CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

Blood smear analysis is an important diagnostic test which is performed to diagnose an array of diseases. The count of various blood cells and their morphological properties are the main focus of this test. Manual analysis of blood smears is time consuming and laborious. By automating this process and ultimately narrowing the scope of possible diseases, a considerable amount of time can be saved. This may in turn help medical staff as well as the patients. In the proposed system an automated technique for blood smear analysis using Convolutional Neural Network of deep learning is proposed to discern the blood cell count. The results of the first CNN model are then employed to generate a range capable of predicting possible diseases using the second CNN model.

A blood smear is essentially is a thin film of blood which is spear on a glass slide and is studied by pathologist under a microscope. It is usually done in combination with CBC (complete blood count). Human blood consists of two types of cells, Red blood corpuscles (RBC) and white blood corpuscles (WBC). WBC’s or leukocytes are cells which constitute the immune system of the body. They fight back infections and foreign bodies. Any abnormality in their count and appearance could be due immunodeficiency diseases, leukemia etc. or simply an infection or allergic reaction. Analysis of blood smear involves examining blood smear slides under the microscope. Examination focuses on various aspects some of which are the number of various types of cells, their shape, size, if they have some foreign bodies or not, their color is right or not etc. This makes this technique prone to human error.

This process can be automated by using deep learning. Convolutional Neural Networks can be used to classify the WBCs in each blood smear into its types digitally, with the help of computer algorithms. In this input is an image, a series of images; the output may be either an image or some parameter of an image. By applying various operations like grey scale conversion, thresholding, boundary analysis, image segmentation etc. useful information from blood smear images can be obtained.

The aim will be to harness the image processing capacity of OpenCV in order to obtain the necessary information required for disease detection. Convolutional Neural Networks (CNN) is a deep learning model which will be used to map the quantification result to a particular disease.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

RESEARCH PAPERS

To understand the process of classification and quantification of WBCs using CNN, many research papers were studied analyzed.

**2.1 Disease Detection using Blood Smear Analysis by Pragati Sharma**

This paper gets the count by steps in order of grey scale conversion, noise removal, image conversion to binary and finally removing the RBCs and platelets. It uses the concepts of image processing and neuro fuzzy systems.

**2.2 Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images**

This paper majorly focuses on robust segmentation and the consecutive image processing by doing preprocessing, estimation of average cell diameter and segmentation. The segmentation is done using varied techniques and the results are combined in order to exploit all available a-priori information.

**2.3 A Framework for White Blood Cell Segmentation in Microscopic Blood Images using Digital Image Processing**

In this paper, the concepts used are of segmentation, canny edge detection, snake algorithm and Zack thresholding. The algorithms are implemented in a series of steps of color scale image conversion, conversion to grey scale, sub image separation, edge detection, finding nuclei, hole filling and cytoplasm and segmenting the cytoplasm.

**2.4 Automatic Detection and Quantification of WBCs and RBCs Using Iterative Structured Circle Detection**

This paper deals with summary ofresults of count of WBCs and RBCs, detection of overlapped cells and irregular shapes. It uses the technological concepts of morphological operations, segmentation and iterative structured circle detection.

This research can be summarized in a tabular format as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Name of Paper** | **Technology used** | **Algorithm/Steps** | **Advantages and Disadvantages** |
| 1. | Disease Detection using Blood Smear Analysis by Pragati Sharma | 1.Image processing  2.neuro fuzzy system | 1.Gray scale conversion  2.Noise removal(Complement image and fill holes)  3.Image conversion to binary  4.Get count by removing rbc and platelets | Advantages:  1. Robustness achieved due to use of neural networks.  Disadvantages:  1.Image pre-processing is time consuming. |
| 2. | Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images | Robust Segmentation  Image Processing | 1. To preprocess the image in order to reduce acquisition noise and background non-uniformities  2. To estimate the average cell diameter  3. To perform segmentation with different techniques and combine the results in order to exploit all the available a-priori information achieving a robust identification of white cells. | Advantages:  1.Proposes a method to enhance the microscope images  by removing the undesired microscope background, a method  for the robust estimation and a new  fully self adaptive segmentation strategy to robustly identify  white cells.  Disadvantages:  1.Tends to segment broken White cells. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Name of Paper** | **Technology used** | **Algorithm/Steps** | **Advantages and Disadvantages** |
| 3. | A Framework for White Blood Cell Segmentation in Microscopic Blood Images using Digital Image Processing | 1.Segmentation  2.Snake Algorithm.  3.Zack thresholding  4.Canny edge detection | 1.The color scale image of microscope blood image  2.Convert to gray level image.  3.Sub image separation of wbc.  4.Edge detection.  5.Finding nuclei using GVF Snake.  6.Hole filling  7.Segmented nucleus.  8.Subtract nucleus from grayscale image  9.Finding cytoplasm using Zack thresholding.  10.Segment cytoplasm. | Advantages:  1.Snake detection algorithm i adaptive and detects active contours.  Disadvantages:  1.Need to connect the resulting edges to extract the complete edges that seem so obvious for the human mind |
| 4. | Automatic Detection and Quantification of WBCs using Iterative Structured Circle Detection  Algorithm by Yazan M. Alomari,1 Siti Norul Huda Sheikh Abdullah. | 1.Segmenting  2.Morphological operations  3.Iterative Structured Circle  Detection. | 1.Summary of results of count of wbcs and rbcs  2.Overlapping cells detected  3.irregular samples detected. | Advantages:  1.Automated the counting process  2.Improvement in capability of detecting irregular cells Disadvantages:  1.Wrong detection of cells in some cases |

CHAPTER 3

EXISTING SYSTEM

CHAPTER 3

EXISTING SYSTEM

Manual analysis of blood smear involves examining blood smear slides under the microscope. Examination focuses on various aspects some of which are the number of various types of cells, their shape, size, if they have some foreign bodies or not, their color is right or not etc. This makes this technique prone to human error. This process can be automated by using image processing. Abnormal increase in WBC disrupts the balance of the blood system. For someone who has leukemia, bone marrow produces abnormal WBC. A large number of medical images in digital format are generated by hospitals and medical institutions every day. The main purpose of the blood cell image analysis is differentiating the components of blood and counting of Red Blood Corpuscles (RBCs), White Blood Corpuscles (WBCs) and platelets by observing blood cell and also detecting various diseases. For example, an image may contain up to 100 red cells and only 1 to 3 white cells. Abnormal high or low counts may indicate the presence of many forms of disease, since blood counts are amongst the most commonly performed blood test in medicine. WBC reveals important diagnostic information about the patients. Various types of cells in normal human blood are Red Blood Corpuscles, White Blood Corpuscles and Blood platelets. RBCs are simple and similar. WBC contains nucleus and cytoplasm. By identifying the WBCs, it also provided some information about the abnormal condition in our body. The proposed method provides a software-based cost effective and an efficient alternative in detecting and counting WBCs. Image processing techniques help to count the cells in the human blood and, at the same time, provide information of the cell morphology. In this proposed work, twelve microscopic human blood cell images have been analyzed and the number of WBC counted accurately for detecting abnormalities in human health.

# CHAPTER 4

# SCOPE

CHAPTER 4

SCOPE

The scope of the project is to design and implement a disease detection system which will be able to diagnose disease using WBC count from microscopic blood smear image samples. The pathologists and doctors who will be the primary users of the system will be able to automatically detect the count and classify the disease using the count and report can be easily generated based on the results obtained. The scope will be to make a system that will be more efficient than the manual system. By automating this process and ultimately narrowing the scope of possible diseases, a considerable amount of time can be saved. This may in turn help medical staff as well as the patients. In this project an automated technique for blood smear analysis using image processing is proposed to discern the blood cell count and blood cell properties. The results of image processing are then employed to generate a neuro-fuzzy system capable of predicting possible diseases. Thus due to this tedious nature of the differential white blood cells an automatic system is more preferable.

# CHAPTER 5

# PROPOSED SYSTEM

# 

# CHAPTER 5

PROPOSED SYSTEM

# The steps followed during the process were as follows:

# **1. Image Classification**

# The main task of image classification is acceptance of the input image and the following definition of its class with the help of computer algorithms. In this input is an image, a series of image the output of image processing may be the classified clusters.

Convolutional Layers: The Convolution layer is always the first.Тhe image (matrix with pixel values) is entered into it. Next the software selects a smaller matrix there, which is called a filter (or neuron, or core). Then the filter produces convolution, i.e. moves along the input image.

