Performance Comparison of Bins Approach with Deep Learning models for Image based Detection and Classification of Covid 19

Submitted in partial fulfillment of the requirements
of MSc in Computer Science
(Research Methodology)

Purva Shah 1184535 pshah30@lakeheadu.ca
Dhruv Patel 1188351 dpate185@lakeheadu.ca

Guide(s):

Name of the Guide
Dr. Jinan Fiaidhi
Professor
Lakehead University

Department of Computer Science
Lakehead University

Abstract

The COVID-19 pandemic is an emerging respiratory infectious disease, also known as coronavirus 2019. It appeared in November 2019 in Hubei province (in China), and more specifically in the city of Wuhan, then spread throughout the whole world. As the number of cases increases with unprecedented speed, many parts of the world are facing a shortage of resources and testing. Faced with this problem, physicians, scientists and engineers, including specialists in Artificial Intelligence (AI), have encouraged the development of a Deep Learning model to help healthcare professionals to detect COVID-19 from chest X-ray images and to determine the severity of the infection in a very short time, with low cost. In this paper, we propose a Deep Convolutional Neural Network (CNN) model to classify COVID-19 infection from normal and other pneumonia cases using chest X-ray images. The proposed architecture is based on the residual neural network and it is constructed by using two parallel levels with different kernel sizes to capture local and global features of the inputs. This model is trained on a dataset publicly available containing a combination of COVID-19,normal and viral pneumonia chest x-ray images. The experimental results reveal that our proposed model represents a promising classification performance on a small dataset which can be further achieved with better results with more training data. Overall, our model can be an interesting tool to help radiologists in the diagnosis and early detection of COVID-19 cases.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	**Abbreviation**	**Expanded form**
1	CXR	Chest X-Ray
2	RT-PCR	Reverse transcription-polymerase chain reaction
3	CT	Computed Tomography
4	AI	Artificial Intelligence
5	CNN	Convolutional Neural Network
6	ReLu	Rectified Linear Unit
7	GGO	Ground-Glass Opacities
8	DCNN	Deep Convolutional Neural Network
9	ADAM	Adaptive moment estimation
10	SGD	Stochastic Gradient Descent
11	VGG	Visual Geometric Group
12	GRAD-CAM	Gradient-weighted Class Activation Mapping
13	ResNet	Residual Neural Network
14	ROC	receiver operating characteristic curve

15	AUROC	Area under ROC curve
16	DFD	Data Flow Diagram
17	API	Application Programming Interface
18	CAM	Class activation mapping
19	SVM	Support Vector Machine

Chapter 1
Introduction

This chapter provides a general overview of the major project through its coherent description, problem formulation and Motivation for the proposed system. It also proffers the problem solution as well as the scope of the project for its future applications.

1.1 Description

The sudden outbreak of the COVID-19 virus, has put an unusual load over healthcare systems across the world. In many countries, the healthcare systems have already been under a lot of pressure. The limited availability of testing kits for diagnosis, limited hospital beds for admission of such patients has made us to find an efficient solution for detection of patients as it is highly contagious.COVID-19 is a respiratory disease that is instigated by a novel coronavirus. The common symptoms appear in the infected person are fever, cough, sore throat, and difficulty in breathing .Vanishing of taste, tiredness, aches, and nasal blockage can also be observed in some patients.

1.2 Problem Formulation

Developing a deep convolutional neural network design tailored for the detection of COVID-19 from chest X-ray(CXR) images by reviewing various methodologies used around the world and attempting to improve the performance of accurate diagnosis of COVID-19.

1.3 Motivation

The coronavirus pandemic has led to 9.35 million cases and around 135k deaths till date in India itself. Although the researchers have produced plausible solutions for covid-19 detection, the reverse transcription-polymerase chain reaction (RT-PCR) tests are costly and take 6-9 hours to deliver results. Due to less sensitivity of RT-PCR, it provides high false-negative results which can lead to faster spread on the virus.To accelerate the process of detection of covid-19 in infected patients we propose radiological imaging techniques such as chest X-rays and computed tomography (CT). Why chest X-Rays? We prefer x-rays over CT scans. The reason behind this is that x-rays machines are available in most of the hospitals and also x-rays machines are cheaper than the CT scan machine. Besides this, x-rays have lower ionizing radiations than CT scans.

1.4. Proposed Solution

As a response to combat against the virus through Artificial intelligence. There are many deep learning approaches that we have explored to tackle various problems pertaining to various aspects of the virus such as diagnosis, spread of infection, and estimation of cardiac involvement caused by the virus and drug recommendation.

For image based detection the use of Deep learning technologies, such as convolutional neural networks CNN have proved to be the right option in the medical domain. The reason is that this type of network is significantly capable of nonlinear modeling and has extensive use in medical image processing and diagnosis processes. The Image based approach is ideal due to the fact that it has taken a leap in the medical domain over the past years. Primarily in anatomical/cell structure detection, tissue segmentation, computer-aided disease diagnosis or prognosis, and so on.

General block Diagram of image based classification using CNN:

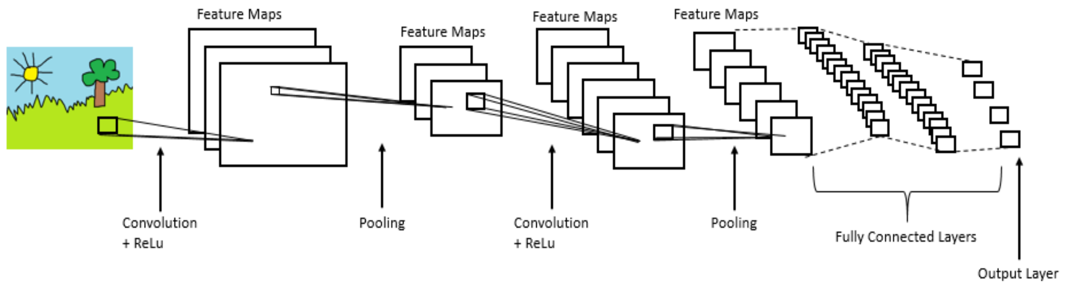


Fig 1.1 Proposed Diagram for Image based classification using CNN

Mainly consists of:

1. Preprocessing
2. Convolutional layers
3. Activation function
4. Fully connected layers
5. Output layer

Preprocessing

The input images are processed according to the requirements of the model in use. For, Example some algorithms require input images of specific size like 220*220 pixels. Removing noise from the images (denoise). We can achieve this using Gaussian Blur.

Segmentation separating the background from foreground objects and we are going to further improve our segmentation with more noise removal. There many other preprocessing techniques that can be explored.

Convolutional layers

These layers are connected layers that help in formation of feature maps with the help of filters for each feature. These filters act as patches which can be placed on images to detect low-level-features such as edges. More number of layers can be used to detect high-level-features such as car, cat, dog, etc. This is passed through a nonlinear activation function mainly ReLu (Rectified Linear Unit) which rectifies non-linearity as the single most important factor in improving the performance.

Pooling Layers

These Layers progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The most common approach used in pooling is max pooling.

Fully connected Network Layers

This fully connected layer is used for classification of the image based on the feature maps from the previous layers. This gives us the probability distribution of the classification.

Output Layer

Softmax layer extends the idea of binary classes into a multi-class world. That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would.

1.5 Scope of the Project

Along with the advancements in Artificial intelligence (AI) we wish to create a web application that can detect positive covid cases based on image processing techniques using images of the lungs. Our goal is to achieve better accuracy than the present models and make it cost and time efficient. We also want to identify the severity of the patient based on the regions of the lungs affected by things like Ground-Glass opacities (GGO), Traction bronchiectasis, Air space consolidation and other pertinent features that are important to analyze. This will help us decide the final output of our project using deep learning techniques.

