AI-ENABLED PNEUMONIA DETECTION SYSTEM

A Thesis presented to the Department ofComputer Science and Engineering

African University of Science and Technology
Abuja, Nigeria.

In partial fulfillment of the Requirements for the Degree of Masters of Science in Computer science

By

[40809] Ogbeh Blessing Okwudo

2021.

CERTIFICATION

This is to certify that the thesis titled AI-enabled pneumonia detection system submitted to the school of postgraduate studies, African University of Science and Technology (AUST), Abuja, Nigeria for the award of theMaster's degree is a record of original research carried out by Ogbeh Blessing Okwudo in the Department of Computer Science.

African University of Science and Technology [AUST]

Knowledge is Freedom

AI-ENABLED PNEUMONIA DETECTION SYSTEM

By

Ogbeh Blessing Okwudo

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED

Supervisor, Prof. Ben Abderazek Abdallah

.....

Head of Department, Prof. Rajesh Prasad

APPROVED

.....

Chief Academic Office

COPYRIGHT

ABSTRACT

The world is currently facing the global pandemic of the Coronavirus disease (COVID-19) disrupting large part of the world, the number of patients has grown progressively. For dealing with this serious emergency, accurate diagnosis and fast reporting are two significant mechanisms.

The conventional medical system is facing many challenges, such as inefficient diagnosis procedure, uncoordinated management, lacking of a fast reporting and response platform.

We aim to implement an AI-Enabled Real-time software and hardware platform for Pneumonia detection and diagnosis.

We propose in this work a smart biomedical detection/diagnosis system. Particularly, we design a smart software. Furthermore, we develop a deep learning (DL)-based system for automatic segmentation and diagnosis. Based on the proposed system, users can conveniently upload their chest X-ray images for automatic diagnosis and returning timely results.

Keywords: Deep Learning, CNN, Image Classification, Pneumonia Detection

DEDICATION

I dedicate this thesis to Almighty God who has been there for me from the start to the finish.

ACKNOWLEDGMENTS

I am grateful to God Almighty for bringing me to this level in my career, to my familyfor their continuous support throughout this work and to my husband for his undeniable support and push.

I like to appreciate AUST and the African Development Bank for sponsoring me through this degree.

Most of all I would like to thank my supervisor Prof Ben Abdallah for his guidance, support, incredible patience throughout the course of my work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
DL	Deep Learning
FDL	Federated Deep Learning
UI	User Interface
CNN	Convolution Neural Network
AI	Artificial Intelligence
COVID-19	Corona Virus Disease 2019

Chapter One

Introduction

1.0 Background of the Study

Researchers from multiple disciplines across the globe are currently burning the nights oil trying to figure out the solution to the COVID-19 pandemic which has hit the world in 2019 and has come with various strains and a very fast transfer rate making it difficult to curtail. One of the ways to reach this solution is quick testing of a patient with indicating symptoms and effective management options as there seems to no cure at the moment.

Several testing methods have been tried with not a sufficiently accurate result in terms of sensitivity to the virus and specificity [3]. A considered method that has proven to have quite a good measure of accuracy is the image-based diagnosis method for the COVID-19 in which CT images or chest X-ray images are investigated for traces of the virus. This can be time consuming and that is where this work comes in.

In this work, we implore a deep learning (DL) algorithm to detect the presence of the virus in a chest X-ray image. This is done through the concept of a deep dream so that it also differentiates between a chest X-ray infected with COVID-19 from one with any other pneumonia.

1.1 Statement of the Problem

With the increasing number of patients diagnosed with COVID-19, we see that the virus is disrupting a large part of the world. This increase comes along with some challenges such as lack of accurate or inefficient diagnosis procedure, lack of fast reporting which may lead to a patient's condition worsening before proper diagnosis and possible management is carried out.

Inefficient Diagnosing Procedure

Many patients displaying various symptoms which may be escalating fast (considering the life span of the disease) all queue up to see a consulting doctor with their X-ray images are attended to one-by-one. This takes time.

The proposed system seeks to establish a platform where each patient can upload his/her X-ray image ang get an immediate diagnosis (more accurate and detailed).

1.2 Justification of the Study

In achieving or providing a system that better handles/takes care of the problem of lack of accurate or inefficient diagnosis procedure and fast reporting, the COVID-19 disease can be better managed as less patients will get to complicated stages of the disease since management procedure has begun on the patient at an earlier stage. Achieving the system will also lower spread of the disease as a patient can take better measure to quarantine since he/she has been swiftly diagnosed.

1.3 Scope of the Study

This work covers fetching (sourcing and collection) and cleaning (removal of wrong data, unclear data, distorted data, normalizing the data and resizing the data) of image data. Using said data to train a convolution neural network to classify the images as accurately as possible and then passing the params of the trained model to a user interface for easy use.

1.4 Aims and Objectives

In this work, we aim to implement an AI-enabled real-time software for the accurate detection and diagnosis of COVID-19 disease. We design a software system, develop a deep learning (DL) based system for automatic segmentation and diagnosis where users can conveniently upload their chest X-ray images for automatic diagnosis and timely return of the results.

1.5 Limitations

This work is limited to classification and diagnosis of a chest X-ray image of a patient.

1.6 List of Abbreviations

ML	Machine Learning
DL	Deep Learning
FDL	Federated Deep Learning

UI	User Interface
CNN	Convolution Neural Network
AI	Artificial Intelligence
COVID-19	Corona Virus Disease 2019

Chapter Two

Literature review

2.0 Introduction

In this chapter, we review work done in the past relating to image classification with neural networks particularly using convolutional neural network (CNN). Also the graphical user interface.

2.1 Basic Concept and Terminology

Classification of images can be seen as a process of categorizing and labelling groups of pixels or vectors within an image based on specific rules [1]. It can also be said to be a process of assigning all pixels in the image to a particular class or theme based on spectral information represented by the digital numbers [9]. Medical image classification is a two-step process. First, feature extraction techniques are used to obtain visual features from image data and secondly, machine learning algorithms that use these features to classify images into defined groups or classes is applied [2]. Among different features that have been used, shape, edge, and other global features are the commonly trusted ones [2]. One of such machine intelligence algorithms is the convolution neural network algorithm which we have used in this work.