The nonlinear layer is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer.

# 

**2. Range Calculation:**

Once the WBCs are classified into their types, each type is quantified into a range of high, medium or low count. This range is the the input for the next CNN model of disease prediction.

**3. Disease Prediction**:

The output of the last CNN model that is the range of each WBC type along with user input attributes like age and gender of the respective patient acts as the input of the second convolutional model. This model maps this input to the most proable disease and returns the name of the predicted disease. In case of normalcy, this model will return the value ‘none’.

# CHAPTER 6

REQUIREMENT GATHERING

## CHAPTER 6

## REQUIREMENT GATHERING

## Software Requirements

* Operating System(Any One):
  1. Microsoft Windows 7/8/10 (Recommended)
  2. Linux Ubuntu
* Front-end: BootStrap, HTML, CSS
* Back-end: OpenCV 3.2.0
* Dataset: Kaggle and Pathology labs
* TensorFlow r1.12/Keras 2.2.0

## Hardware Requirements

Master Computer at the security room with specification:

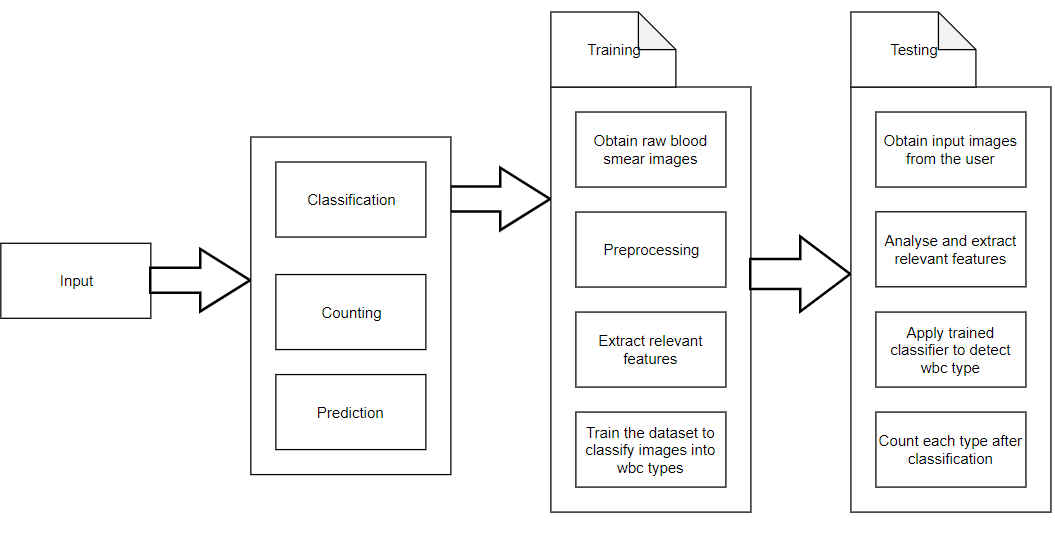
* Intel i3 Processor or above
* 4GB RAM (Minimum)
* 1 TB HDD

CHAPTER 7

SYSTEM DESIGN

CHAPTER 7

SYSTEM DESIGN

The section defines the diagrams which define the complete flow of modules as described in the proposed system:  


7.1.Block Diagram

**7.1.Input**

This module allows users to enter their name, age , gender along with 12 blood smear images which would further help the system to predict the probable disease that the patient is suffering from.

**7.2.Classification**

This module helps the system to classify the wbcs into its four sub types namely lymphocyte , neutrophil ,monocyte and eosinophil. They are present in a particular ratio in the blood of humans. So every type of the wbc has a threshold value beyond which the presence of the same is considered s abnormal and hence the patient might suffer from the disease.

**7.3. Prediction**

The prediction module helps to predict the most probable disease that the patient is suffering from. This module is built using cnn that takes into consideration the gender and age of the patient to predict the disease. For example, it is not possible to have breast cancer in males. Thus the dataset helps to accurately predict the disease. **7.4.Training model**

For training, python is used as a syntax whereas framework used is Keras which is a high levelneural network API written in python. But Keras cant work by itself , hence it needs a backend for low level operations. Therefore, TensorFlow is used for the same.

**7.4.1. Obtain raw blood smear images**

The raw blood images used to train the cnn module were obtained from Kaggle. The pathologist also has nearly a set of 1000 images from the blood sample of the patient. The images are stained in such a way that it is easy to recognize wbc and rbc separately.

**7.4.2. Preprocessing**

In the preprocessing stage, the data is transformed in an understandable format . It solves the issue of cleaning and presenting the data in a proper format to the cnn model.Many issues are resolved in this stage. To increase the size of dataset the images are duplicated , rotated and flipped .since larger the size of dataset higher the accuracy of the training model.

**7.4.3. Extract relevant features**

The training model extracts relevant features from the images. It is a supervised training model so as the layers are added the feature extraction increases.We are using a 5 layer model which helps to train the dataset .

**7.4.4.Train the dataset to classify wbc into its types**

After the model construction is done we next move to the model training part. After adding a sufficient number of layers the model is complied. At this moment the Keras communicates with TensorFlow for constraction of the model. The trained model can be then used for real world.

**7.5.Testing**

Once the model has been trained it is possible to carry out testing. During this phase a second set of data is loaded.

**7.5.1. Obtain input images from the user**

The user that is the pathologist needs to upload a total of twelve images of blood smear collected from the patient.

**7.5.2. Analyse and extract relevant features**

The features from the input images are matched with the trained dataset. The relevant features are extracted and matched. This is how the supervised learning model works.

**7.5.3. Apply trained classifier to detect wbc type**

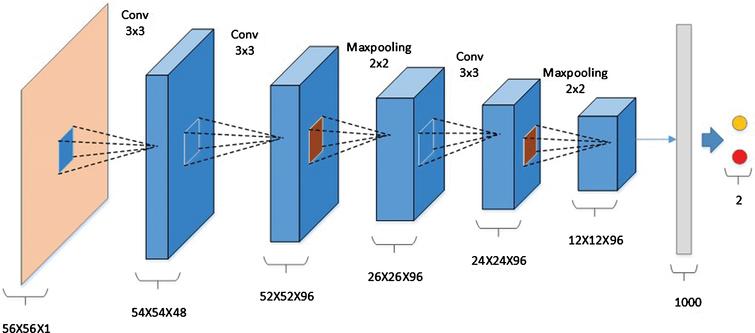
The twelve images are classified into its four classes monocytes , lym;hocytes ,neutrophil and basophil.

**7.5.4. Counting**

In this phase the counting of each type of wbc is done. If the value of the count exceeds or is lower than the threshold value assigned to the wbc type then the person is considered to have an abnormal count . This helps the doctor to have an idea of what the patient might be suffering from.

**7.6. CNN model structure**

Let us have a look at the cnn model that we used.



7.2.Cnn model structure

The model consists of following layers that help in feature extraction:

**1.Convolution layer:**

It is always the first. Тhe image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a **filter**(or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter’s task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then a input matrix.

This operation, from a human perspective, is analogous to identifying boundaries and simple colours on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.

The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

**2.Non Linear Layer**

This operation, from a human perspective, is analogous to identifying boundaries and simple colours on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.

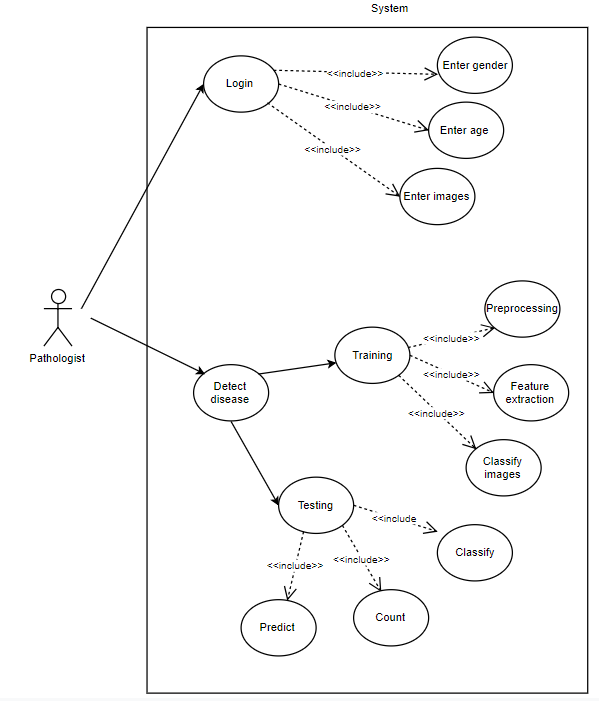
The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

