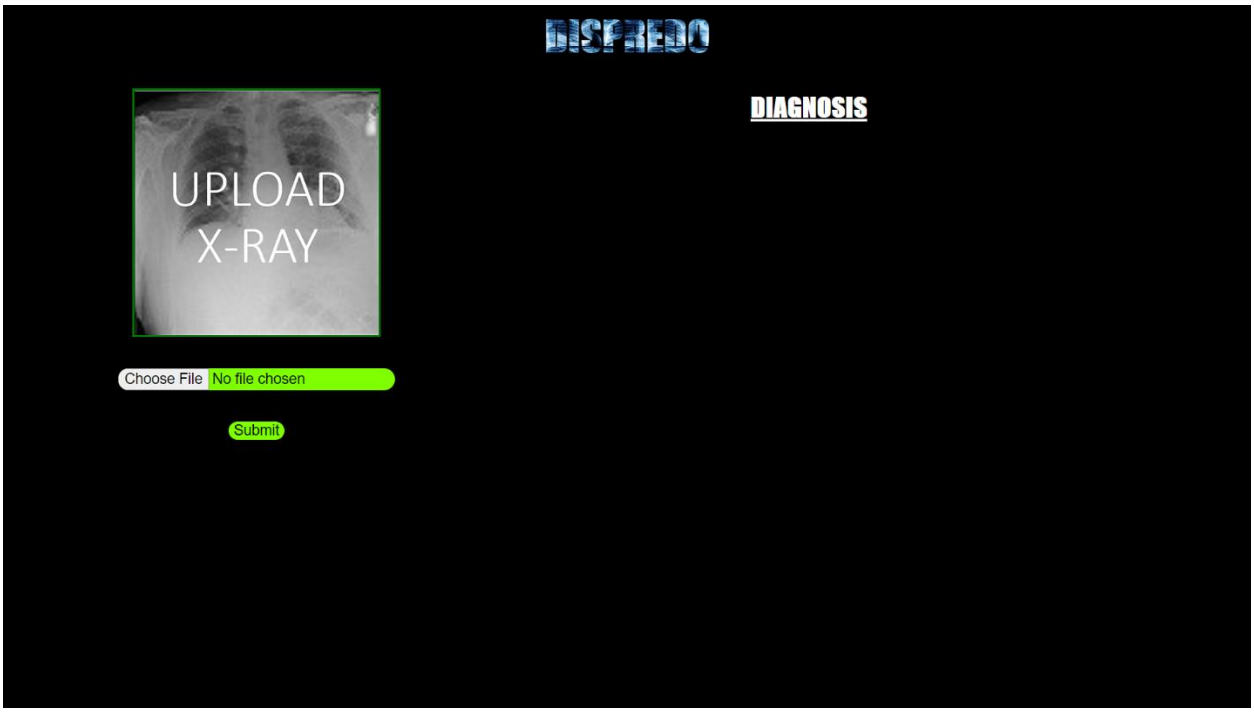

Figure 6.3.3.1 Webpage before uploading image

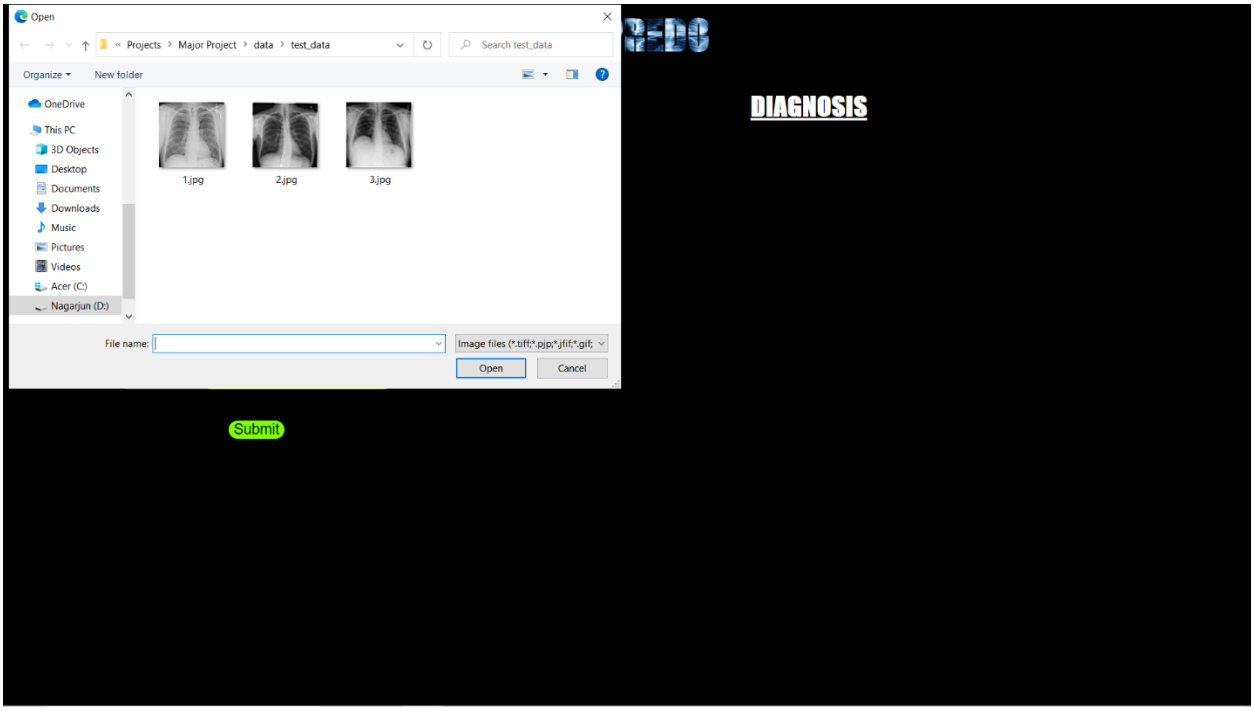

Figure 6.3.3.2 Popup window to select file