2.2 Neural Network for Image Classification

A neural network [Fig2.1] is a system of neurons, either organic or artificial in our case, thattries using a series of algorithms to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. To a computer, data is recognized as numbers. Image data is made of pixels and each pixel can be represented by a number, so the image is seen just as a 2-dimensional matrix of numbers [Fig 2.2].

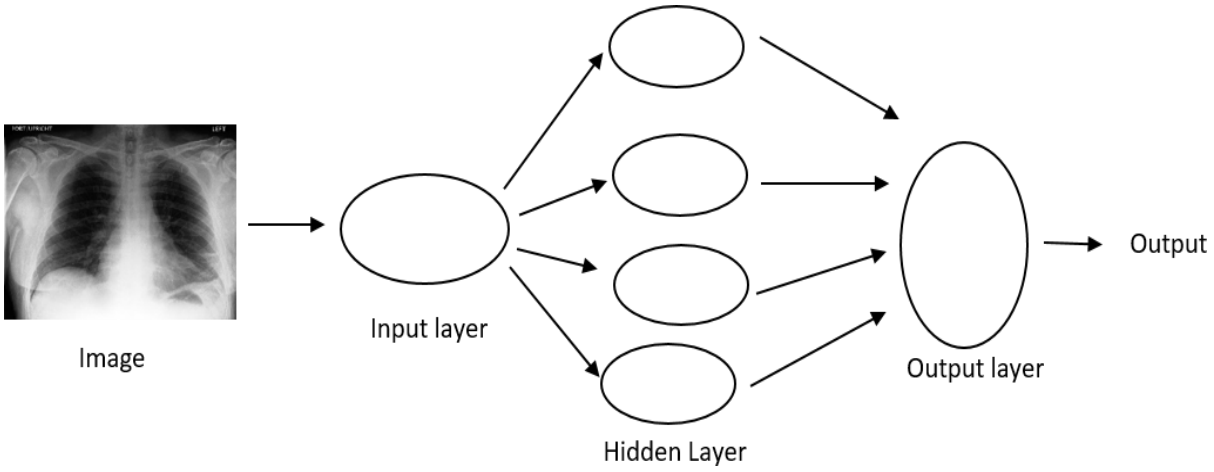


Fig 2.1: A simple Neural Network

2.3 Convolutional Neural Network (CNN)

CNN is a deep learning algorithm that is used for the classification of images. It takes an image as input and assigns importance to various features of the image. CNN is able to capture spatial and temporal dependencies in an image through the application of relevant filters. To a computer, images are numbers. An image is made of pixels, and each pixel can be represented by a number, so the image is just as a

2-dimensional matrix of numbers [Fig 2.2].

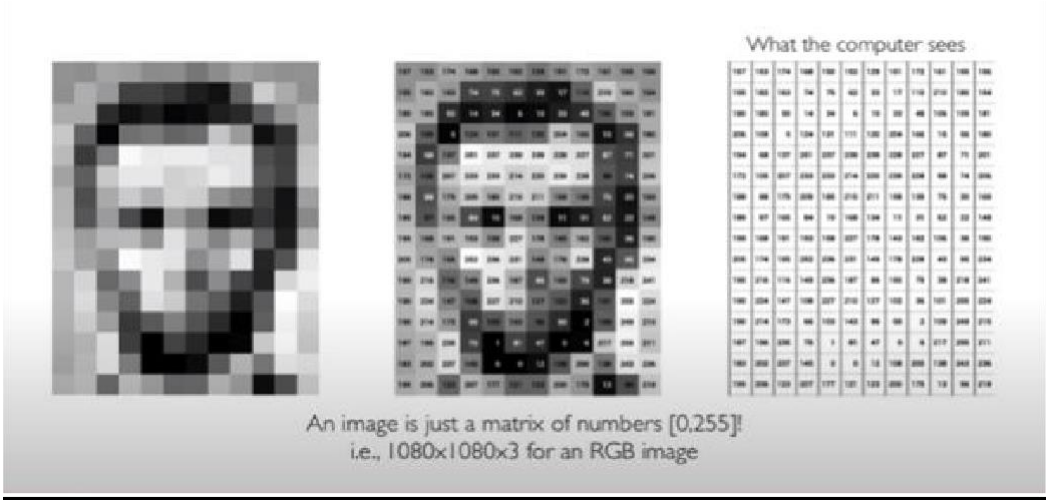


Fig 2.2: Pixelated image seen as a matric [12]

Classification: Output variables take class label and can produce probability of belonging to a particular class. Classification can be done by detecting unique features of an image in a class.

Manual extraction of these features is brittle, so we use Neural networks to learn a hierarchy of features (low level – edges and dark spots, mid-level – ears, nose, eyes of a face, high level – facial structure) directly from the data.

2.3 Graphical User Interface (GUI)

A Graphical User Interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation [10]. It is a tool that is used for simplifying a computing environment. Theend user of the interface accesses computing resources by manipulation symbolic representations of

the services provided by those computing resources. The end user is no longer restricted to typing commands one line at a time [4].

2.4 Similar works

Similar research is being carried out in the Adaptive Systems Lab, University of Aizu, Japan on AI Enabled Real time Biomedical System (AIRBIS) [3]. The lab is implementing a similar system while perfecting accuracy at a hardware [5] level in order to improve diagnosisspeed and efficiency.

Chapter Three

Implementation

3.0 Introduction

In this chapter, we discuss the methodology and the algorithm we have used in the course of this work to achieve the aims and objectives of the work. The algorithm was used because of the better effect it has on the dataset with which we are working (images). The general approach for solving classification problems first training a set whose classification labels are already known using a model and then testing the actual performance of the model with a set of data whose labels are previously unknown to the classification model. In this work, we have used a software approach to try and solve the problems stated in section 1.1. The model used was implemented using python programming language in Spyder IDE, imploring keras and Tensorflow frameworks. A GPU was not used in the training of the model used, because of unavailability.

3.1 Convolutional Neural Networks Model

A special kind of Artificial Neural Network (ANN) that uses convolution in place of general matrix multiplication in order to maintain spatial quality of an input variable. It consists of layers: convolution layer(s), pooling layer, fully connected layer (one or more hidden layers), classification layer (activation function) see Fig 3.1. This allows the CNN represent highly non-linear functions.