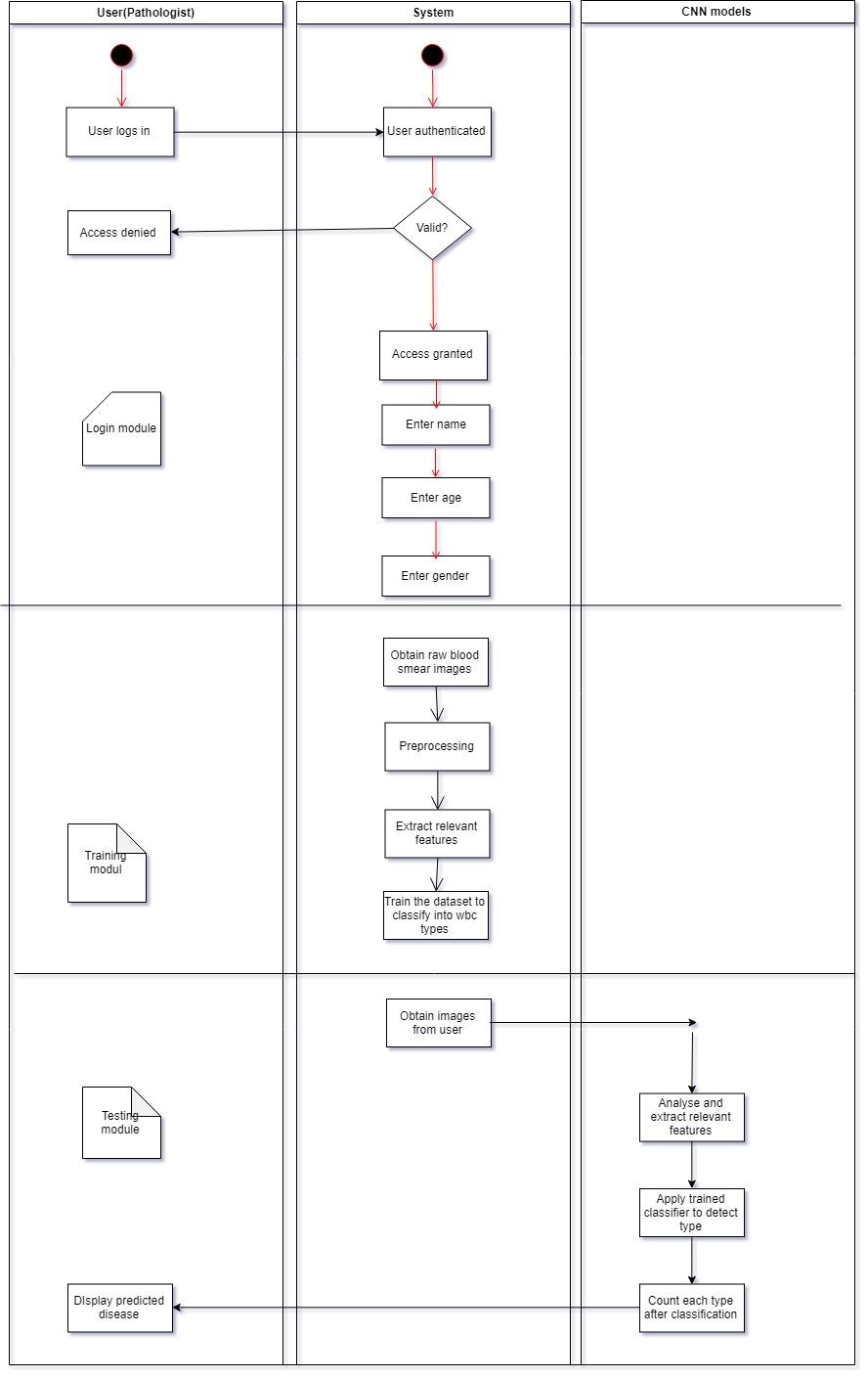
**3.Pooling layer**

**The pooling layer** follows the nonlinear layer. It works with width and height of the image and performs a downsampling operation on them. As a result the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, than a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach **a** **fully connected layer**. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class.

A fragment of the code of this model written in Python will be considered further in the practical part.

Followint are the usecase and activity diagram for our system:7.3.Use case diagram