Chapter 2
Literature Review

Ref.	Year	Introduction	Dataset used	Methodology	Advantages	Disadvantages
[1]	May 2020	A deep learning method that could extract COVID-19's graphical features so as to give an analysis, a machine learning classification technique used to classify the Chest X-Ray images.	https://github.com/ieee8023/covid-chestxray-dataset	DCNN algorithm is executed with TensorFlow and Inception V3.	TensorFlow bundles together ML and deep learning models/algorithms and makes them useful by way of a common metaphor	1.CNN does not encode the position and orientation of the object into their predictions.
[2]	June 2020	Proposed an explainable method aimed to automatically detect the areas of interest in the chest X-ray, symptomatic of the COVID-19 disease.	COVID-19 image data collection.	The method is aimed to mark as first step a chest X-ray as related to a healthy patient or to a patient with pulmonary diseases,	.VGG-16 is a convolutional neural network with 16 layers. 92.7% test accuracy on ImageNet	1. VGG-16 It is very slow to train. 2.Transfer learning utilizes a pre-trained neural network.
[3]	Aug 2020	CVDNet , Deep CNN model to classify COVID-19 infection from normal and other pneumonia cases using chest x-rays.	Dataset used- Kaggle's COVID-19 radiography database	Reviewed certain number of models designed to identify COVID-19 infection and in turn proposed a deep CNN model,	The ResNet50 model is most widely used in image classification as it uses a technique called skip connections- skips training from a few layers and connects directly to the output.	1.CNN does not encode the position and orientation of the object into their predictions.
[4]	April 2020	COVID-Net, as the baseline, and comparing the results achieved in the proposed model with theirs.	covid-chestxray-dataset for COVID-19 frontal-view chest X-Ray images	CovidAID, A PyTorch (python3) based implementation, to identify COVID-19 cases from X-Ray images.	CheXNet is a 121-layer CNN trained on ChestX-Ray14	1.the labels dont match the images . 2.the dataset has questionable clinical relevance.
[5]	Sept 2020		The database is constructed	3 different deep learning	SVM on DCNN features is slightly	The size of the publicly

			using 180 covid 19 samples and 200(nor mal) healthy samples	approaches for covid detection	better than the results of neural network classification on DCNN extracted features.	available dataset is small
[6]	July 2020	Initially a convolutional layer is applied by using different masks to produce low level features. activation function used for this is ReLU	A COVID-19 X-ray image database was developed by Cohen JP.	cnn , VGGnet , ResNet50,alexnet ,googlenet , inception V3 and the proposed model.	Faster than real-time polymerase chain reaction.	The size of the publicly available dataset is small
[7]	June 2020	proposed a Convolutional Neural Network model to automatically detect COVID patients.	chest X-ray images of 341 COVID-19 patients obtained from GitHub Repository Shared by Dr. Joseph Cohen.	three different layers that are a convolutional layer, pooling layer, and fully connected layer are used	no manual feature extraction, feature selection and classification in this method.	Transfer learning utilizes a pre-trained neural network.

Chapter 3
System Analysis

This chapter gives an idea about the various functional and non-functional requirements of the proposed system. It also gives an overview of the hardware and software requirements as well as the use case diagram of the system.

3.1 Functional Requirements

Functional Requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

- The system must perform image acquisition in which the image must be a chest xray taken from the patient.
- Before testing the image sample of a patient, all the details of patients should be entered.
- Preprocessing of images as per the image quality includes removal of noise, filtering mechanisms, morphological transformation, image enhancement, etc.
- The system has to extract the features and intensity.
- The system should be connected with a good internet.

3.2 Non-Functional Requirements

Non-Functional Requirements will describe how a system should behave and what limits are constrained on its functionality. It generally specifies the system’s quality attributes or characteristics.

I. **Performance Evaluation Parameters**:

The necessary condition for a medical diagnosis system is its performance. To evaluate the performance of the entire system, the end result of the classification is considered. For the evaluation of the performance, we use the concepts used in binary classification such as accuracy, precision, recall, F1 measure.

A. Accuracy: It is the ratio of the number of correct predictions to the total number of input samples. This is shown in Equation (1).

$$\text{Accuracy}= \frac{TP + TN}{TP+TN+FP+FN} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \text{(1)}$$

B. Precision: It is the number of correct positive results divided by the number of positive results predicted by the classifier. This is shown in Equation (2).

$$\text{Precision}= \frac{TP}{TP + FP} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2)}$$

C. Recall: It is the number of correct positive results divided by the number of all relevant samples. This is shown in Equation (3)

$$\text{Recall} = \frac{TP}{TP + FP} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3)}$$

D.F1 measure: It is the harmonic mean between precision and recall. It tells you how precise your classifier is, as well as how robust it is. This is shown in Equation (4)

F1 measure = frac2 * recall * precisionrecall + precision …………………(4)

II. Other parameters:

- Usability: The system must be easy to learn for a new user.

- Portability: The system must be able to run on any operating system.
- Resource utilization: The system should use less resources as possible thereby making it accessible to as many clinics as possible.
- Availability: The system should be available to any radiologist .
- Correctness: The accuracy of the system should be as maximum as possible for better prediction.
- Maintainability: The system should maintain correct history of records.

3.3 Specific Requirements

3.3.1 Hardware Requirements

- Minimum 8GB RAM. Higher Capability for training model.
- 10GB free space on secondary storage.
- CPU- QuadCore Intel Core i7 SkyLake
- GPU- Nvidia GTX710 2G

3.3.2 Software Requirements

- Python3/MATLAB
- Operating system -Windows/Linux
- Web Browser

3.4 Use Case Diagram:

This Use Case diagram gives an idea about the behavior of the system with respect to various actions performed by the proposed system. It includes the various actors and use cases involved in the system as well as the relationships between them.

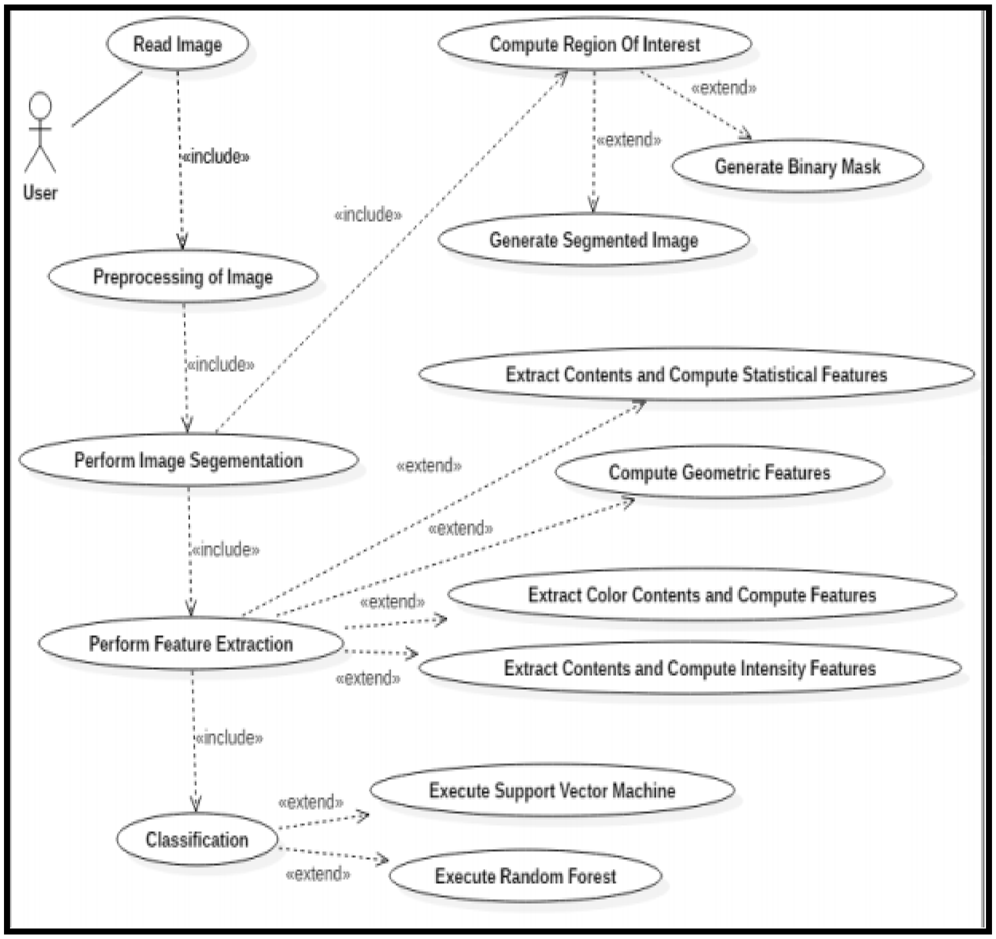

Read Image
User
«include»
Preprocessing of Image
«include»
Perform Image Segementation
«include»
Perform Feature Extraction
«include»
Classification
«include»
Compute Region Of Interest
«extend»
«extend»
Generate Binary Mask
Generate Segmented Image
Extract Contents and Compute Statistical Features
«extend»
Compute Geometric Features
«extend»
«extend»
Extract Color Contents and Compute Features
Extract Contents and Compute Intensity Features
«extend»
«extend»
Execute Support Vector Machine
«extend»
Execute Random Forest

Fig 3.1 Use case diagram for RL system

Actors: User

Use Cases: Input Image, Image Processing, Image Segmentation, Feature Extraction, Classification, etc.