Figure 6.3.3.3 Webpage after uploading image

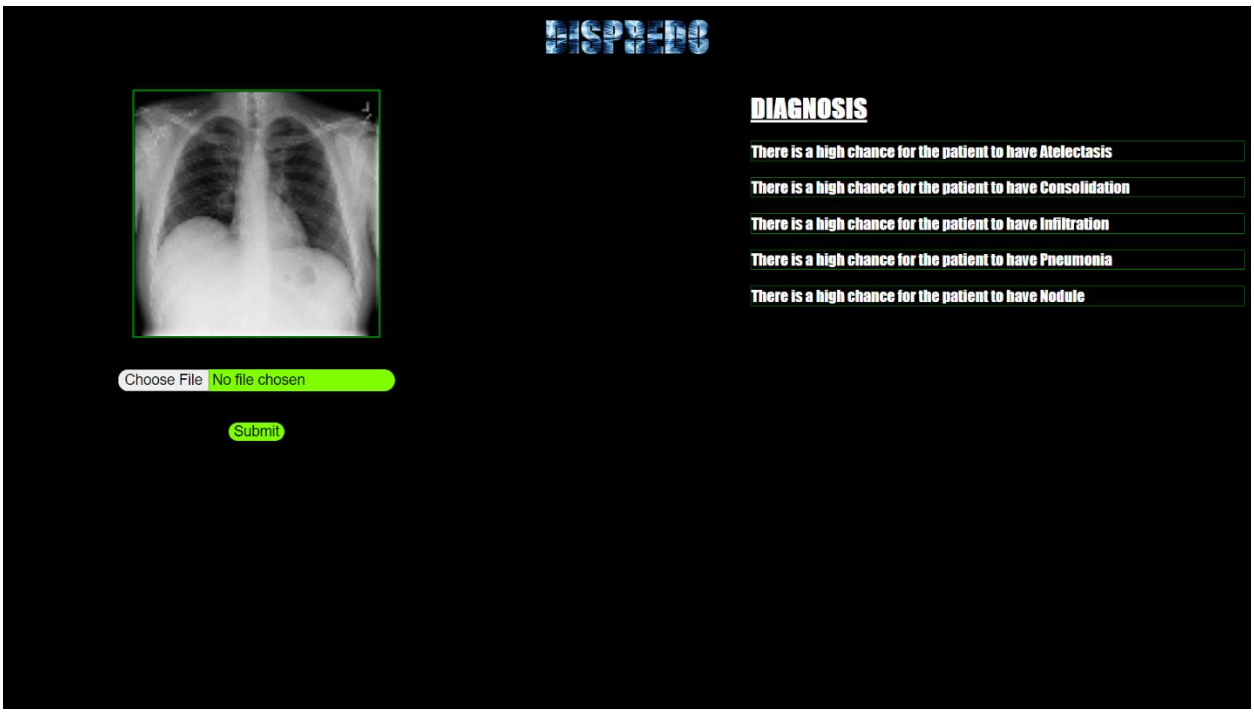

Figure 6.3.3.4 Webpage showing diagnosis of the diseases

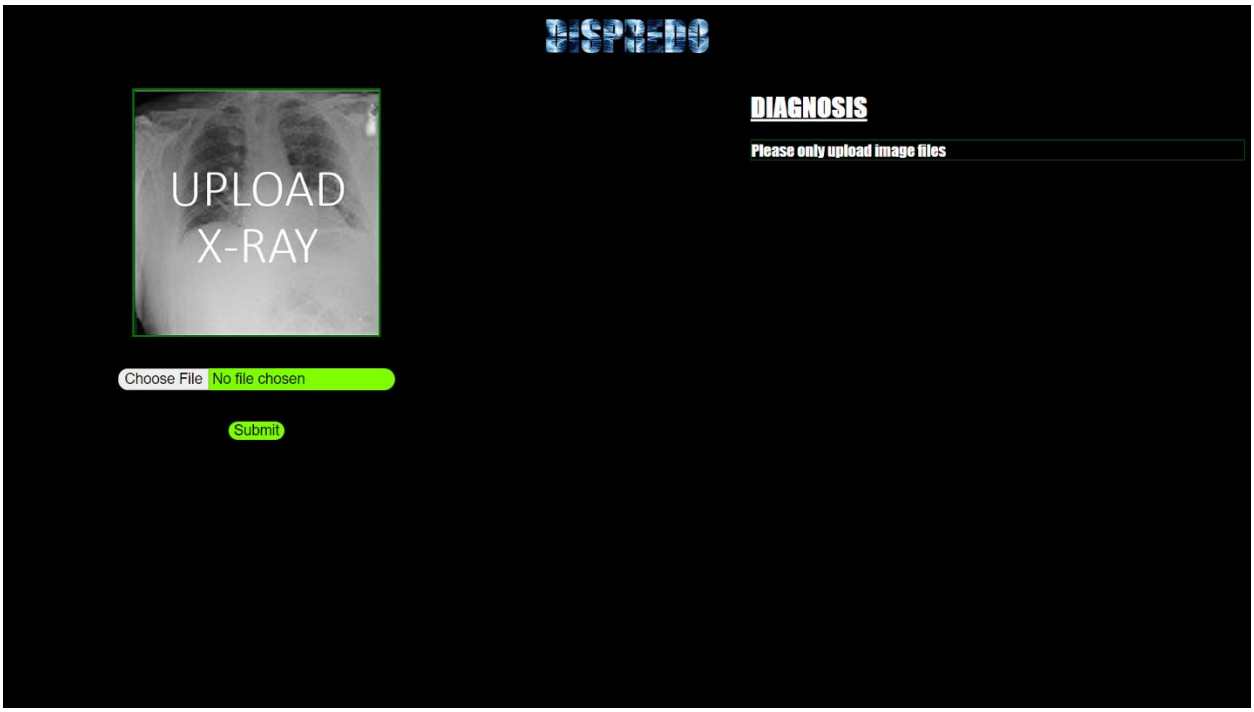

Figure 6.3.3 5 Webpage when invalid file is submitted

7. TESTING AND VALIDATIONS

Testing is a process of executing a program with the intent of finding an error .