7.4.Sequential diagram

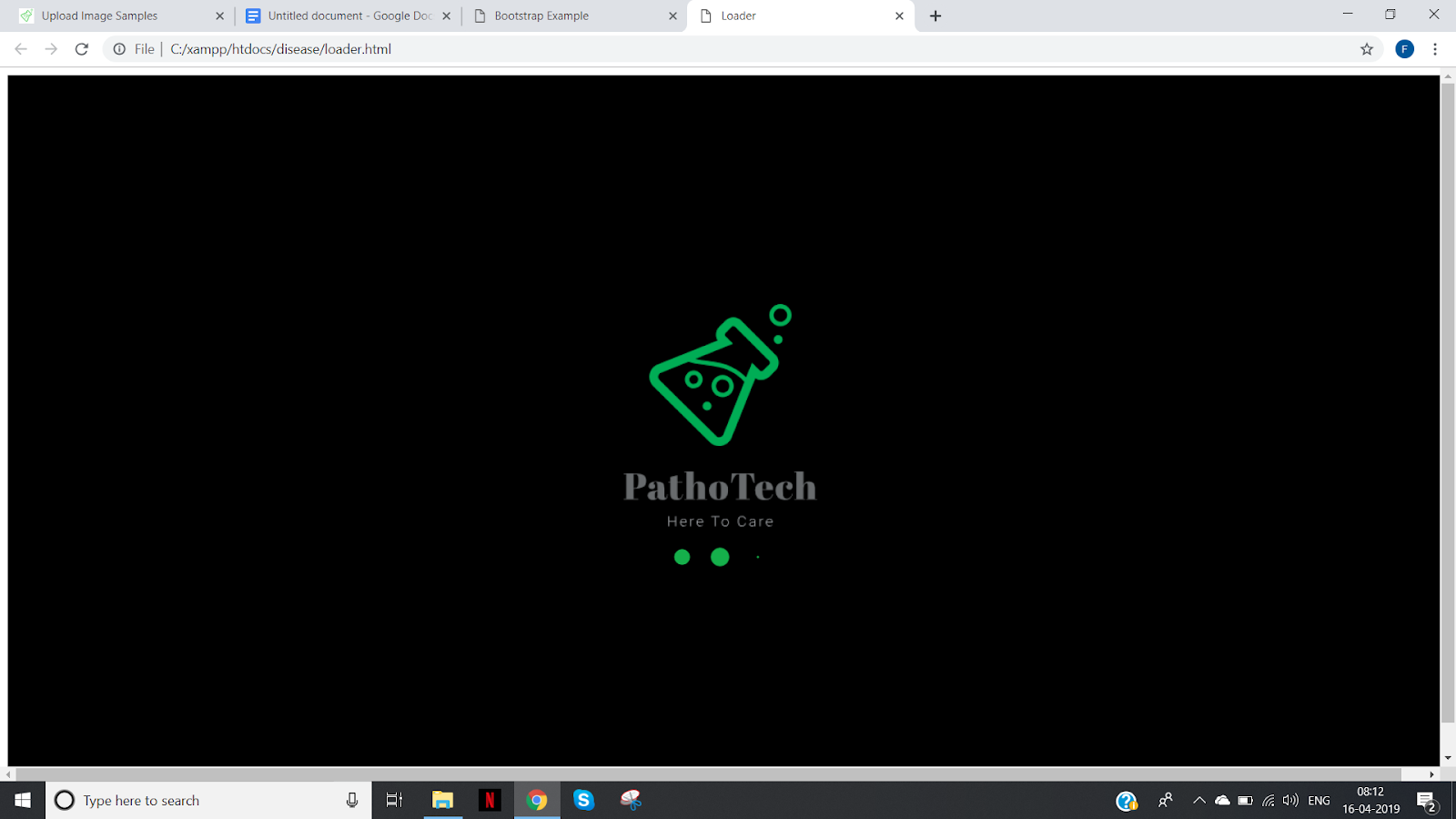
CHAPTER 8

IMPEMENTATION

CHAPTER 8

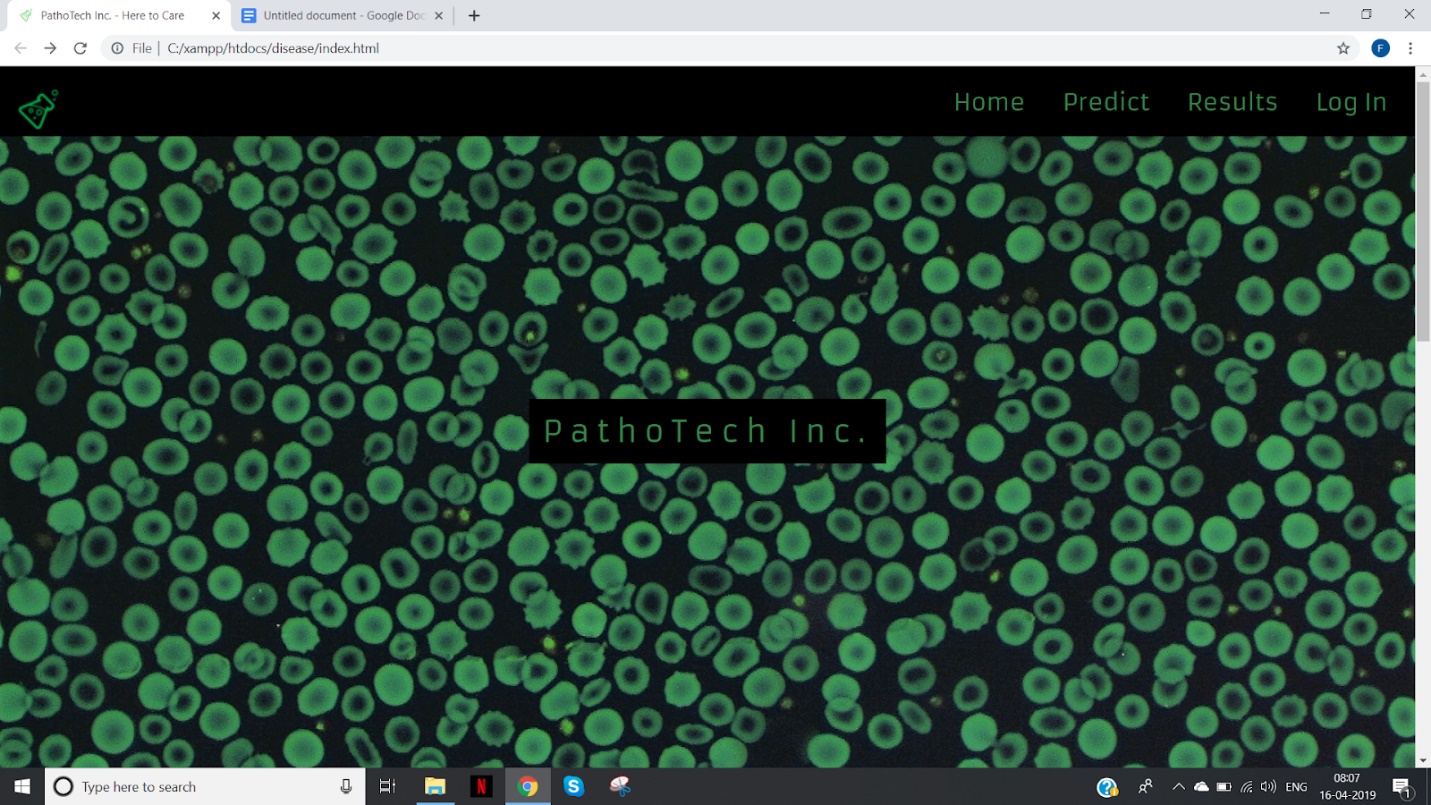
IMPLEMENTATION

8.1.Open the website from localhost

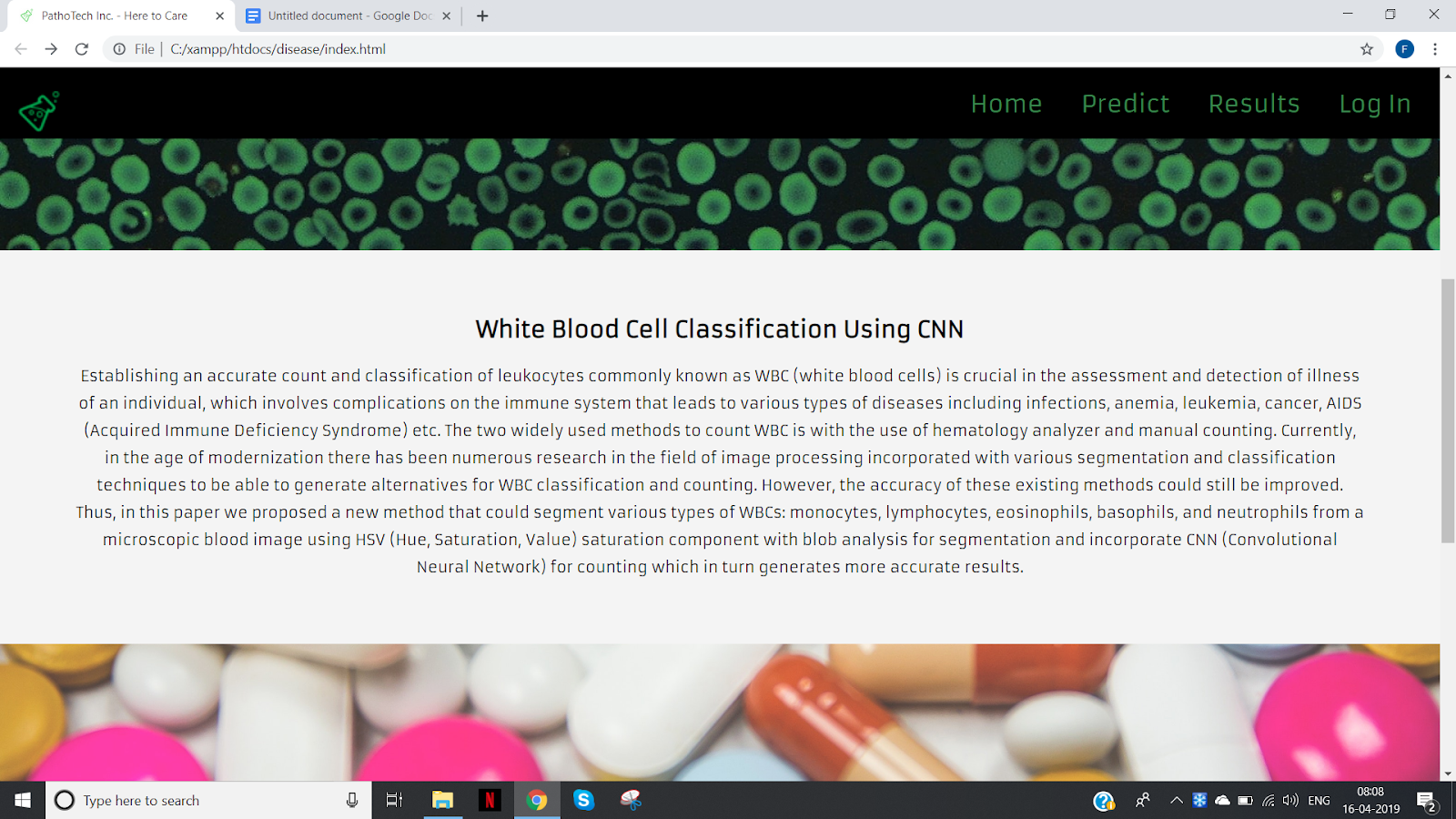
  
  
8.1. Logo

* The website is opened using the link “localhost:88/disease”.
* As soon as the website is opened, the user (pathologist ) is greeted with the official logo designed for the website.
* The name of the website is Pathotech with a tagline of “Here to care”.