Chapter 4
Analysis Modeling

4.1 Activity Diagram

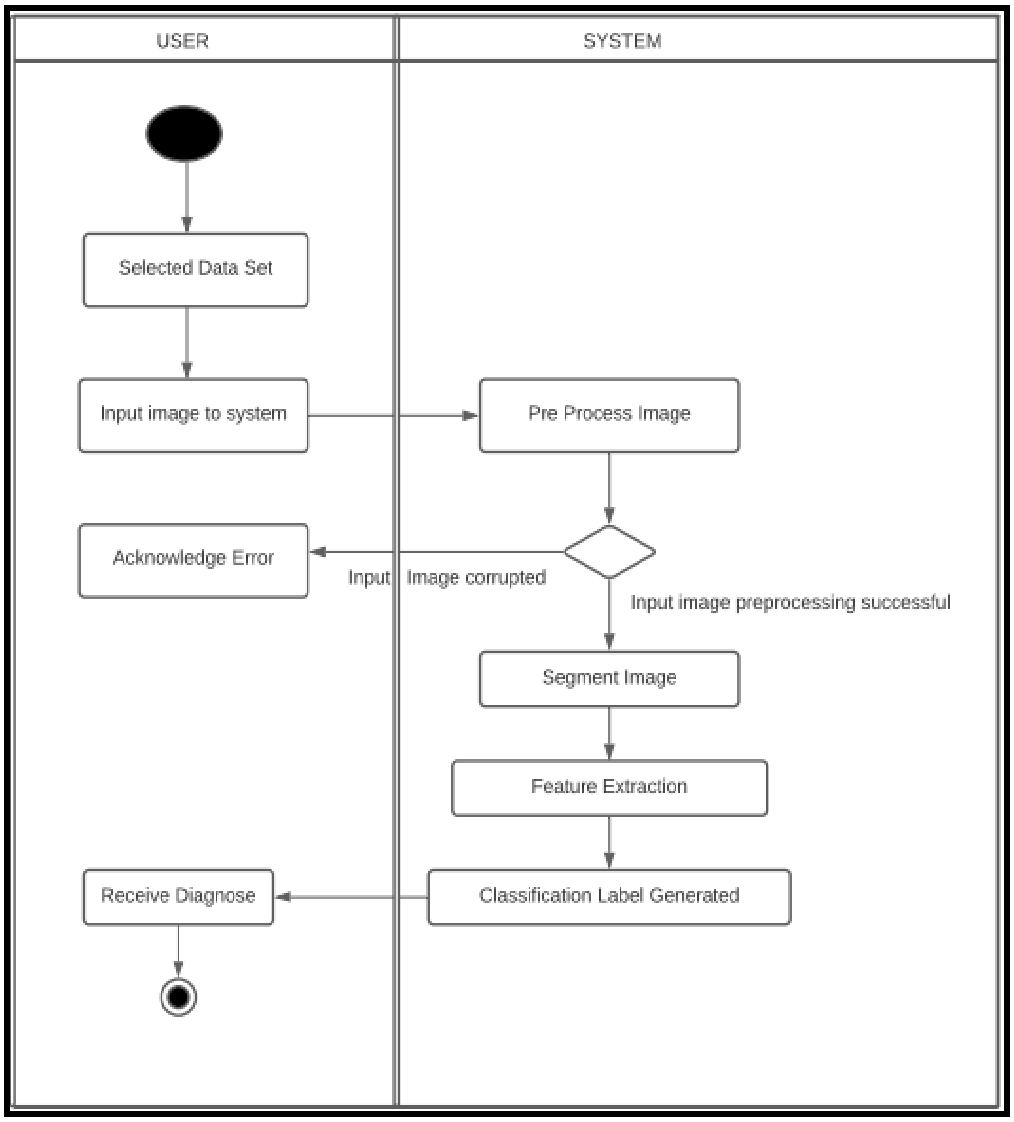


Fig. 4.1 Activity Diagram of Covid-19 Detection using Chest X-Rays

This is our Activity Diagram that represents a flow from one activity to another. Our activity begins with capturing all abnormalities in chest X-rays. The image is taken as input, preprocessed and checked if the image analysis is complete. It then goes for segmentation, followed by feature extraction. System generates the classification label and is checked, and in this way, proper diagnosis is done.

4.2 Functional Modeling:

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on flow of information, where data comes from, where it goes and how it gets stored.

Context Level/ Level 0 DFD:

A context level data flow diagram (DFD) provides an at-a-glance look at an information system and the ways it exchanges data with outside entities. They are often used for high-level planning. The following figure is the Context Level DFD of the Classification System.

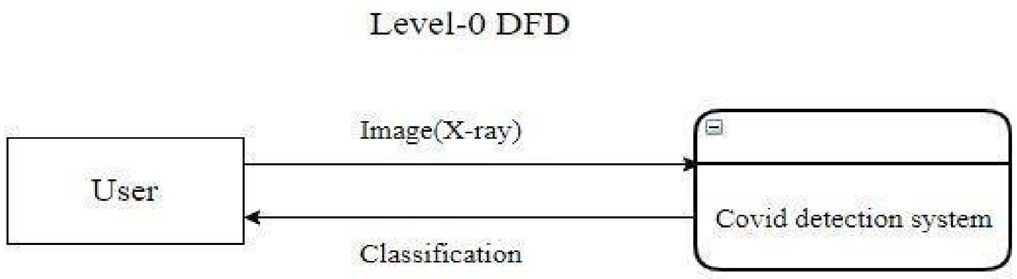


Fig.4.2 Context Level DFD Diagram of the Covid-19/Pneumonia Detection System

Level 1 DFD:

A level 1 data flow diagram (DFD) is more detailed than a level 0 DFD but not as detailed as a level 2 DFD. It breaks down the main processes and subprocesses that can then be analyzed and improved on a more intimate level. The following figure is the Level 1 DFD of the Classification System.