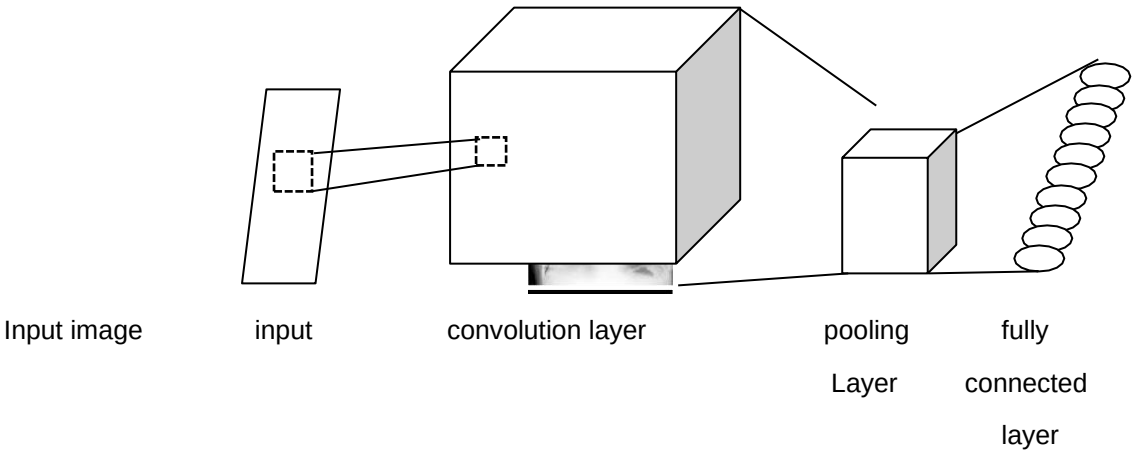
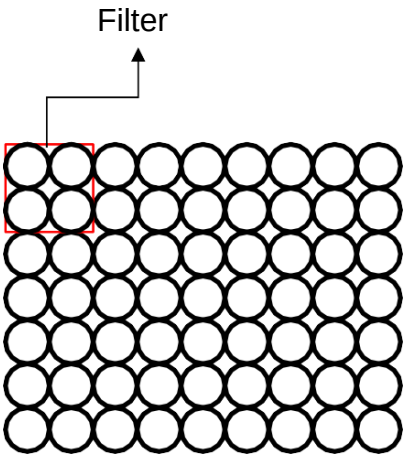


Fig 3.1: Layers of a CNN model

Convolution Layer:

The spatial structure lost with the typical fully connected neural network (ANN), can be rectified. We connect patch or region (also known as Filter) of the image to the neurons as input. We connect this patch (as seen below) into input layer to a single neuron in subsequent layer. We slide this across the input image and weigh each of these pictures to detect/extract the feature.

This 'patchy' operation is called **Convolution.**



A dot product of both the filter matrix and covering image portion is performed and its result is added to the bias or weight. This output is placed in the appropriate matrix position. See Fig 3.2

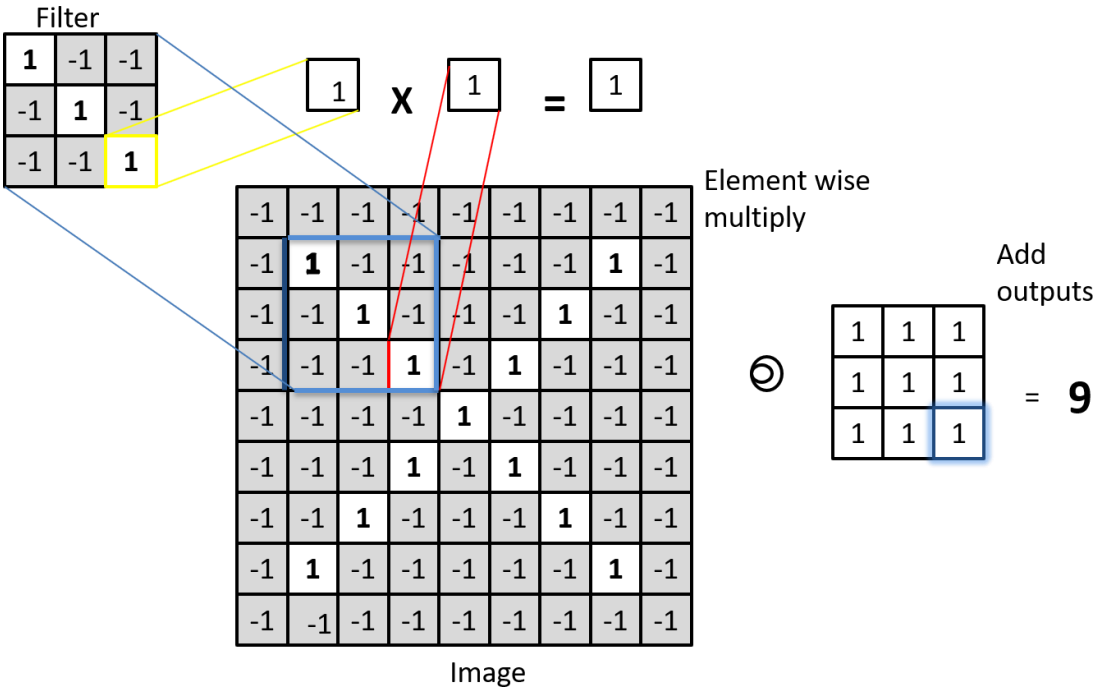


Fig. 3.2: The Convolution Operation

We slide the 3 x 3 filter over the input image, element wise multiply and addthe outputs as seen below in Fig 3.3.