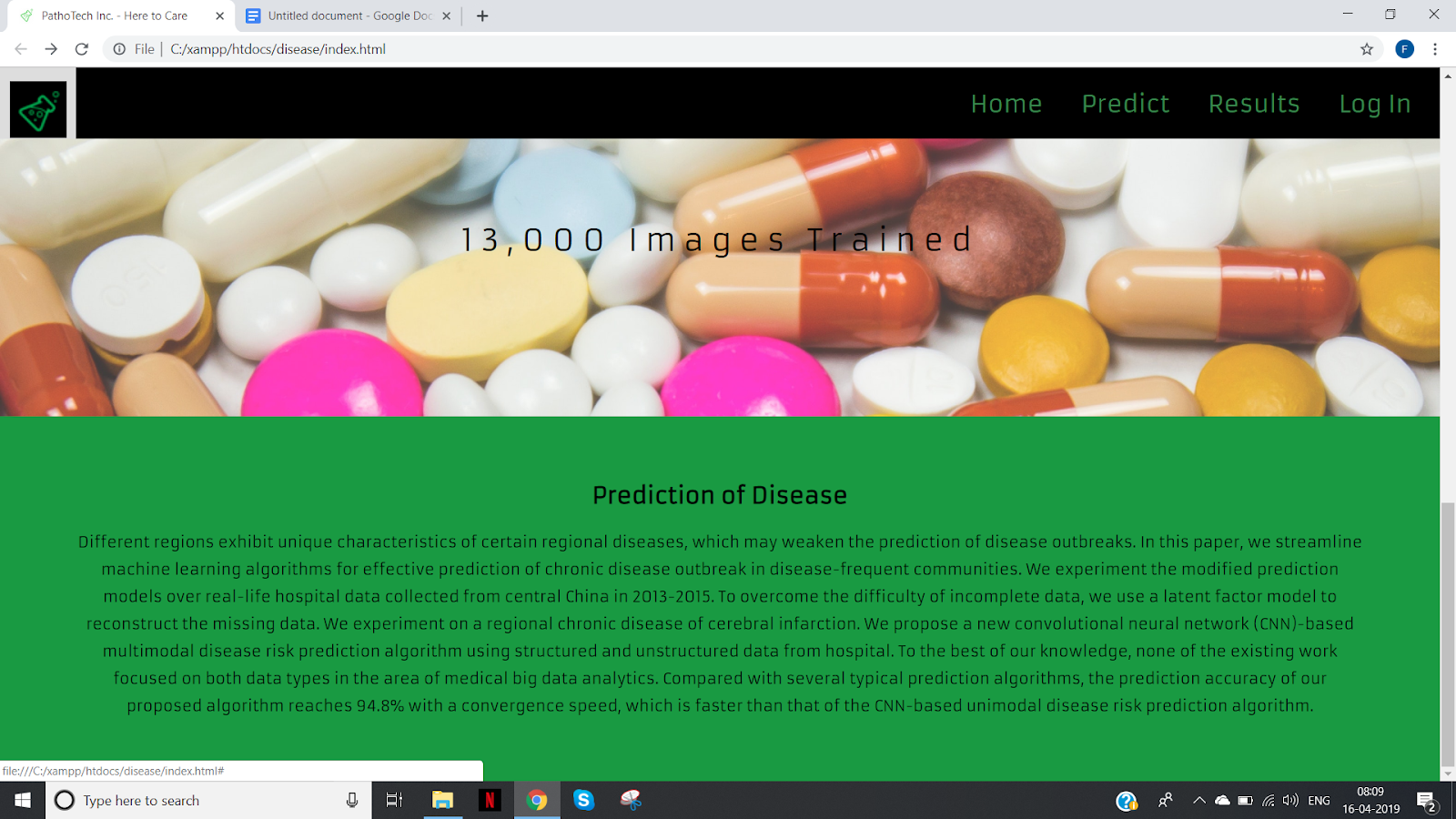
8.2. Website look



8.2. Website look



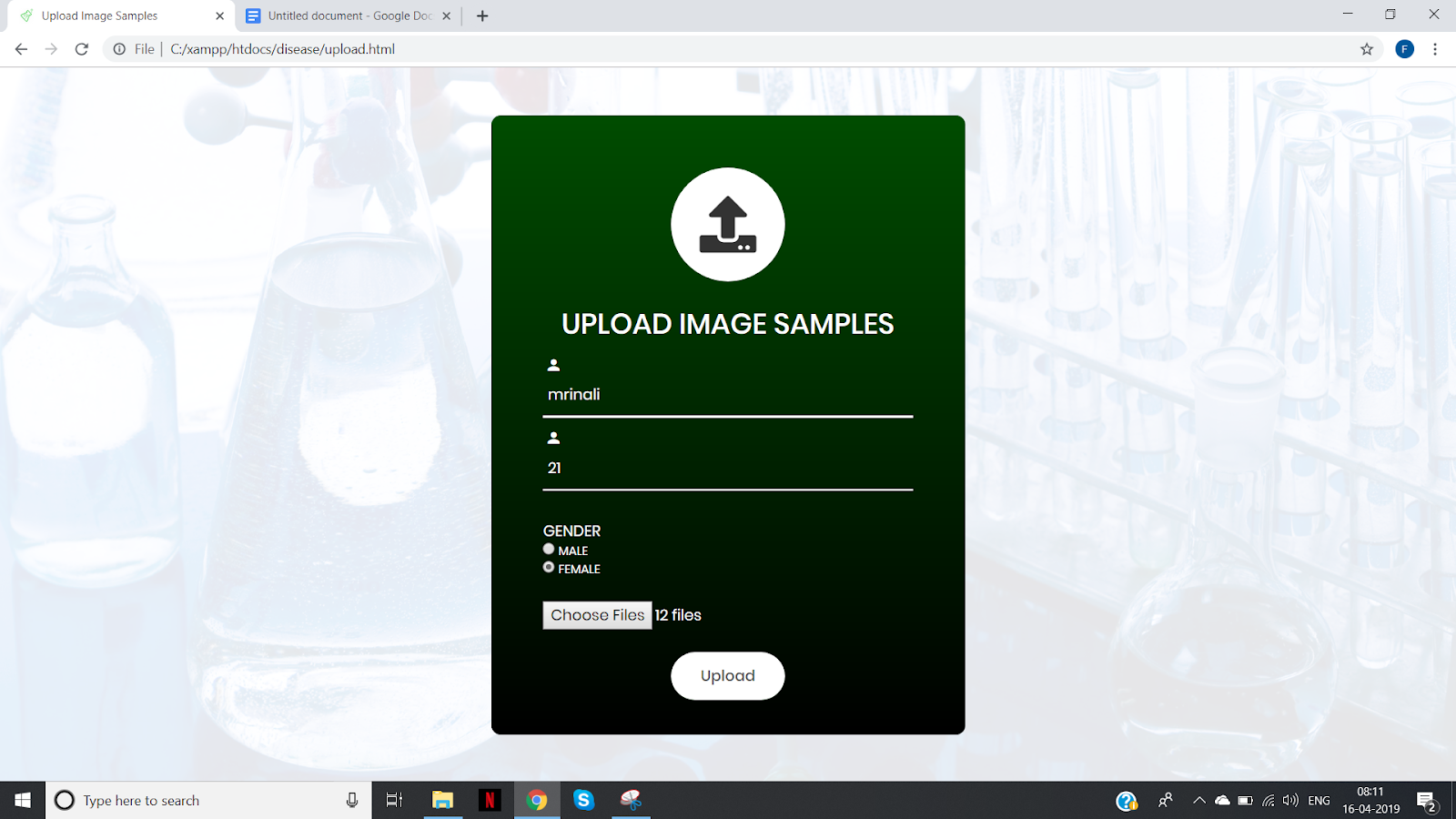
8.3. Website look



8.4. Website look

* The website look is as shown in the above diagrams.
* An effect of parallax has been added to make the website look more attractive.
* The site is quite informative so that the user can easily access it withour having any second thoughts.
* There are a totsl of four categories wherein the user can click namely home, predict, results and login.
* If the user wants to predict a disease or check the normality of the blood samples he can go to the website and click on predict .