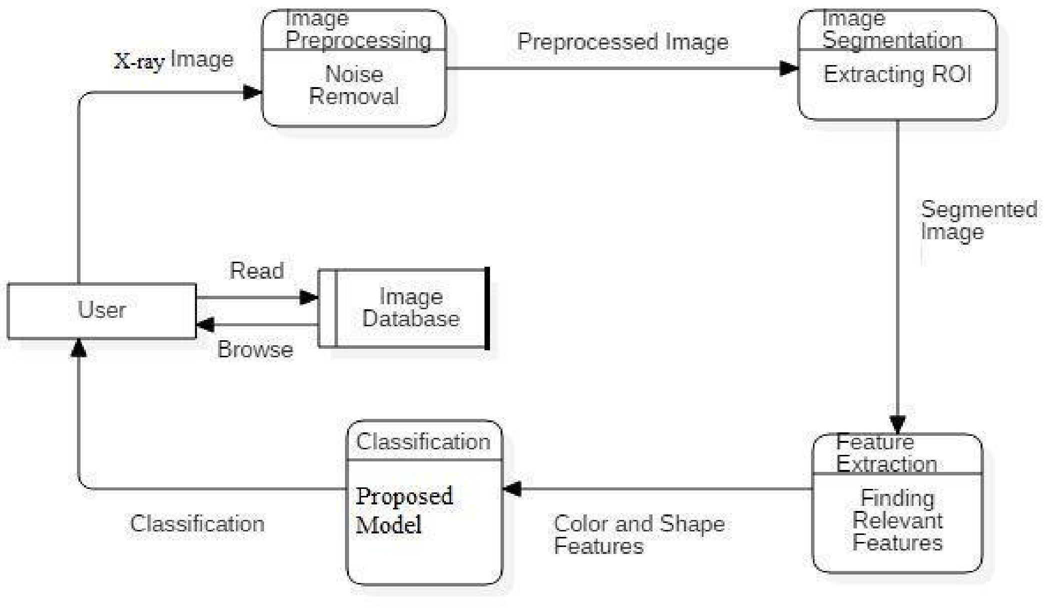


Fig.4.3 Context Level DFD Diagram of the Covid-19/Pneumonia Detection System

Chapter 5

Design

5.1 Architectural Design

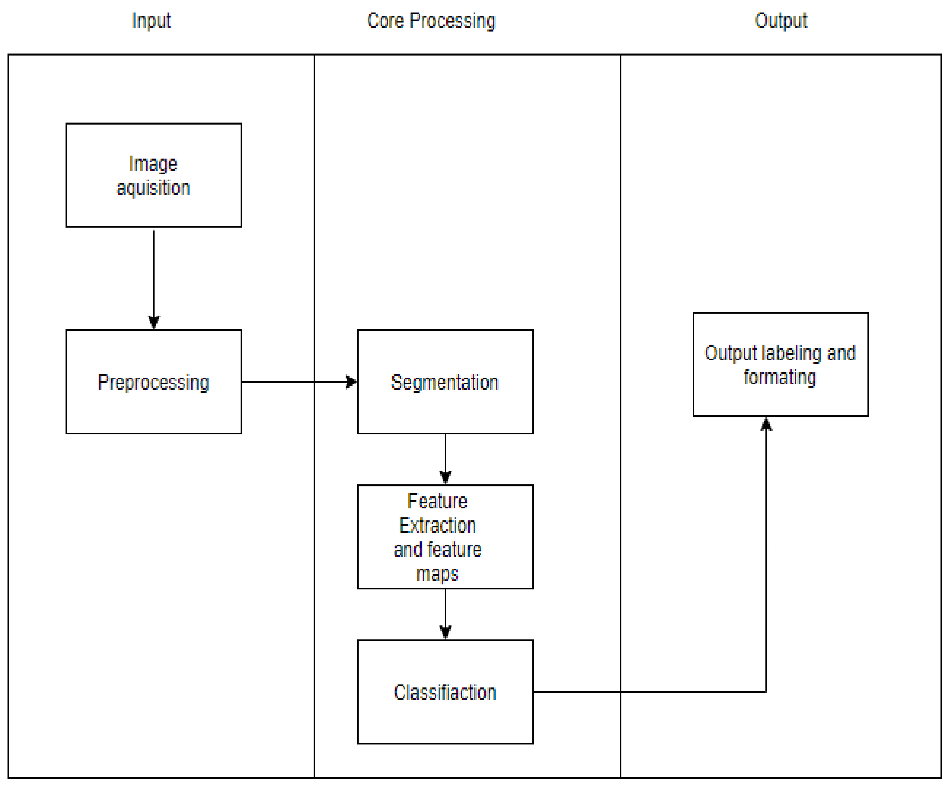


Fig.5.1 Architecture of Covid-19/Pneumonia detection system from Chest X-rays

In this project, we have proposed an automatic method of detecting and classification of Covid-19 using image processing and machine learning techniques. The fundamental aim of this project is to build a system for detecting covid-19 patients from normal ones using X-ray images. This system will help to detect covid-19 patients from X-ray samples images. The image processing techniques that will be used are image pre-processing, segmentation, image enhancement, feature extraction etc. in order to enhance image contents and extract the identifiable features.

5.2 Algorithmic View

5.2.1 Feature Extraction

A CNN is composed of two basic parts of feature extraction and classification. Feature extraction includes several convolution layers followed by max-pooling and an activation function. The classifier usually consists of fully connected layers.

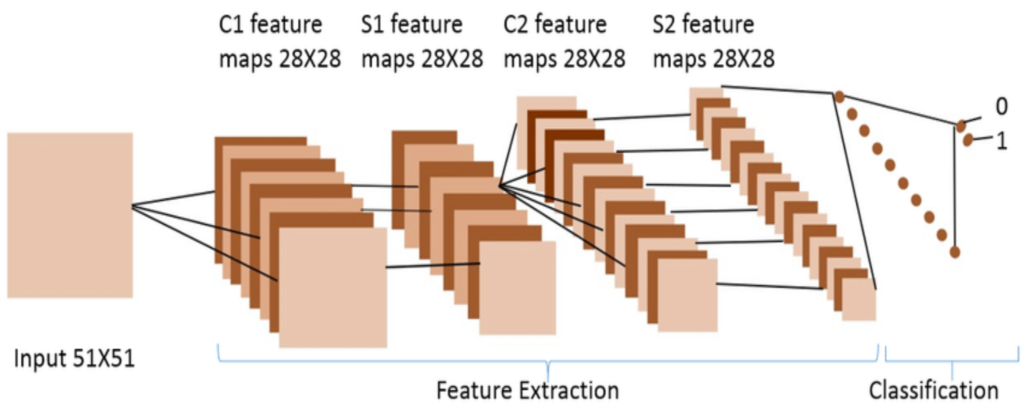
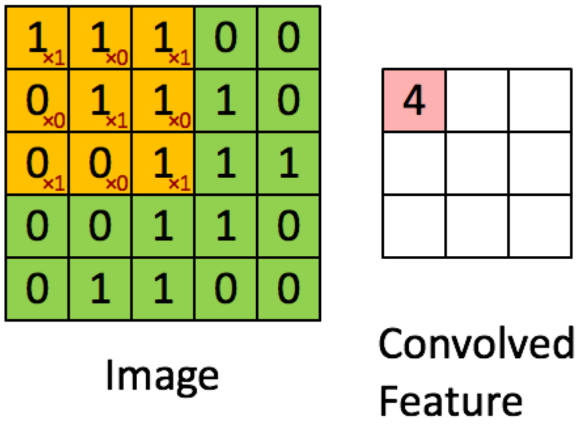


Fig. 5.2 Feature extraction in CNN

Natural images have the property of being" 'stationary"', meaning that the statistics of one part of the image are the same as any other part. This suggests that the features that we learn at one part of the image can also be applied to other parts of the image, and we can use the same features at all locations.

More precisely, having learned features over small (say 8x8) patches sampled randomly from the larger image, we can then apply this learned 8x8 feature detector anywhere in the image. Specifically, we can take the learned 8x8 features and" 'convolve" ' them with the larger image, thus obtaining a different feature activation value at each location in the image.

To give a concrete example, suppose you have learned features on 8x8 patches sampled from a 96x96 image. Suppose further this was done with an auto encoder that has 100 hidden units. To get the convolved features, for every 8x8 region of the 96x96 image, that is, the 8x8 regions starting at (1,1), (1,2), … (89,89), you would extract the 8x8 patch, and run it through your trained sparse auto encoder to get the feature activations. This would result in 100 sets 89x89 convolved features.



Formally, given some large $r\times c$ images xlarge, we first train a sparse autoencoder on small $a\times b$ patches xsmall sampled from these images, learning k features $f=\sigma(W(1)xsmall+b(1))$ (where σ is the sigmoid function), given by the weights W(1) and biases b(1) from the visible units to the hidden units. For every $a\times b$ patch xs in the large image, we compute $fs=\sigma(W(1)xs+b(1))$, giving us fconvolved, a $k\times(r-a+1)\times(c-b+1)$ array of convolved features.

Chapter 6

Algorithms

6.1 Algorithm / Methods Used

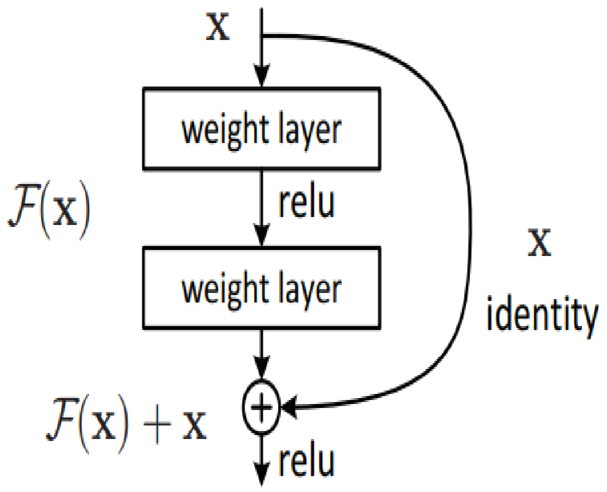
1] Resnet 50

ResNet, which was proposed in 2015 by researchers at Microsoft Research, introduced a new architecture called Residual Network.

In order to solve the problem of the vanishing/exploding gradient, this architecture introduced the concept called Residual Network. In this network we use a technique called ***skip connections***.

The skip connection skips training from a few layers and connects directly to the output.

The approach behind this network is instead of layers learn the underlying mapping; we allow network fit the residual mapping. So, instead of say H(x), initial mapping, let the network fit, $F(x) := H(x) - x$ which gives $H(x) := F(x) + x$.



The advantage of adding this type of skip connection is because if any layer hurts the performance of architecture then it will be skipped by regularization. So, this results in training a very deep neural network without the problems caused by vanishing/exploding gradients. The authors of the paper experimented on 100-1000 layers on the CIFAR-10 dataset.

There is a similar approach called "highway networks", these networks also use skip connection. Similar to LSTM these skip connections also use parametric gates. These gates determine how much information passes through the skip connection. This architecture however has not provided accuracy better than ResNet architecture.

Network Architecture:

This network uses a 34-layer plain network architecture inspired by VGG-19 in which then the shortcut connection is added. These shortcut connections then convert the architecture into a residual network.

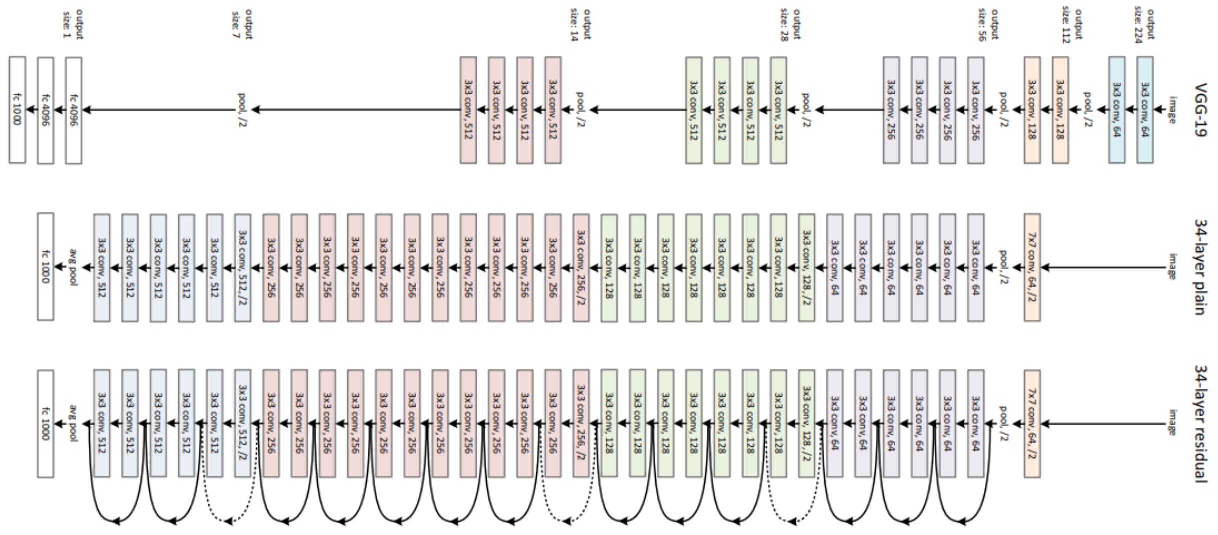


Fig.6.1 Architecture of Resnet-50

Implementation:

Using the Tensorflow and Keras API, we can design ResNet architecture (including Residual Blocks) from scratch. Below is the implementation of different ResNet architecture. For this implementation we use the CIFAR-10 dataset. This dataset contains 60, 000 32×32 color images in 10 different classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks) etc. These datasets can be assessed from k*eras.datasets* API function.