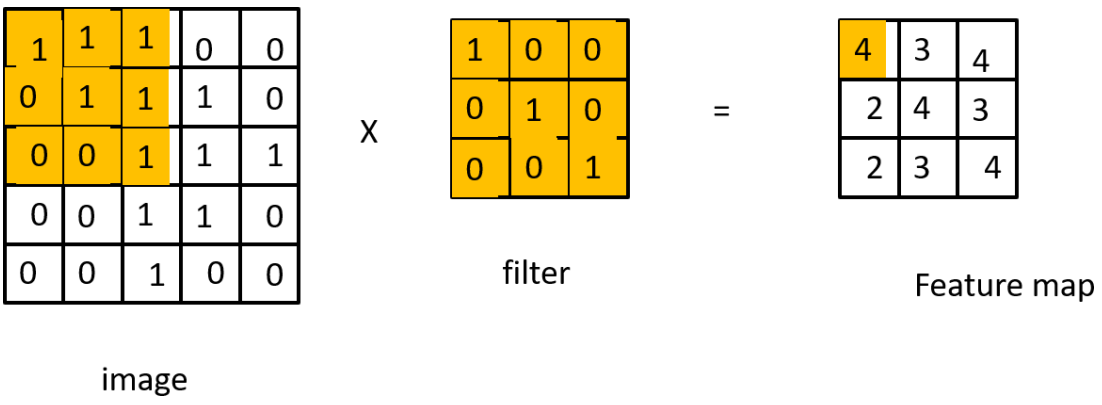


Fig. 3.3: The Convolution Operation 2

Pooling Layer:

This layer allows the downsizing of the image and deal with multiple scale of the image features. This can be done on any layer after every convolution layer. The common types are Max Pooling and Average Pooling. An illustration is seen in Fig 3.4 below.

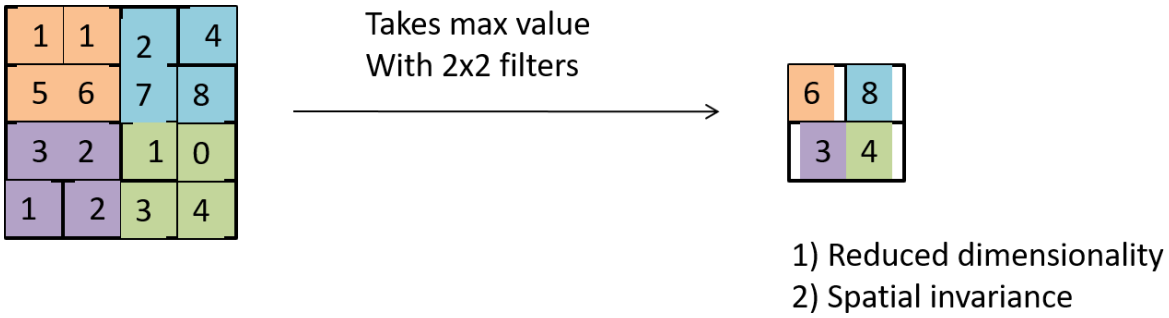


Fig. 3.4: Max Pooling using a 2x2 Filter matrix

We stack convolution, nonlinearity with pooling operations and repeat till wehave high level features of the image.

Fully Connected Layer:

The learned features of the image are then fed a fully connected networkwhich will output a categorical distribution.

3.1.1 Model Evaluation

We have used a large sample data of 9544 images extracted from a public dataset [4], Kaggle, containing both COVID diagnosed and non-COVID diagnosed images for this work. The model adopted uses a sequential method. We have added two blocks of

convolution and max pooling layers with the rectified Linear unit (ReLu) activation function, each block having 64 nodes and a 4x4 filter size chosen after much tuning for the best result. The output of the blocks is then fed to a flattened dense layer of 64 nodes whose output is finally fed to a dense layer of 1 node with the sigmoid activation function for classification. We have compiled the network using binary cross entropy to check loss and adam optimizer under the metrics of accuracy. The network was run for 10 epochs. Our 9544-image dataset was split 90% for training and 10% for testing, it was fed with a batch size of 64 images, shuffling each time. The image size was reduced to 28x28 pixels for both training and testing sets due to memory constraints of the CPU it was being run on. The network was run on a Pentium 2.0GHz processor with 4GB of memory.

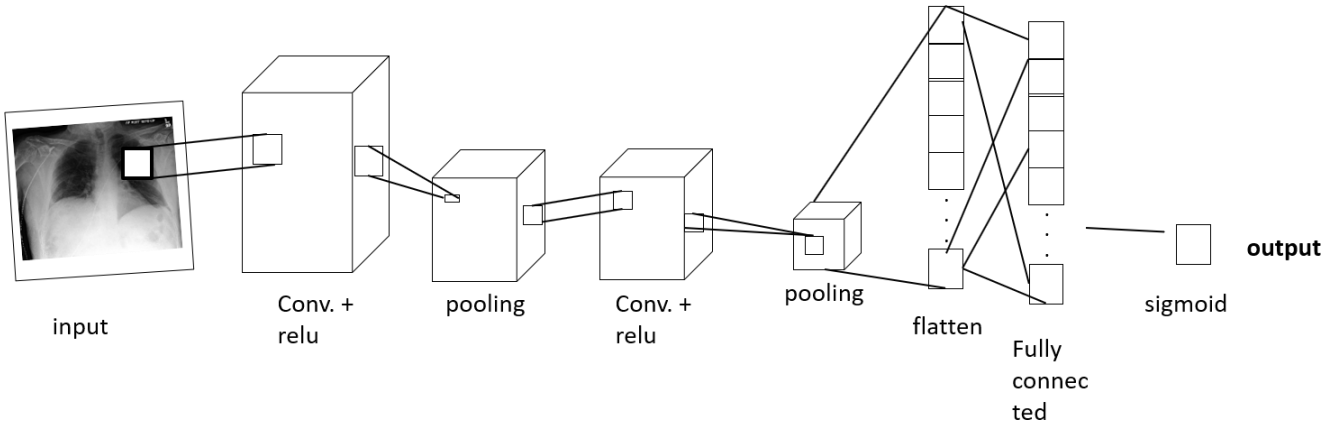


Fig. 3.5: CNN Model

3.2 Design of the interface

In this section, we discuss the user interface that we have created with python's TKinter. The user interface is intended to display a patient's diagnosis based on the COVID-19 detection as well as the patient's information. The interface (displayed in Fig 3.6) contains the following functions: Sample selector, Sample of interest, Detailed information,

Diagnosis result, Monitor patient Status.

3.1.1 Sample Selector

A new patient can either enter their details which is name, age, gender and chest X-ray image for diagnosis or an existing user enters their ID to get a history of their recent engagements.

3.1.2 Sample of interest

This displays a chest X-ray image of the patient whose ID number has beenentered in the sample selector.