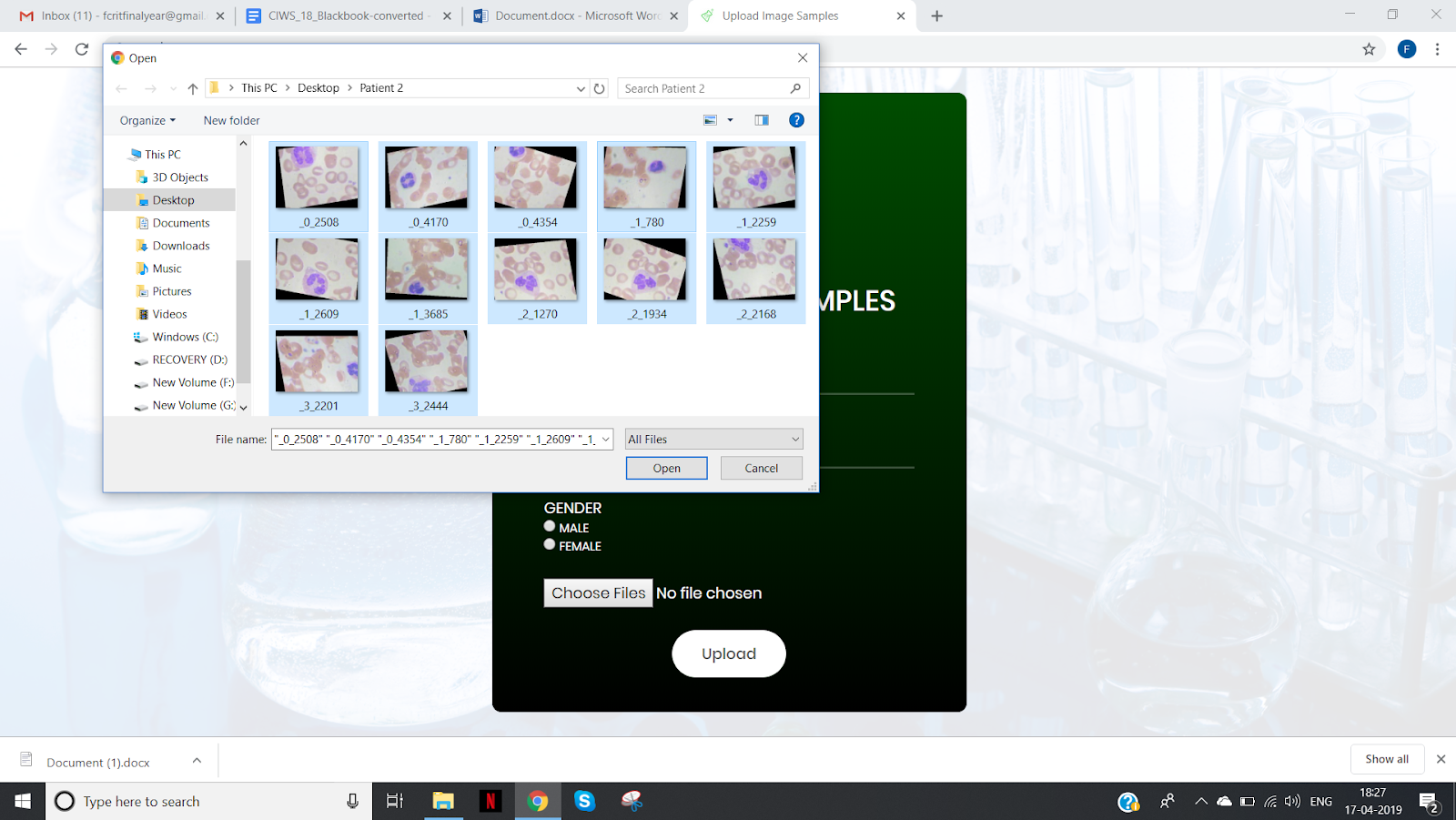
8.3. Enter user details



8.5.Enter user details

* Enter the user details consisting of the name , age and gender.
* A radio button is used for selecting gender that is male or female.
* Along with these details the user needs to also upload the images of blood samples.
* The samples should be stained and rbc and wbc should be distinguishable in the images.