- First, we import the keras module and its APIs. These APIs help in building architecture for the ResNet model.

```
# Import Keras modules and its important APIs
import keras
from keras.layers import Dense, Conv2D, BatchNormalization, Activation
from keras.layers import AveragePooling2D, Input, Flatten
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from keras.regularizers import l2
from keras import backend as K
from keras.models import Model
from keras.datasets import cifar10
import numpy as np
import os
```

- Now, We set different hyper parameters that are required for ResNet architecture. We also preprocessed our datasets to prepare it for training.
- In this step, we set the learning rate according to the number of epochs. As the number of epochs the learning rate must be decreased to ensure better learning.
- Then we train and test the architecture.

	Plain	ResNet
18 Layers	27.94	27.88
34 Layers	28.54	**25.03**

- The result above shows that shortcut connections would be able to solve the problem caused by increasing the layers because as we increase layers from 18 to 34 the error rate on ImageNet Validation Set also decreases unlike the plain network.

Model	Top-1 err.	Top-5 err.
VGG-16	28.07	9.33
GoogLeNet	-	9.15
PReLU-net	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-24 C	24.19	7.40
ResNet-50	**22.85**	**6.71**
ResNet-101	21.75	6.05
ResNet-152	21.43	5.71

- Below are the results on the ImageNet Test Set. The *3.57%* top-5 error rate of ResNet was the lowest and thus ResNet architecture came first in the ImageNet classification challenge in 2015.

method	Top-5 err.(test)
VGG	7.32
GoogLeNet	6.66
VGG (v5)	6.8
PReLU-net	4.94
BN-inception	4.82
ResNet	**3.57**

References:

- ResNet paper.
- Keras ResNet Implementation**.**

2] CheXnet:

CheXnet is a 121-layer convolutional neural network that inputs a chest X-ray image and outputs the probability of pneumonia along with a heatmap localizing the areas of the image most indicative of pneumonia.

CheXNet is trained on the recently released ChestX-ray14 dataset, which contains 112,120 frontal-view chest X-ray images individually labeled with up to 14 different thoracic diseases, including pneumonia. We use dense connections and batch normalization to make the optimization of such a deep network tractable.
The dataset(ChestX-ray14) is questionable. So, we trained using our own dataset.

The dataset, released by the NIH, contains 112,120 frontal-view X-ray images of 30,805 unique patients, annotated with up to 14 different thoracic pathology labels using NLP methods on radiology reports. We label images that have pneumonia as one of the annotated pathologies as positive examples and label all other images as negative examples for the pneumonia detection task.

We collected a test set of 420 frontal chest X-rays. Annotations were obtained independently from four practicing radiologists at Stanford University, who were asked to label all 14 pathologies. We then evaluate the performance of an individual radiologist by using the majority vote of the other 3 radiologists as ground truth. Similarly, we evaluate CheXNet using the majority vote of 3 of 4 radiologists, repeated four times to cover all groups of 3.

Architecture

We compared the performances of different classic neural networks including DenseNet121, DenseNet169 and ResNet50. Additionally, CheXNet paper also mentioned that they pre-trained the DenseNet121 on ImageNet dataset. Referring to the architecture of ResNet50 and SE- ResNet50 shown in *Squeeze-and-Excitation Networks*. We added the SE modules into the Dense blocks in DenseNet121, the architecture is described in table

Figure 4 shows the schema of the original Inception module (left) and the SE-Inception module (right). The purpose of SE block is to weigh the importance of the channel of filters so that the network will enhance more useful channels and ignore the less useful ones.

We totally employed 5 neural networks (DenseNet121, SE-DenseNet121, DenseNet169, ResNet50, SE- ResNet50) to solve this problem and CAM(Class activation mapping) to generate heatmap on top of the X-Ray scan to interpret the neural networks' output.

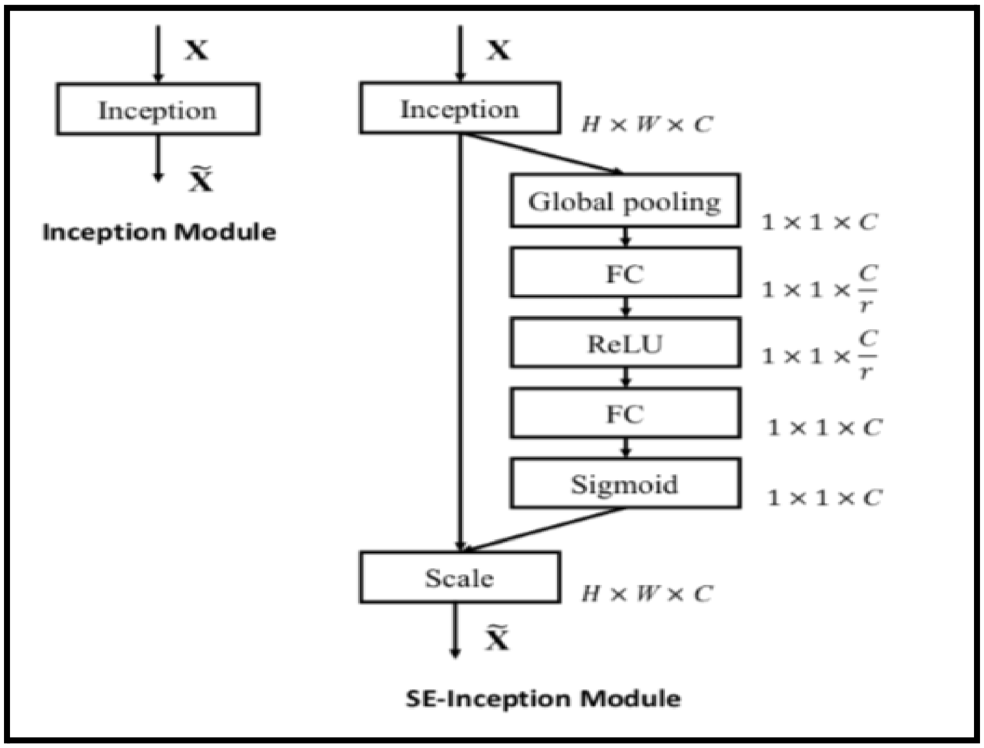


Fig. 6.2 Architecture of CheXnet

References:

- CheXnet Paper
- CheXnet implementation github.

3] SVM (Support Vector Machines) Classifiers:

A support vector machine (SVM) is a type of deep learning algorithm that performs supervised learning for classification or regression of data groups.

In AI and machine learning, supervised learning systems provide both input and desired output data, which are labeled for classification. The classification provides a learning basis for future data processing. Support vector machines are used to sort two data groups by like classification. The algorithms draw lines (hyperplanes) to separate the groups according to patterns.

An SVM builds a learning model that assigns new examples to one group or another. By these functions, SVMs are called a non-probabilistic, binary linear classifier. In probabilistic classification settings, SVMs can use methods such as Platt Scaling.

Like other supervised learning machines, an SVM requires labeled data to be trained. Groups of materials are labeled for classification. Training materials for SVMs are classified separately in different points in space and organized into clearly separated groups. After processing numerous training examples, SVMs can perform unsupervised learning. The algorithms will try to achieve the best separation of data with the boundary around the hyperplane being maximized and even between both sides.