3.1.3 Detailed Information

When the patient's ID has been entered, the patient's details such as ID,name, age and gender is displayed in this function.

3.1.4 Monitor patient Status

This function displays the results so far of all the patients entered in the system. The ID of each patient and corresponding diagnosis. For full detailsof a patient's diagnosis, the ID of that patient can be entered at the sample selector section.

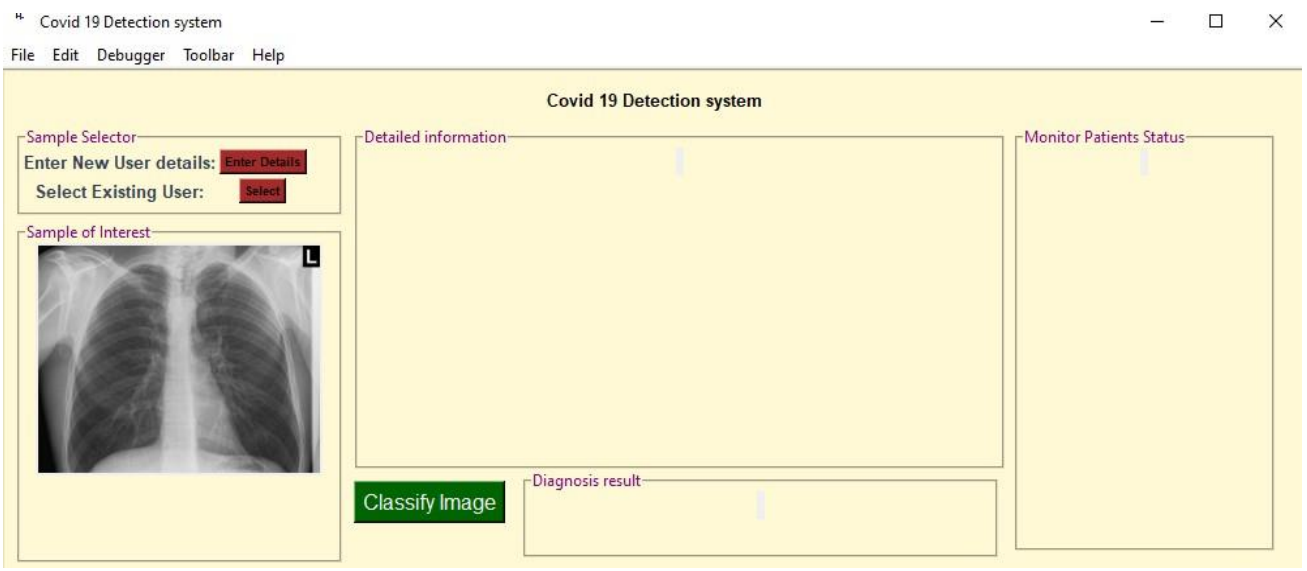


Fig: 3.6 User Interface

3.2 Conclusion and Future Work

The data input of a given patient is to be stored and managed in a database, so when users use it, all they need do is enter a user ID and get a retrieval of the information imputed and in addition, a diagnostic result. This database management system has not been successfully executed and added to the system due to inadequate time, and some other API challenges encountered that time will not permit to tackle in the course of this work. This however, will be considered in a future work.

Chapter Four

Results

4.1 Introduction

In this chapter, we present the results of our proposed system implementation. We examine the accuracy of our model, and make comparisons between different filter sizes and several activation functions and display the accuracy measure of the model output. The accuracy values are plotted over 10 epochs inthe figures below. Finally, we draw conclusions based on the outcome of the model designs.

Table 4.3: Table of results for filter tunning

Size of Filters On Conv Layer	Epoch	Training Loss (%)	Training Accuracy (%)	Validation Loss (%)	Validation Accuracy (%)
3 x 3	10	22.93	89.99	29.75	86.91
4 x 4	10	25.07	89.22	26.35	89.74