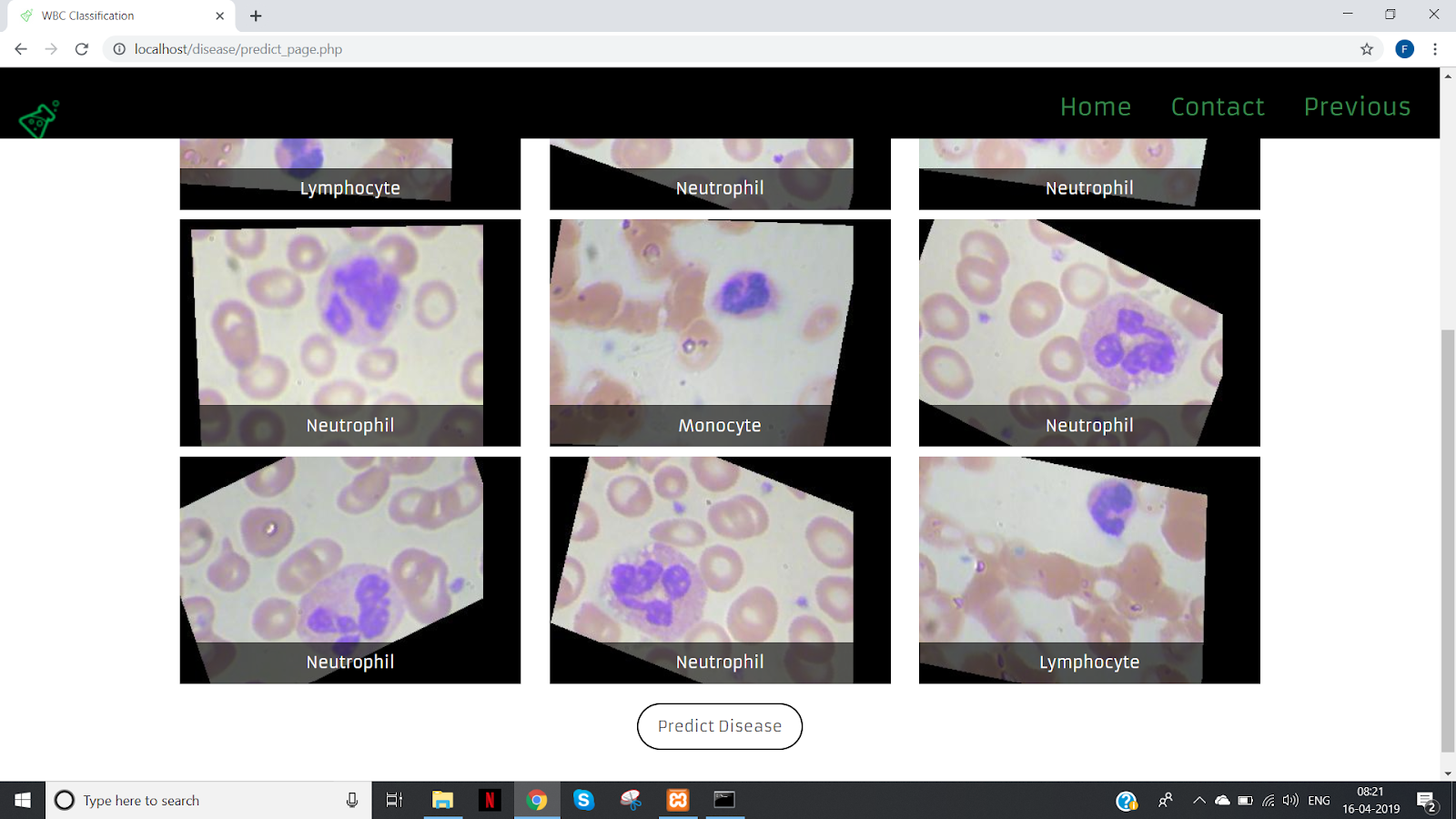
**8.4. Uploading images**



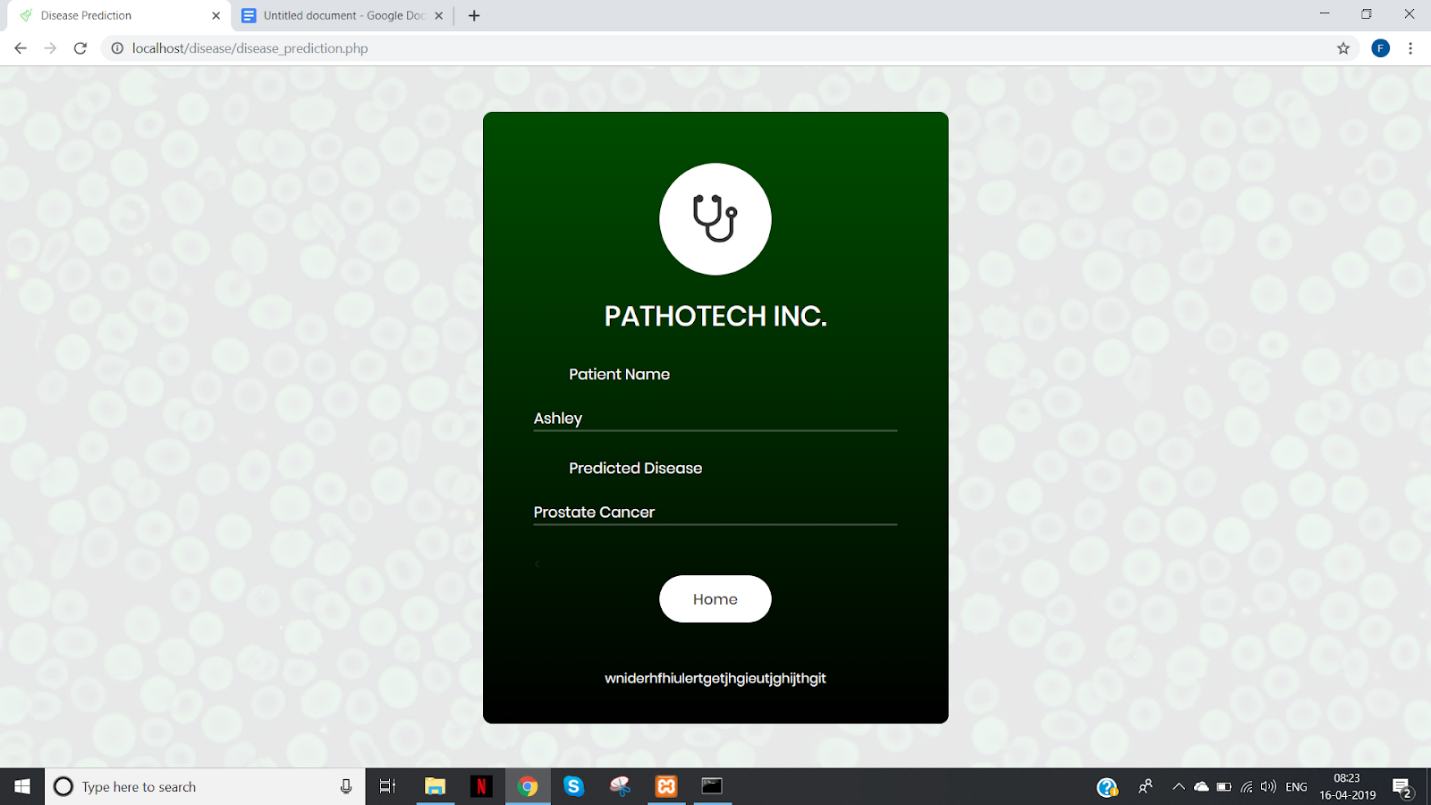
8.6. Upload images

The user should upload 12 images from their respective folder.

**8.5. Prediction**

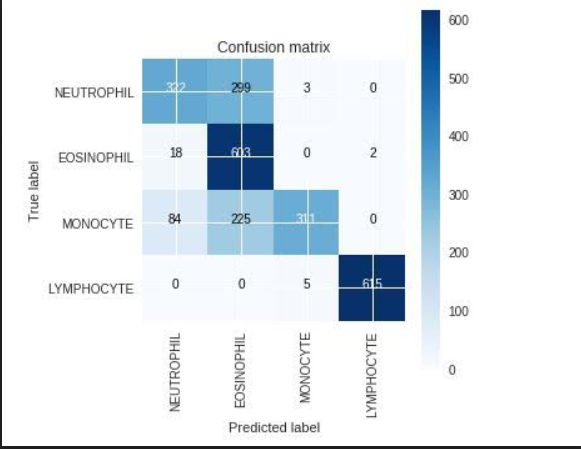


8.7. Wbc image classification

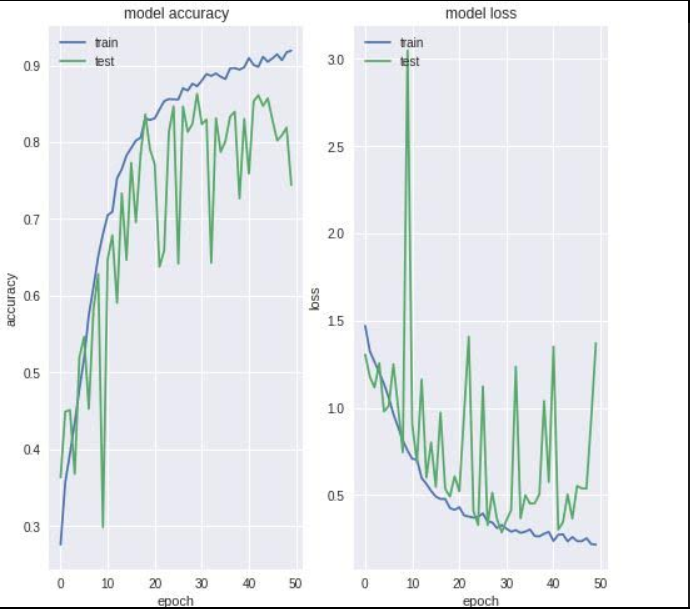


8.8.Predicted disease

The final output is displayed in the form of classification and prediction.



8.9. Confusion matrix



8.10.Graphs

CHAPTER 9

TIMELINE CHART

CHAPTER 9

TIMELINE CHART



9.1. Timeline chart

CHAPTER 10

APPENDIX

CHAPTER 10

APPENDIX

10.1. CNN CLASSIFICATION MODEL

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import pandas as pd

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, Lambda, MaxPool2D, BatchNormalization

from keras.utils import np\_utils

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import RMSprop

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

import xml.etree.ElementTree as ET

import sklearn

import itertools

import cv2

import scipy

import os

import csv

import matplotlib.pyplot as plt

%matplotlib inline

#import os

dict\_characters = {1:'NEUTROPHIL',2:'EOSINOPHIL',3:'MONOCYTE',4:'LYMPHOCYTE'}

dict\_characters2 = {0:'Mononuclear',1:'Polynuclear'}

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import pandas as pd

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, Lambda, MaxPool2D, BatchNormalization

from keras.utils import np\_utils

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import RMSprop

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

import xml.etree.ElementTree as ET

import sklearn

import itertools

import cv2

import scipy

import os

import csv

import matplotlib.pyplot as plt

%matplotlib inline

from tqdm import tqdm

def get\_data(folder):

    """

    Load the data and labels from the given folder.

    """

    X = []

    y = []

    z = []

    for wbc\_type in os.listdir(folder):

        if not wbc\_type.startswith('.'):

            if wbc\_type in ['NEUTROPHIL']:

                label = 1

                label2 = 1

            elif wbc\_type in ['EOSINOPHIL']:

                label = 2

                label2 = 1

            elif wbc\_type in ['MONOCYTE']:

                label = 3

                label2 = 0

            elif wbc\_type in ['LYMPHOCYTE']:

                label = 4

                label2 = 0

            else:

                label = 5

                label2 = 0

            for image\_filename in tqdm(os.listdir(folder + wbc\_type)):

                img\_file = cv2.imread(folder + wbc\_type + '/' + image\_filename)

                if img\_file is not None:

                    img\_file = scipy.misc.imresize(arr=img\_file, size=(60, 80, 3))

                    img\_arr = np.asarray(img\_file)

                    X.append(img\_arr)

                    y.append(label)

                    z.append(label2)

    X = np.asarray(X)

    y = np.asarray(y)

    z = np.asarray(z)

    return X,y,z

X\_train, y\_train, z\_train = get\_data('DatasetMasterr/dataset2-master/images/TRAIN/')

X\_test, y\_test, z\_test = get\_data('DatasetMasterr/dataset2-master/images/TEST/')

# Encode labels to hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])

from keras.utils.np\_utils import to\_categorical

y\_trainHot = to\_categorical(y\_train, num\_classes = 5)

y\_testHot = to\_categorical(y\_test, num\_classes = 5)

z\_trainHot = to\_categorical(z\_train, num\_classes = 2)

z\_testHot = to\_categorical(z\_test, num\_classes = 2)

print(dict\_characters)

print(dict\_characters2)

# Helper Functions  Learning Curves and Confusion Matrix

from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

class MetricsCheckpoint(Callback):