7.1. INTRODUCTION

The completion of a system is achieved only after it has been thoroughly tested. Though this gives a feel the project is completed, there cannot be any project without going through this stage. Hence in this stage it is decided whether the project can undergo the real time environment execution without any break downs, therefore a package can be rejected even at this stage.

Testing is a set of activities that can be planned in advance and conducted systematically. The proposed system is tested in parallel with the software that consists of its own phases of analysis, implementation, testing and maintenance.

7.2 TESTING STRATEGIES

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers he set of activities that ensure that the software that has been built is traceable to customer's requirements

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

Top-down Integration

Top down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the

main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

Bottom-up Integration

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

Validation Testing

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

Security Testing

Attempts to verify the protection mechanisms built into the system.

Performance Testing

This method is designed to test runtime performance of software within the context of an integrated system.

7.3 TEST CASES

Website

Case	Expected Result	Observed Result
Main Screen	Displays the main screen contents	Displays the main screen contents
Choose File Button clicked	Opens a popup window to choose a file	Opens a popup window to choose a file
Open popup window	Displays the files available in the system and allows to select the file	Displays the files available in the system and allows to select the file
Open button	Selects the file and closes the popup window	Selects the file and closes the popup window
Submit button	i) If the selected file is a valid image with valid extension, then the predictions are displayed under the DIAGNOSIS heading ii) If the selected file is not a valid image, then a message is displayed to upload only image files	i) If the selected file is a valid image with valid extension, then the predictions are displayed under the DIAGNOSIS heading ii) If the selected file is not a valid image, then a message is displayed to upload only image files
File name displayer	Displays the name of the file selected	Displays the name of the file selected
Image displayer	i) Before submitting the file, a default placeholder image is displayed	i) Before submitting the file, a default placeholder image is displayed

	ii) After submitting the file, if the file is a valid image with valid extension, then the image file chosen is displayed iii) After submitting the file, if the file is not a valid image, then the default placeholder is displayed	ii) After submitting the file, if the file is a valid image with valid extension, then the image file chosen is displayed iii) After submitting the file, if the file is not a valid image, then the default placeholder is displayed

Table 7.3.1 Test Cases

8. CONCLUSION AND FUTURE ENHANCEMENTS

While developing the system, a conscious effort has been made to create and develop a website, making use of available tools, techniques and resources – that would generate a proper system for Disease Predictor.

While making the system, an eye has been kept on making it as user-friendly. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. As in case of any system development process where there are a number of short comings, there have been some shortcomings in the development of this system also.

Disease Predictor is a revolutionary a with a very bright future with further scope for advancements. The opportunities provided from this project are immense and many people can use this website to know the diagnosis of the diseases from the Chest X-rays. Coming to the future enhancements, additional features will be added to the website like Prediction of diseases from the X-rays of other parts of the body and the ability to predict many other diseases will also be added.

REFERENCES

Textbooks Referred

- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems by Aurelien Geron
- Flask Web Development by Miguel Grinberg
- Programming Python by Mark Lutz

Websites Referred

The following links were searched and exploited extensively for project development and implementation.

- https://towardsdatascience.com/
- http://www.w3schools.com/css/
- https://www.w3schools.com/html/
- https://pythonprogramming.net/
- https://www.tutorialspoint.com/flask/
- https://keras.io/
- https://imbalanced-learn.org/

Papers Referred

1. Intelligent Pneumonia Identification from Chest X-Rays by WASIF KHAN , NAZAR ZAKI , and LUQMAN ALI
2. Diagnosis of Pneumonia from Chest X-Ray Images using Deep Learning by Enes AYAN, Halil Murat ÜNVER
3. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning by Pranav Rajpurkar, 1 Jeremy Irvin
4. CXNet-m1: Anomaly Detection on Chest X-Rays With Image-Based Deep Learning by SHUAIJING XU, HAO WU, AND RONGFANG BIE
5. Diagnosis of Chest Diseases in X-Ray images using Deep Convolutional Neural Network by Arjun Choudhary, Abhishek Hazra, Prakash Choudhary