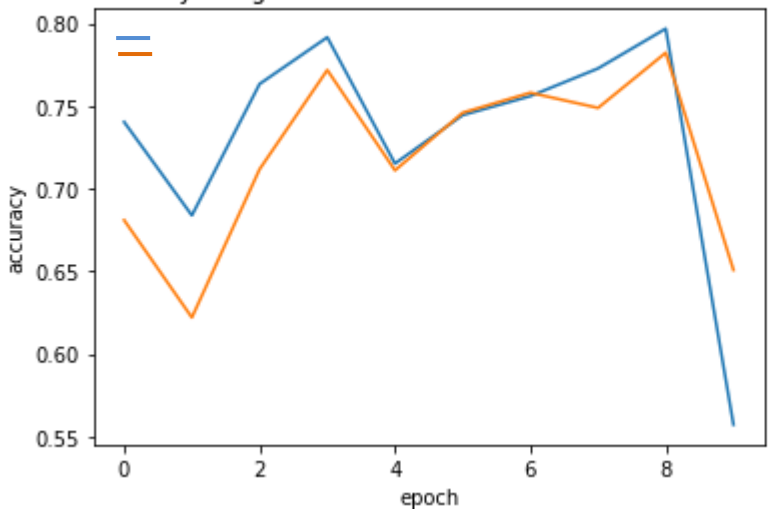
Table 4.2: Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 64)        640
_________________________________________________________________
activation (Activation)      (None, 26, 26, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
activation_1 (Activation)    (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 32)          18464
_________________________________________________________________
activation_2 (Activation)    (None, 9, 9, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2592)              0
_________________________________________________________________
dense (Dense)                (None, 64)                165952
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 222,049
Trainable params: 222,049
Non-trainable params: 0
```

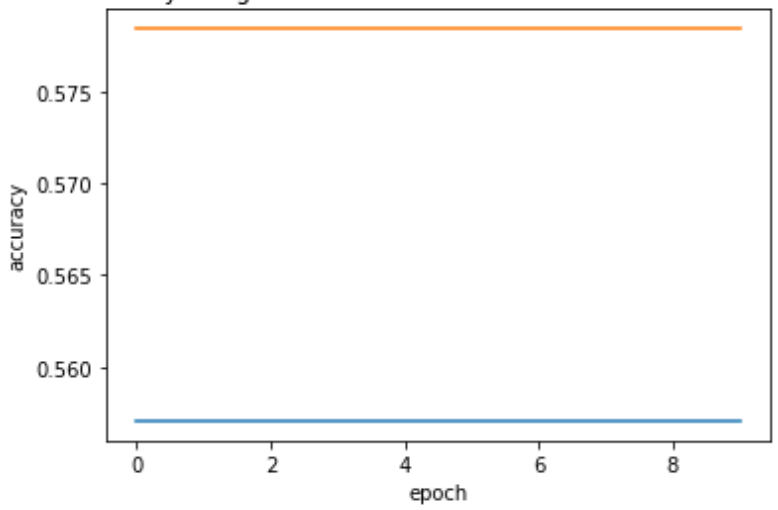
Table 4.3: Table of results for activation function tunning

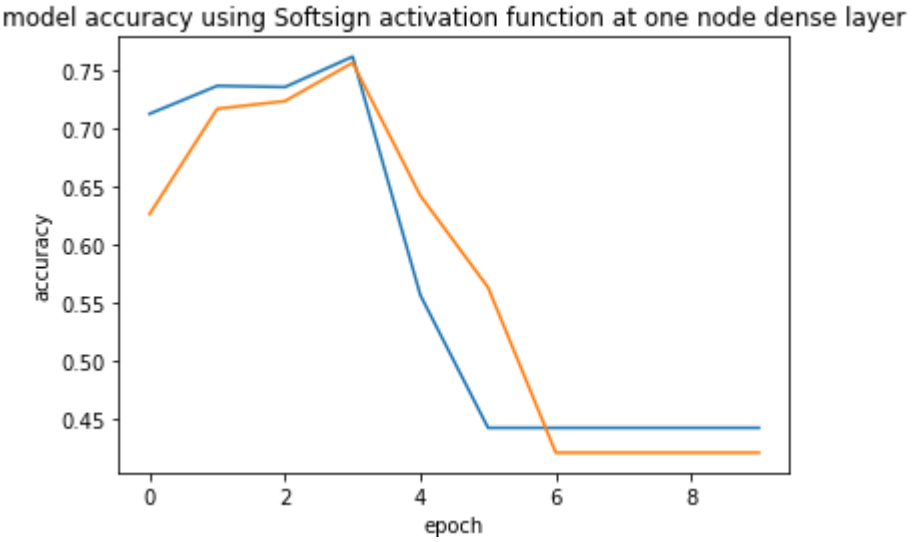
Activation functions At 1 node dense layer	**Epoch**	**Training Loss**	**Training Accuracy (%)**	**Validation Loss**	**Validation Accuracy (%)**
ReLu	10	6.4289	57.84	6.7544	55.71
Softmax	10	6.4289	57.84	6.7544	55.71
SoftSign	10	8.9220	42.16	8.5927	44.29
Sigmoid	10	0.2406	89.77	0.2748	89.01

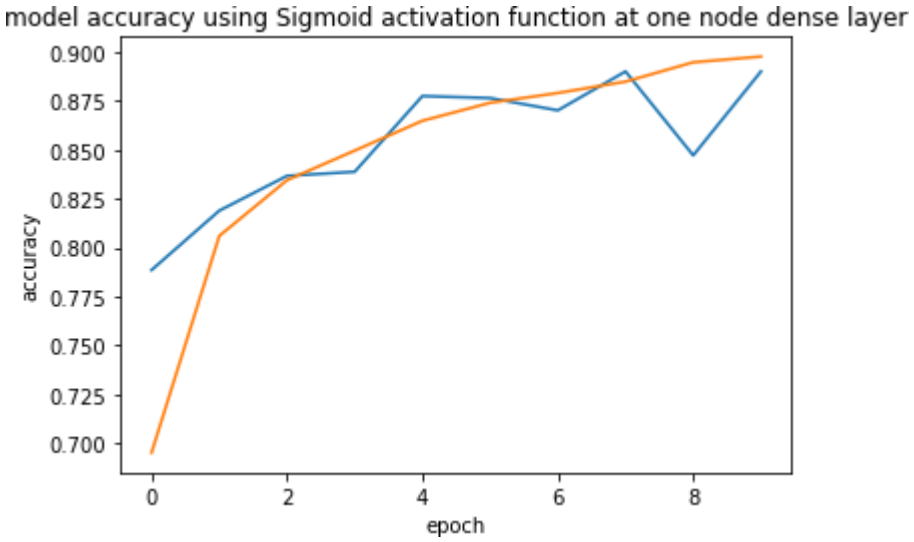
model accuracy using relu activation function at one node dense layer



model accuracy using Softmax activation function at one node dense layer







4.2 Result Discussion

In this work, the focus was on performance (accuracy) of the classifying model seeing that we are dealing with health diagnosis. The implication will be expensiveif we have a model performing at a sub-par accuracy percentage. Detailed summary of the results achieved is seen in the table (Table 4.1) above. We can see that both models classify quite well with a better validation (test) accuracy

seen when a filter size of a 4 x 4 matrix after 10 epochs. The model has not overfiteither as we see the difference in train and test accuracy is quite insignificant. Theabove result suggests that for every 10 images given to the proposed model for classification, only 1 may be incorrectly classified. In other words, 1 in 10 chest X- ray images, may be wrongly diagnosed as a COVID image or not as the case maybe. In figures 4.1 through 4.4, we see the performance of the model when we tweaked the activation functions at the one node dense layer between the ReLu, Softmax, Softsign and the Sigmoid activation functions. The result of the Sigmoid activation function at that layer proves more reliable since the graph shows an ascending trend as compared to the the rest of the action functions. We can say that the best performing for our model given the dataset is the sigmoid activationfunction.

Chapter Five

Conclusion and Future work

5.0 Introduction

In This chapter, we summarize the work we have done so far; the presented problem statement, the method we have used in combating the problems, the results reached and then we draw our conclusions based on the results. Also, we discuss the future work envisioned for a better approach at better results.

5.1 Conclusions

In this research, we proposed a neural network architecture and perform its evaluation. We employed the use of deep learning techniques to categorize chest x-ray images as either infected with COVID-19 or not infected. We have sourced available dataset from an online source, preprocess them by cleaning, dropping, cropping and normalizing them so we are left with quality data to train our neural network. Many network architecture designs with scalability was performed but the best performing one was picked for evaluation.

After the evaluation process, the proposed architecture with a 4x4 matrix filter size on the 2D conv layers was seen to have performed better as compared with a 3x3 matrix filter size on the 2D conv layers indicating that more features were extracted at the conv layer with the 4x4 filter matrix size. This performance was even better when paired with a sigmoid function at the one node dense classification layer. We also see the role the User interface plays in the time saving of a classification of an input image for diagnosis or classification as a use need not return to the model code before they can have their image classified. The UI uses the saved parameters of the best performing model according to our evaluation.

5.2 Future work

A large number of research areas pertaining to this research has not been able to be covered due to lack of time, inadequate resources (such as a GPU processor, an FPGA) among others. Highlighted below are some of interest.

1. Successfully integrating this User Interface to a database so that a user's details can be stored and retrieved when required.
2. Optimizing Execution time by reducing the number of operations taking place in the convolution layer. More so this will also reduce the computation complexity of the network and hence even the power consumption.
3. An effort to parallelize the Convolution Neural Network in order to accelerate the inference on an FPGA chip is needed as this may also increase the network speed to really achieve real time diagnosis.

4. Embracing a collaborative learning approach of the hardware and software will better the accuracy and give an even better accuracy which is required since we aim at a high accuracy for medical diagnosis.

REFERENCES

1. Abderazek Ben Abdallah, Huankun Huang, Nam Khanh Dang, Jiangning Song, “ＡＩ Processor " 特願2020-194733 (2020年11月24日), Patent, Japan.
2. Miyuka Nakamura, Jiangkun Wang, Sinchhean Phea, Abderazek Ben Abdallah, "Comprehensive Study of Coronavirus Disease 2019 (COVID-19) Classification based on Deep Convolution Neural Networks," 3rd ETLTC2021-ACM Chapter Int. Conference on Information and Comm. Technology, January 27-30, 2021, Aizu-Wakamatsu, Japan
   SHS Web of Conferences 102, 04007 (2021), DOI:10.1051/shsconf/202110204007
3. Sinchhean Phea, Zhishang Wang, Jiangkun Wang, Abderazek Ben Abdallah, "Optimization and Implementation of a Collaborative Learning Algorithm for an AI-Enabled Real-time Biomedical System," 3rd ETLTC2021-ACM Chapter Int. Conference on Information and Comm. Technology, January 27-30, 2021, Aizu-Wakamatsu, Japan SHS Web Conf., 102 (2021) 04017, DOI: 10.1051/shsconf/202110204017 (Best Paper Award)
4. Chest X-ray images https://Kaggle.com/khoongweihao/covid19-xray-dataset-train-test-sets ; https://github.com/ieee8023/covid-chestxray-dataset
5. Kamel Abdelouahab, Maxime Pelcat, Jocelyn Serot, Francois Berry. Accelerating CNN inference on FPGAs: A Survey. arXiv:1806.01683v1 [cs,DC] 26 may 2018.
6. Adaptive Systems Laboratory: https://www.u-aizu.ac.jp/~benab/ABALab/
7. M Shinozuka, B Mansouri. Synthetic aperture radar and remote sensing technologies for structural health monitoring of civil infrastructure systems 2009
8. Nisar Wani, Khalid Raza in Soft Computing based medical image Analysis 2018
9. ORY Six Diagnosing COVID-19 using AI-based medical image analysis March 2020
10. Susan Q. Sherrick An introduction to graphical user interfaces and their use by CITIS

11. Abderazek Ben Abdallah, Huankun Huang, Nam Khanh Dang, Jiangning Song. AI-ENABLED PNEUMONIA DETECTION, ANALYSIS AND DIAGNOSIS SYSTEM 2020
12. http://introtodeeplearning.com

APPENDIX I

CODES FOR CNN MODEL

#loading dataset

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
```