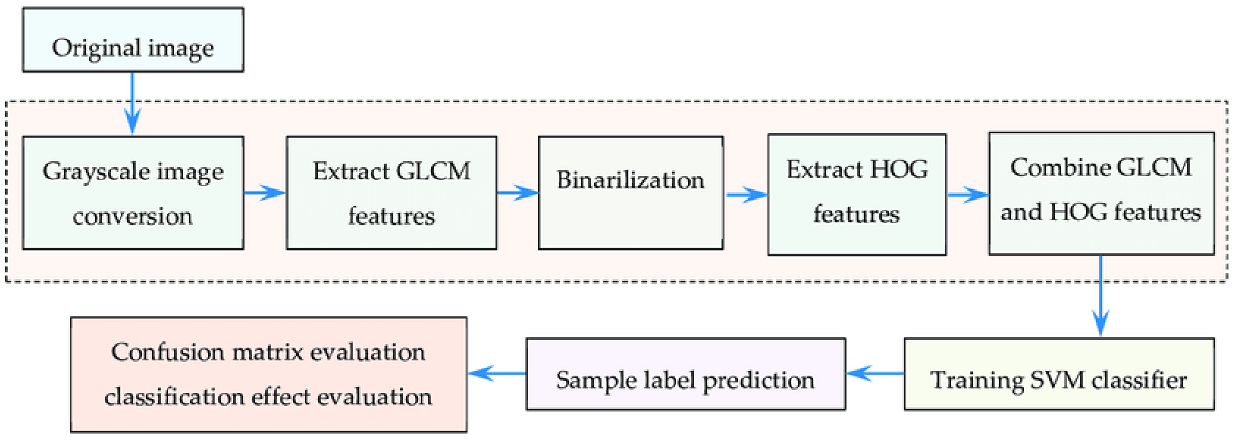

Original image
Grayscale image conversion
Extract GLCM features
Binarilization
Extract HOG features
Combine GLCM and HOG features
Training SVM classifier
Sample label prediction
Confusion matrix evaluation classification effect evaluation

Fig.6.3 Working of Support Vector Machine(SVM) Classifier

References:

- SVM- ResearchGate
- SVM for image classification ieee papers

4] Random Forest

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Rrandom forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:

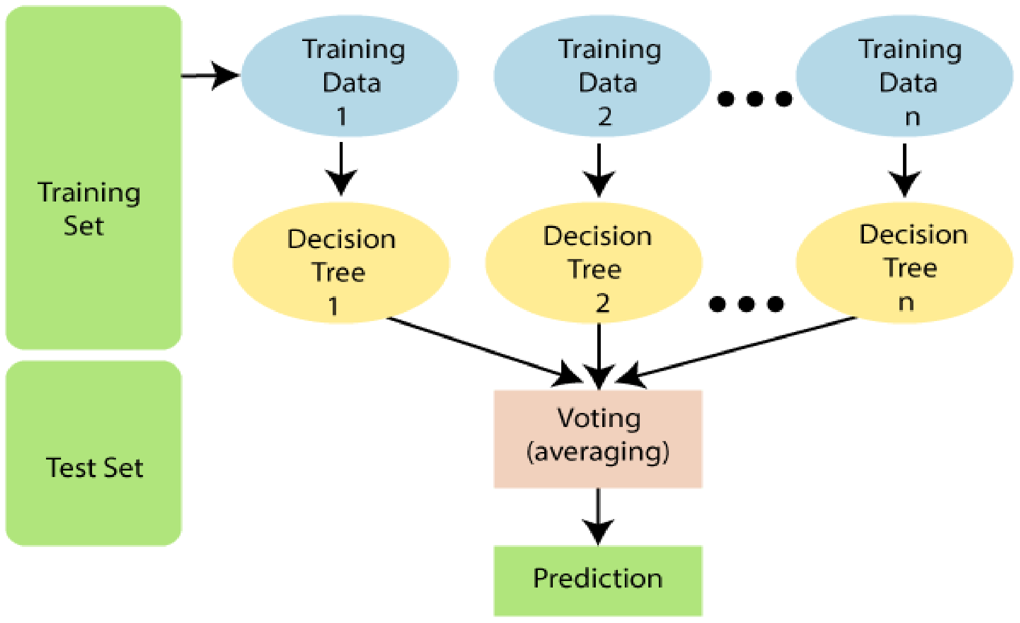


Fig.6.4: Working of Random Forest Classifier

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

5] Logistics Regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted to a particular college.

The resulting analytical model can take into consideration multiple input criteria. In the case of college acceptance, the model could consider factors such as the student’s grade point average, SAT score and number of extracurricular activities. Based on historical data about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into a particular outcome category.

6.2 Working of Project

1] Resnet 50

Code: covidCNN.py

```
import torch

import torchvision

from torch.utils.data import DataLoader,random_split

from torchvision.datasets import ImageFolder

from sklearn.metrics import confusion_matrix

# Set device

device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")

print(device)

#save_checkpoint

#def save_checkpoint(state,filename='my_checkpoint.pth.tar'):

#print('=> saving checkPoint')

#torch.save(state,filename)

#load_checkpoint

#def load_checkpoint(checkpoint):

#print('=>loading checkpoint')

#model.load_state_dict(checkpoint['state_dict'])

#optimizer.load_state_dict(checkpoint['optimizer'])
```

```
# load data

root_dir = './data'

train_transform = torchvision.transforms.Compose([

torchvision.transforms.Resize(size=(224,224)),

torchvision.transforms.RandomHorizontalFlip(),

torchvision.transforms.ToTensor(),

torchvision.transforms.Normalize(mean = [0.485, 0.456, 0.406],

std = [0.229, 0.224, 0.225]) ])

test_transform=torchvision.transforms.Compose([torchvision.tra
nsforms.Resize(size=(224,224)),

torchvision.transforms.ToTensor(),

torchvision.transforms.Normalize(mean = [0.485, 0.456,
0.406],std = [0.229, 0.224, 0.225]) ])

train_dataset = ImageFolder('./data/train',train_transform)

test_dataset = ImageFolder('./data/test',train_transform)

test_data = 50

val_data= len(test_dataset)-test_data

val_ds,test_ds =random_split(test_dataset,[val_data,test_data])

train_dl = DataLoader(train_dataset, batch_size=6,
shuffle=True)

val_dl = DataLoader(val_ds, batch_size=6, shuffle=True)
```

```
test_dl = DataLoader(test_ds, batch_size=1, shuffle=True)

# create model

load_model = False

batch_size = 6

num_epochs = 3

class_names = train_dataset.classes

model = torchvision.models.resnet18(pretrained=True)

model.cuda()

model.fc = torch.nn.Linear(in_features=512, out_features=3)

model.fc.cuda()

loss_fn = torch.nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=3e-5)

#if load_model:

#load_checkpoint(torch.load('my_checkpoint.pth.tar'))

for epoch in range(num_epochs):

losses =[]

#if epoch % 3 == 0:


#checkpoint={'state_dict':model.state_dict(),'optimizer':optimize
r.state_dict()}

# save_checkpoint(checkpoint)
```

```
for batch_idx, (data, targets) in enumerate(train_dl):

# Get data to cuda if possible

data = data.to(device)

targets = targets.to(device)

# forward

scores = model(data)

loss = loss_fn(scores, targets)

losses.append(loss.item())

# backward

optimizer.zero_grad()

loss.backward()

#gradient descent or adam step

optimizer.step()

mean_loss = sum(losses)/len(losses)

print(f'loss at each epoch {epoch} was {mean_loss:.3f}')

#Check accuracy on training & test to see how good our model

def check_accuracy(loader, model):

if train_dl:

print("Checking accuracy on training data")

else:
```

```
print("Checking accuracy on test data")

num_correct = 0

num_samples = 0

model.eval()

with torch.no_grad():

for x, y in loader:

x = x.to(device=device)

y = y.to(device=device)

scores = model(x) _, predictions = scores.max(1)

num_correct += (predictions == y).sum()

num_samples += predictions.size(0)

print(f"Got {num_correct} / {num_samples} with accuracy
{float(num_correct)/float(num_samples)*100:.2f}")

model.train()

check_accuracy(train_dl, model)

check_accuracy(val_dl, model)
```