    """Callback that saves metrics after each epoch"""

    def \_\_init\_\_(self, savepath):

        super(MetricsCheckpoint, self).\_\_init\_\_()

        self.savepath = savepath

        self.history = {}

    def on\_epoch\_end(self, epoch, logs=None):

        for k, v in logs.items():

            self.history.setdefault(k, []).append(v)

        np.save(self.savepath, self.history)

def plotKerasLearningCurve():

    plt.figure(figsize=(10,5))

    metrics = np.load('logs.npy')[()]

    filt = ['acc'] # try to add 'loss' to see the loss learning curve

    for k in filter(lambda x : np.any([kk in x for kk in filt]), metrics.keys()):

        l = np.array(metrics[k])

        plt.plot(l, c= 'r' if 'val' not in k else 'b', label='val' if 'val' in k else 'train')

        x = np.argmin(l) if 'loss' in k else np.argmax(l)

        y = l[x]

        plt.scatter(x,y, lw=0, alpha=0.25, s=100, c='r' if 'val' not in k else 'b')

        plt.text(x, y, '{} = {:.4f}'.format(x,y), size='15', color= 'r' if 'val' not in k else 'b')

    plt.legend(loc=4)

    plt.axis([0, None, None, None]);

    plt.grid()

    plt.xlabel('Number of epochs')

    plt.ylabel('Accuracy')

def plot\_confusion\_matrix(cm, classes,

                          normalize=False,

                          title='Confusion matrix',

                          cmap=plt.cm.Blues):

    """

    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """

    plt.figure(figsize = (5,5))

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick\_marks = np.arange(len(classes))

    plt.xticks(tick\_marks, classes, rotation=90)

    plt.yticks(tick\_marks, classes)

    if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

                 horizontalalignment="center",

                 color="white" if cm[i, j] > thresh else "black")

    plt.tight\_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

def plot\_learning\_curve(history):

    plt.figure(figsize=(8,8))

    plt.subplot(1,2,1)

    plt.plot(history.history['acc'])

    plt.plot(history.history['val\_acc'])

    plt.title('model accuracy')

    plt.ylabel('accuracy')

    plt.xlabel('epoch')

    plt.legend(['train', 'test'], loc='upper left')

    plt.savefig('./accuracy\_curve.png')

    #plt.clf()

    # summarize history for loss

    plt.subplot(1,2,2)

    plt.plot(history.history['loss'])

    plt.plot(history.history['val\_loss'])

    plt.title('model loss')

    plt.ylabel('loss')

    plt.xlabel('epoch')

    plt.legend(['train', 'test'], loc='upper left')

    plt.savefig('./loss\_curve.png')

import keras

from google.colab import files

GOOGLE\_COLAB=True

MFILE='deeps\_wbc2.h5'

dict\_characters = {1:'NEUTROPHIL',2:'EOSINOPHIL',3:'MONOCYTE',4:'LYMPHOCYTE'}

dict\_characters2 = {0:'Mononuclear',1:'Polynuclear'}

def runKerasCNNAugment(a,b,c,d,e):

    batch\_size = 128

    num\_classes = len(b[0])

    epochs = 50

#     img\_rows, img\_cols = a.shape[1],a.shape[2]

    img\_rows,img\_cols=60,80

    input\_shape = (img\_rows, img\_cols, 3)

    model = Sequential()

    model.add(Conv2D(32, kernel\_size=(3, 3),

                     activation='relu',

                     input\_shape=input\_shape,strides=e))

    model.add(Conv2D(64, (3, 3), activation='relu'))

    model.add(MaxPooling2D(pool\_size=(2, 2)))

    model.add(Dropout(0.25))

    model.add(Flatten())

    model.add(Dense(128, activation='relu'))

    model.add(Dropout(0.5))

    model.add(Dense(num\_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical\_crossentropy,

                  optimizer=keras.optimizers.Adadelta(),

                  metrics=['accuracy'])

    datagen = ImageDataGenerator(

        featurewise\_center=False,  # set input mean to 0 over the dataset

        samplewise\_center=False,  # set each sample mean to 0

        featurewise\_std\_normalization=False,  # divide inputs by std of the dataset

        samplewise\_std\_normalization=False,  # divide each input by its std

        zca\_whitening=False,  # apply ZCA whitening

        rotation\_range=10,  # randomly rotate images in the range (degrees, 0 to 180)

        width\_shift\_range=0.1,  # randomly shift images horizontally (fraction of total width)

        height\_shift\_range=0.1,  # randomly shift images vertically (fraction of total height)

        horizontal\_flip=True,  # randomly flip images

        vertical\_flip=False)  # randomly flip images

    history = model.fit\_generator(datagen.flow(a,b, batch\_size=32),

                        steps\_per\_epoch=len(a) / 32, epochs=epochs, validation\_data = [c, d],callbacks = [MetricsCheckpoint('logs')])

    score = model.evaluate(c,d, verbose=0)

    print('\nKeras CNN #1C - accuracy:', score[1],'\n')

    y\_pred = model.predict(c)

    map\_characters = dict\_characters

    print('\n', sklearn.metrics.classification\_report(np.where(d > 0)[1], np.argmax(y\_pred, axis=1), target\_names=list(map\_characters.values())), sep='')

    Y\_pred\_classes = np.argmax(y\_pred,axis=1)

    Y\_true = np.argmax(d,axis=1)

    plotKerasLearningCurve()

    plt.show()

    plot\_learning\_curve(history)

    plt.show()

    confusion\_mtx = confusion\_matrix(Y\_true, Y\_pred\_classes)

    plot\_confusion\_matrix(confusion\_mtx, classes = list(dict\_characters.values()))

    plt.show()

    model.save(MFILE)

runKerasCNNAugment(X\_train,y\_trainHot,X\_test,y\_testHot,1)

10.2. DISEASE PREDICTION

import keras

from keras.models import load\_model

from keras.preprocessing import image

from PIL import Image

import cv2

import numpy as np

import os

#image folder

folder\_path='test\_fyp/test\_fyp1'

#path to model

model\_path='deeps\_wbc1.h5'

img\_width,img\_height=60,80

model=load\_model(model\_path)

images=[]

for filename in os.listdir(folder\_path):

  imagepath=folder\_path + "/" + filename

  img=image.load\_img(imagepath,target\_size=(img\_width,img\_height))

  x=image.img\_to\_array(img)

  x=np.expand\_dims(x,axis=0)

  input\_image=np.vstack([x])

  images.append(input\_image)

for i in images:

  classes=model.predict\_classes(i,batch\_size=10)

  if 1 in classes:

    print('Neutrophil')

  elif 2 in classes:

    print('Eosinophil')

  elif 3 in classes:

    print('Monocyte')

  else:

    print('Lymphocyte')

  #print(classes)

CHAPTER 11

CONCLUSION

CHAPTER 11

CONCLUSION

Thus we have successfully implemented the project titled ‘Disease Detection Using Quantification Of Wbc Count’. The user had to input the patient’s details such as name , gender ,age and a set of twelve images of the blood smear samples. The output obtained was classified images into its wbc types, their count and the most probable disease that the patient might be suffering from. This project will help the doctors to follow a path in a particular direction to check what the patient is suffering from. The results obtained have and an accuracy of about 80% and hence the system is quite efficient and trustworthy to rely on. Therefore, the successfully completion of this project can help the doctors , pathologists and patients to a great extent.

REFERENCES

[1][www.wikipedia.com](http://www.wikipedia.com)

[2] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3996871/>

[3] <https://www.seminarsonly.com/Engineering-Projects/Computer/disease-identification-using-WBC-count.php>

[4] <https://www.ncbi.nlm.nih.gov/books/NBK261/>

[5] <https://www.healthline.com/health/wbc-count>

[6] <https://emedicine.medscape.com/article/2054452-overview>

[7] <https://medlineplus.gov/ency/article/003643.htm>

[8] <https://www.verywellhealth.com/white-blood-cell-wbc-count-1942660>

[9] <https://en.wikipedia.org/wiki/Reference_ranges_for_blood_tests>

[10] <https://labtestsonline.org/tests/white-blood-cell-count-wbc>

[11] Disease Detection using Blood Smear Analysis by Pragati Sharma

ACKNOWLEDGEMENT

Success of a project like this involving high technical expertise, patience and massive support of

guides, is possible when team members work together. We take this opportunity to express our

gratitude to those who have been instrumental in the successful completion of this project. We

would like to show our appreciation to Ms. Kavita Wale for her tremendous support and help,

without her this project would have reached nowhere. We would also like to thank our project

coordinator Ms. Rakhi Kalantri for providing us with regular inputs about documentation and

project timeline. A big thanks to our HOD, Dr. Lata Ragha for all the encouragement given to

our team. We would also like to thank our principal, Dr. S. M. Khot, and our college, Fr. C.

Rodrigues Institute of Technology, Vashi, for giving us the opportunity and the environment to

learn and grow.

1. Ashley Antony 101504

2. Mrinali Sonawne 101558

3. Prajakta Wani 101564

4. Deepshikha Zutshi 101565