2] ChexNet

Code: main.py

```
import numpy as np # linear algebra

import pandas as pd # data processing,

import os
import zipfile

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.image import imread

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPool2D,
Flatten, Dense, Dropout

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.preprocessing import image

data_dir = "C:\\input\\Data"

os.listdir(data_dir)

train_dir = data_dir+"/train"

os.listdir(train_dir)

test_dir = data_dir+"/test"

os.listdir(test_dir)

print(len(os.listdir(train_dir+"/COVID19")))

print(len(os.listdir(train_dir+"/NORMAL")))

print(len(os.listdir(train_dir+"/PNEUMONIA")))

print(len(os.listdir(test_dir+"/COVID19")))

print(len(os.listdir(test_dir+"/NORMAL")))
print(len(os.listdir(test_dir+"/PNEUMONIA")))

pneumonia_sample=imread(train_dir+"/PNEUMONIA"+"/PNEUMONIA(3188).jpg")

plt.imshow(pneumonia_sample)

normal_sample = imread(train_dir+"/NORMAL"+"/NORMAL(342).jpg")

plt.imshow(normal_sample)

covid_sample = imread(train_dir+"/COVID19"+"/COVID19(189).jpg")

plt.imshow(covid_sample, cmap= "gray")

image_size = (400,400,3)

generator = ImageDataGenerator(

rotation_range=0,

width_shift_range=0.1,

height_shift_range=0.1,

shear_range=0.1,

zoom_range=0.2,

fill_mode='nearest',

horizontal_flip=False,

vertical_flip=False,

rescale=1/255,)

train_generator = generator.flow_from_directory (

train_dir,

target_size=image_size[:2],

color_mode='rgb',
class_mode='categorical',

batch_size=32,

shuffle=True,)

test_generator = generator.flow_from_directory (

test_dir,

target_size=image_size[:2],

color_mode='rgb',

class_mode='categorical',

batch_size=32,

shuffle=False,)

stop = EarlyStopping(monitor="val_loss", mode="min", patience=6)

#creating model

model = Sequential()

model.add(Conv2D(filters = 32, padding = "same", kernel_size = (2,2), strides = (2,2), activation = "relu", input_shape = image_size))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 32, padding = "same", kernel_size = (2,2), strides = (2,2), activation = "relu", input_shape = image_size))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 64, padding = "same", kernel_size = (2,2), strides = (2,2), activation = "relu", input_shape =image_size))

model.add(MaxPool2D(pool_size = (2,2)))

model.add(Flatten())
model.add(Dense(units = 132, activation = "relu"))

model.add(Dense(units = 60, activation = "relu"))

model.add(Dense(units = 3, activation = "softmax"))

model.compile(loss="categorical_crossentropy", optimizer =
"adam", metrics = ["accuracy"])

#fit Model

history=model.fit_generator(train_generator,validation_data=tes
t_generator, epochs=10, callbacks=[stop])

model.summary()

#Visualise training

pd.DataFrame(model.history.history)[["loss",
"val_loss"]].plot(figsize =(16,6), marker = "o", mfc = "g")

pd.DataFrame(model.history.history)[["accuracy",
"val_accuracy"]].plot(figsize =(16,6), marker = "o", mfc = "g")

#predict

predictions = model.predict(test_generator)

predictions

pred_labels = np.argmax(predictions, axis = 1)

#Random Testing

from random import randint

random_index = randint(1,
len(os.listdir(train_dir+"/COVID19")))

random_covid_image_name =
os.listdir(train_dir+"/COVID19")[random_index]

random_covid_image_name

covid_img_path =
train_dir+"/COVID19/"+random_covid_image_name
```

```
        random_covid_img=image.load_img(covid_img_path,
        target_size=image_size)

        random_covid_img

        random_covid_img_array                                        =
image.img_to_array(random_covid_img)

        covid_img_array = np.expand_dims(random_covid_img_array,
axis=0)

        covid_img_array.shape

        np.argmax(model.predict(covid_img_array), axis =1)

        test_generator.class_indices

        #Evaluation

        from       sklearn.metrics       import       confusion_matrix,
classification_report

        print(classification_report(test_generator.classes, pred_labels))
```

3] Bins

Code:

```
import numpy as np
from matplotlib import pyplot as plt
import torchvision.transforms as transforms
import torchvision
import torch
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
from numpy import asarray
import math
def cube_root(x):
return x**(1/3.0)
def fourth_root(x):
return x**(1/4.0)
def std(data):
n = len(data)
if n <= 1:
return 0.0
mean, sd = Mean(data), 0.0
# calculate stan. dev.
for el in data:
```

```
sd += (float(el) - mean)**2
sd = math.sqrt(sd / float(n-1))
return sd
def Mean(ls):
n, mean = len(ls), 0.0
if n < 1:
return 0.0;
# calculate average
for el in ls:
mean = mean + float(el)
mean = mean / float(n)
return mean
def skew(data):
n = len(data)
if n <= 1:
return 0.0
mean, sd = Mean(data), 0.0
# calculate stan. dev.
for el in data:
sd += (float(el) - mean)**2
skew = cube_root(sd / float(n))
return skew
def kurto(data):
n = len(data)
if n <= 1:
return 0.0
mean, sd = Mean(data), 0.0
# calculate stan. dev.
for el in data:
sd += (float(el) - mean)**4
sd = fourth_root(sd / float(n))
return sd
data_transforms = transforms.Compose([
transforms.Grayscale(num_output_channels=1)])
ds = torch.utils.data.ConcatDataset([ImageFolder(root='./data/train',
transform=data_transforms),
ImageFolder(root='./data/test',
transform=data_transforms)])
kvals = []
mvals = []
stdvals=[]
svals =[]
bins=[0,32,64,96,128,160,192,224,256]
cnt =0
for img,label in ds:
img = asarray(img)
```

```
data = img.ravel()
digitized = np.digitize(data, bins)
bin_means = [Mean(data[digitized == i]) for i in range(1, len(bins))]
bin_std = [std(data[digitized == i]) for i in range(1, len(bins))]
bin_skew = [skew(data[digitized == i]) for i in range(1, len(bins))]
bin_kurtosis = [kurto(data[digitized == i]) for i in range(1, len(bins))]
# print(bin_means)
# print(bin_std)
# print(bin_skew)
# print(bin_kurtosis)
# plt.hist(img.ravel()*255,bins,(0,256),ec='red')
# plt.xticks(bins)
# adjusted= img.ravel()
# data = adjusted*255
# for i in range(0,8):
# if(str(bin_means[i]) == 'nan'):
# bin_means[i]= 0.0
# if(str(bin_skew[i]) == 'nan'):
# bin_skew[i]= 0.0
# if(str(bin_std[i]) == 'nan'):
# bin_std[i]= 0.0
# if(str(bin_kurtosis[i]) == 'nan'):
# bin_kurtosis[i]= 0.0
bin_means.append(label)
bin_std.append(label)
bin_skew.append(label)
bin_kurtosis.append(label)
mvals.append(bin_means)
kvals.append(bin_kurtosis)
stdvals.append(bin_std)
svals.append(bin_skew)
#     print(mvals)
#     break;
#     print('bin_means: {}'.format(bin_means))
#     print('bin standard deviation: {}'.format(bin_std))
#     print('bin skewess: {}'.format(bin_skew))
#     print('bin Kurtosis: {}'.format(bin_kurtosis))
# saving to files
np.savetxt('MFVDB1.csv', mvals, delimiter=',',fmt='%3.4f')
np.savetxt('STDFVDB1.csv', stdvals, delimiter=',',fmt='%3.4f')
np.savetxt('SFVDB1.csv', svals, delimiter=',',fmt='%3.4f')
np.savetxt('KFVDB1.csv', kvals, delimiter=',',fmt='%3.4f')
```

Chapter 7
Result and Discussions

We have assessed and compared the approaches using following parameters Accuracy, Precision, Recall, and F1- score, and we have correspondingly found the results of the outputs are: From above Table, we can observe that Resnet50 works as considerably as good as CheXnet and bins approach, with accuracy scaling at 96% a result, we can infer that Resnet50 works fairly well with our classification strategy, but slightly lesser than CheXnet approach. The above table also describe that Resnet50 is the best classifier for this approach, with an accuracy of 96%, which is significantly higher than the other methods cited so far (the results are highlighted in yellow). Furthermore,the SVM classifier has the lowest accuracy. It should be observed that in Table, we although models such as CheXnet achieved high precision, there are a few disadvantages to using this approach. For starters, it takes a long time to train the model. We trained the model on basic Windows 10 PC for about 60 to 100 minutes during the process. Although processing times may vary between PCs, loading even a simple model and performing the required com- putations is more costly.The derived results are summarized in below Tables:

TABLE I: COMPARING DEEP LEARNING MODEL

Approach Parameter	CheXnet	Resnet
Accuracy	93%	96%
Precision	92%	96%
Recall	93%	96%
F1-Score	92%	93%

From table 1, Comparing the deep learning models we can clearly see that the accuracy of Resnet50 is more than CheXnet in terms of all parameters.

TABLE II: COMPARING PERFORMANCE OF DIFFERENT CLASSIFIERS FOR PLANE- BINS APPROACH

Approach Parameter	Bins with Logistic Regression	Bins with SVM	Bins with Random Forest
Accuracy	69%	79%	82%
Precision	68%	79%	82%
Recall	70%	76%	81%
F1-Score	67%	78%	81%

We can see from the below table 2 and 3 that results obtained by linear grouping-based bins approach getting highest accuracy i.e. 82% for random forest classifier. So we further try to implement to bins using polynomial function in order to improve the image contents with the hope of getting better results.

TABLE III: COMPARING PERFORMANCE OF DIFFERENT CLASSIFIERS FOR BINS APPROACH WITH POLYNOMIAL FUNCTION.

Approach Parameter	Bins with Logistic Regression	Bins with SVM	Bins with Random Forest
Accuracy	70%	76%	81%
Precision	69%	76%	80%
Recall	70%	76%	81%
F1-Score	67%	76%	80%

We can observe in table 3 that even after enhancing the contents using polynomial function, we could not achieve much improvement in the results. It is nearly same for both the variations.
Table 4 shows the overall comparison of bins approach with existing approaches i.e deep learning models. These results are clearly showing that CheXnet and Resnet are giving much better accuracy (93%, 96% respectively) as compared to bins approach with random forest classifier (82%) for Covid 19 detection and classification. In bins approach, converting the original image into grey image must be losing some useful image information which is impacting the feature vector formation and in turn in the classification accuracy. However if

TABLE IV: RESULT OBTAINED FROM DEEP LEARNING MODEL AND BINS APPROACH WITH ML CLASSIFIERS.

Approach Parameter	CheXnet	ResNet	Bins with SVM	Bins with Random Forest	Bins with Logistic Regression
Accuracy	93%	96%	79%	82.21%	72%
Precision	92%	96%	79%	82%	71%
Recall	93%	96%	79%	82%	72%
F1-score	92%	93%	78%	81%	70%

we analyse the feature engineering process of bins approach its it is a simplified approach and also computationally less complex. Feature vector dimensions have been reduced to great extent by simply performing the linear grouping of the histogram bins.

Chapter 9
Conclusion

In the course of our research, we have studied the various approaches contributing to the medical field of diseases, especially highlighting the repercussions of false detection of Covid-19 that can lead to more spread of the virus. Among all the possible ways to detect and diagnose the virus, we have chosen the image processing techniques to facilitate the doctors with a second opinion on the currently present testing mechanisms and also determine the severity of the patient.

Through the first half of the project, we have thoroughly read research papers related to present work in image-based classification of images and various optimal techniques used for it, and we have also shortlisted the ones to be implemented so as to fulfil the technical specifications and requirements. We have also successfully completed the flow of the entire system regarding image processing and CNN (Convolutional neural networks) techniques. Image pre-processing techniques that will be used are image resizing, segmentation, removing noise and image enhancement

AI mainly uses computer techniques to perform clinical diagnoses and suggest treatments. AI has the capability of detecting meaningful relationships in a data set and has been widely used in many clinical situations to diagnose, treat, and predict the results. We are working on two models, the first one is obtained for reviewing research papers which uses a CNN architecture for generating feature maps and SVM algorithm for classification of the disease. In the later approach, we plan to use a pretrained model CheXnet and fine tune it to work for COVID-19 X-ray images.

In the remainder of our project's duration, we aim to successfully implement these approaches and evaluate based on their performances. As an end result provides a suitable approach for classification of patients based on X-rays in an accurate manner.

Appendix

1) StarUML

StarUML is an open source UML modeling application. StarUML supports most of the diagrams like use case diagrams, activity diagrams, sequence diagrams, class diagrams, etc. It is rich in feature set and formatting options. The diagrams can be exported in jpg, jpeg and png formats. It provides various advancements than other modeling applications.

2) Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

3) Libraries

a) OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing; video capture and analysis including features like face detection and object detection.

b) Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. Human Behaviour Detection System Universal College of Engineering 43 Year: 2020 - 2021 [41]

c) Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis.

d) Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

e) Numpy NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

f) Skimage

Scikit-image (a.k.a. skimage) is a collection of algorithms for image processing and computer vision.

5) TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming.

Bibliography

1. Michelle Alva and Kavita Sonawane. Hybrid feature vector generation for alzheimer's disease diagnosis using mri images. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–6. IEEE, 2019.
2. Sohaib Asif, Yi Wenhui, Hou Jin, Yi Tao, and Si Jinhai. Classification of covid-19 from chest x-ray images using deep convolutional neural networks. *MedRxiv*, 2020.
3. Ahmedelmubarak Bashir, Zeinab A Mustafa, Islah Abdelhameid, and Rimaz Ibrahem. Detection of malaria parasites using digital image processing. In *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pages 1– 5. IEEE, 2017.
4. Gaurav Chauhan. All about logistic regression, Oct 2018.
5. Biraja Ghoshal and Allan Tucker. Estimating uncertainty and inter- pretability in deep learning for coronavirus (covid-19) detection. *arXiv preprint arXiv:2003.10769*, 2020.
6. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
7. Ezz El-Din Hemdan, Marwa A Shouman, and Mohamed Esmail Karar. Covidx-net: A framework of deep learning classifiers to diagnose covid- 19 in x-ray images. *arXiv preprint arXiv:2003.11055*, 2020.
8. HB Kekre. Image retrieval using histogram based bins of pixel counts and average of intensities. *International Journal of Computer Science and Information Security*, 10(1):74, 2012.
9. HB Kekre and Kavita Sonawane. Bins formation using cg based partitioning of histogram modified using proposed polynomial transform 'y= 2x-x 2'for cbir. 2011.
10. HB Kekre and Kavita Sonawane. Comparative performance of linear and cg based partitioning of histogram for bins formation in cbir. *Interna- tional Journal of Engineering Research and Development*, 5(7):75–83, 2013.
11. HB Kekre and Kavita Sonawane. Histogram bins matching approach for cbir based on linear grouping for dimensionality reduction. *International Journal of Image, Graphics and Signal Processing*, 6(1):68, 2013.
12. HB Kekre and Kavita Sonawane. Role of histogram modification functions eqh, log, poly, linear equations for cbir based on 64 bins approach. *International Journal of Advanced Science and Technology*, 59:53–70, 2013.
13. HB Kekre and Kavita Sonawane. Use of equalized histogram cg on statistical parameters in bins approach for cbir. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–6. IEEE, 2013.
14. Asif Iqbal Khan, Junaid Latief Shah, and Mohammad Mudasir Bhat. Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images. *Computer Methods and Programs in Biomedicine*, 196:105581, 2020.

15. Akib Mohi Ud Din Khanday, Syed Tanzeel Rabani, Qamar Rayees Khan, Nusrat Rouf, and Masarat Mohi Ud Din. Machine learning based approaches for detecting covid-19 using clinical text data. *International Journal of Information Technology*, 12(3):731–739, 2020.
16. Rahul Kumar, Ridhi Arora, Vipul Bansal, Vinodh J Sahayasheela, Himanshu Buckchash, Javed Imran, Narayanan Narayanan, Ganesh N Pandian, and Balasubramanian Raman. Accurate prediction of covid- 19 using chest x-ray images through deep feature learning model with smote and machine learning classifiers. *MedRxiv*, 2020.
17. Lin Li, Lixin Qin, Zeguo Xu, Youbing Yin, Xin Wang, Bin Kong, Junjie Bai, Yi Lu, Zhenghan Fang, Qi Song, et al. Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct. *Radiology*, 2020.
18. Arpan Mangal, Surya Kalia, Harish Rajgopal, Krithika Rangarajan, Vinay Namboodiri, Subhashis Banerjee, and Chetan Arora. Covidaid: Covid-19 detection using chest x-ray. *arXiv preprint arXiv:2004.09803*, 2020.
19. Chaimae Ouchicha, Ouafae Ammor, and Mohammed Meknassi. Cvdnet: A novel deep learning architecture for detection of coronavirus (covid-19) from chest x-ray images. *Chaos, Solitons & Fractals*, 140:110245, 2020.
20. Tulin Ozturk, Muhammed Talo, Eylul Azra Yildirim, Ulas Baran Baloglu, Ozal Yildirim, and U Rajendra Acharya. Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in biology and medicine*, 121:103792, 2020.
21. Prashant Patel. Chest x-ray (covid-19 pneumonia), Sep 2020.
22. Prabira Kumar Sethy, Santi Kumari Behera, Pradyumna Kumar Ratha, and Preesat Biswas. Detection of coronavirus disease (covid-19) based on deep features and support vector machine. 2020.
23. Ying Song, Shuangjia Zheng, Liang Li, Xiang Zhang, Xiaodong Zhang, Ziwang Huang, Jianwen Chen, Huiying Zhao, Yusheng Jie, Ruixuan Wang, et al.Deep learning enables accurate diagnosis of novel coronavirus (covid-19) with ct images. *MedRxiv*, 2020.
24. Hrishikesh Telang and Kavita Sonawane. Effective performance of bins approach for classification of malaria parasite using machine learning. In *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, pages 427–432. IEEE, 2020.
25. Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid- 19 cases from chest x-ray images. *Scientific Reports*, 10(1):1–12, 2020.
26. Xiaowei Xu, Xiangao Jiang, Chunlian Ma, Peng Du, Xukun Li, Shuangzhi Lv, Liang Yu, Qin Ni, Yanfei Chen, Junwei Su, et al. A deep learning system to screen novel coronavirus disease 2019 pneumonia. *Engineering*, 6(10):1122–1129, 2020.

Acknowledgements

A project is always coordinated, guided and scheduled team effort aimed at realizing a common goal. We are grateful and gracious to all those people who have helped and guided us through this project and make this experience worthwhile.

We would also like to express our deep gratitude to our Project Guide and our Professor, **Dr. Jinan Fiaidhi** who constantly guided and supervised us, and also furnished essential information concerning the betterment of our project. This work would not have been possible without her necessary insights and intellectual suggestions that has helped us achieve this juncture of our progress.