In [2]:

```
Datadir = "C:/train/traindata"
CATEGORIES = ["covid", "non"]

for category in CATEGORIES:
    path = os.path.join(Datadir, category)   #path to image directory
    for image in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, image), cv2.IMREAD_GRAYSCALE)
        #plt.imshow(img_array, cmap="gray")
        #plt.show()
        break
    break
```

In [3]:

```
#print(img_array)
#print(img_array.shape)
```

In [3]:

```
image_size = 28

new_array = cv2.resize(img_array, (image_size, image_size ))
#plt.imshow(new_array, cmap="gray")
#plt.show()
```

In [4]:

```
#collecting training data

train_data = []

Datadir = "C:/train/traindata"
CATEGORIES = ["covid", "non covid"]

def training_data():
    for category in CATEGORIES:
        path = os.path.join(Datadir, category)   #path to image directory
        classification_num = CATEGORIES.index(category)
        for image in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, image),
cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (image_size, image_size ))
                train_data.append([new_array, classification_num])
            except Exception as e:
                pass

training_data()
```

In [5]:

```
print(len(train_data))
```

```
9544
```

In [6]:

```
import random       #to shuffle data
random.shuffle(train_data)
```

In [7]:

```
for img in train_data[:10]:
    print(img[1])
```

```
1
1
1
1
1
0
1
1
0
1
```

In [8]:

```
#packing data into variables to use

X = []
Y = []

for features, label in train_data:
    X.append(features)
    Y.append(label)

X = np.array(X).reshape(-1, image_size, image_size, 1)
Y = np.array(Y)
```

In [9]:

```
# saving the data

import pickle

pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("Y.pickle", "wb")
pickle.dump(Y, pickle_out)
pickle_out.close()
```

In [10]:

```
#to load back data


pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
```

In [11]:

```
#CNN

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, Conv2D, Flatten,
MaxPooling2D
```

In [12]:

```
#lets normalize the data

x = X/255.0
```

In [17]:

```
model = Sequential()
model.add(Conv2D(64, (4,4), input_shape = x.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (3,3)))

model.add(Conv2D(64, (4,4)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (3,3)))

model.add(Flatten())
model.add(Dense(64))

model.add(Dense(1))
model.add(Activation('sigmoid'))

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

model.fit(x, Y, batch_size=64, validation_split=0.1, epochs=10, shuffle=True)
```

In [19]:

```
model.save('bcnnmodel.h5')
```

In [13]:

APPENDIX II

CODES FOR USER INTERFACE

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
import cv2
import os
from PIL import Image, ImageTk
import numpy as np

#from keras.models import load_model

#import mysql.connector

#mydb = mysql.connector.connect(host="localhost", user="root",
    passwd="")



#model = load_model('bcnnmodel.h5')

classes = ['covid','non covid']



def enter_details():
```

```
new_window = Toplevel(window)
new_window.wm_iconbitmap(r'favicon.ico')
new_window.geometry("400x300")
new_window.title("User Details Entry")
new_window.resizable(False,False)
Name = Label(new_window, text = "Name:",
 font=('ariel',10,'bold'), foreground="#354157",
 justify='left')
Name.grid(row=0,column=0)
Entry1=Entry(new_window, font=20)
Entry1.grid(row=0,column=1)
Age = Label(new_window, text = "Age:",
 font=('ariel',10,'bold'), foreground="#354157",
 justify='left')
Age.grid(row=1,column=0)
Entry2=Entry(new_window, font=20)
Entry2.grid(row=1,column=1)
Gender = Label(new_window, text = "Gender:",
 font=('ariel',10,'bold'), foreground="#354157",
 justify='left')
Gender.grid(row=2,column=0)
Entry3=Entry(new_window, font=20)
Entry3.grid(row=2,column=1)
Blank = Label(new_window, text = "")
Blank.grid(row=3,column=0)
Image = Label(new_window, text = "Enter Image:",
 font=('ariel',10,'bold'),  foreground="#354157",
 justify='left')
Image.grid(row=4,column=0)
```

```
    upload_botton = Button(new_window, text="Upload Image",
    command=upload_image, font=('Helvetica', 8,
    'bold'),background="brown",foreground="black")
    upload_botton.grid(row=4, column=1)


    #Save_botton = Button(new_window, text="Save", command=,
    font=('Helvetica', 8,
    'bold'),background="brown",foreground="black")
    #Save_botton.place(relx=0.8, rely=0.8)




def upload_image():
    file_path = filedialog.askopenfilename()
    uploaded = Image.open(file_path)

     uploaded.thumbnail(((window.winfo_width()/2.2),(window.winfo_
     height()/2.2)))
    im = ImageTk.PhotoImage(uploaded)
    sign_image.configure(image=im)
    sign_image.image = im
    messagebox.showinfo('Upload Status', 'Image Uploaded
     Successfully')
    #label.configure(text= '')
```

```
    show_classify_button(file_path)


def show_classify_button(file_path):
    classify_btn = Button(window, text="Classify Image", font=10,
     background="darkgreen", foreground="white", command = lambda:
     classify(file_path))
    classify_btn.place(relx=0.27,rely=0.82)



def classify(file_path):
    img_size=28
    image = Image.open(file_path)
    image = image.resize((img_size,img_size))
    image = np.expand_dims(image, axis = 0)
    image = np.array(image).reshape(-1,img_size,img_size,1)
    predict = model.predict([image])
    sign = classes[int(predict[0][0])]
    print(sign)
    label4 = Label(window, font = ('ariel', 15, 'bold'), text=
     sign)
    label4.configure(foreground='#011638')
    label4.place(relx=0.4,rely=0.8,anchor='n')




window = tk.Tk()
window.title("Covid 19 Detection system")
window.wm_iconbitmap(r'favicon.ico')
window.geometry('1000x400')
```

```
#canvas = tk.Canvas(window, height=HEIGHT, width=WIDTH)
#canvas.pack()

menu = Menu(window)
window.config(menu=menu)
#menu items
file_menu = Menu(menu)
menu.add_cascade(label="File", menu=file_menu)
#file_menu.add_command(label="New")
file_menu.add_command(label="Exit", command=window.quit)

edit_menu = Menu(menu)
menu.add_cascade(label="Edit", menu=edit_menu)
#file_menu.add_command(label="New")
#file_menu.add_command(label="Exit", command=window.quit)

Debugger_menu = Menu(menu)
menu.add_cascade(label="Debugger", menu=Debugger_menu)
#file_menu.add_command(label="New")

toolbar_menu = Menu(menu)
menu.add_cascade(label="Toolbar", menu=toolbar_menu)
#file_menu.add_command(label="New")

Help_menu = Menu(menu)
menu.add_cascade(label="Help", menu=Help_menu)
#file_menu.add_command(label="New")
```

```
mainframe = LabelFrame(window, background="#FFF9D5")
mainframe.place(relwidth=1, relheight=1)

heading = Label(mainframe, text = "Covid 19 Detection system",
     font=('Helvetica', 10, 'bold'), background="#FFF9D5")
heading.place(relx=0.5,
     rely=0.01,relwidth=0.7,relheight=0.1,anchor='n')

frame1 = LabelFrame(mainframe, text="Sample Selector",
     background="#FFF9D5", foreground="purple")
frame1.place(relx=0.01, rely=0.11,relwidth=0.25,relheight=0.18)

label1=Label(frame1, text = "Enter New User details:",
     font=('ariel',10,'bold'), foreground="#354157",
     background="#FFF9D5")
label1.grid(row=0,column=0)
enter_details_button = Button(frame1, text="Enter Details",
     command=enter_details, font=('Helvetica', 7,
     'bold'),background="brown",foreground="black")
enter_details_button.grid(row=0,column=2)

label2=Label(frame1, text = "Select Existing User:",
     font=('ariel',10,'bold'),  foreground="#354157",
     background="#FFF9D5", justify='left')
label2.grid(row=2,column=0)
select_user_button = Button(frame1, text="Select",
     font=('Helvetica', 7,
     'bold'),background="brown",foreground="black")
select_user_button.grid(row=2,column=2)
```

```
frame2 = LabelFrame(mainframe, text="Sample of Interest",
    background="#FFF9D5", foreground="purple")
frame2.place(relx=0.01, rely=0.3,relwidth=0.25,relheight=0.69)
sign_image = Label(frame2)
sign_image.pack(padx=0.1,pady=0.1)

frame3 = LabelFrame(mainframe, text="Detailed information",
    background="#FFF9D5", foreground="purple")
frame3.place(relx=0.27, rely=0.11,relwidth=0.5,relheight=0.69)
sign_details = Label(frame3)
sign_details.pack(padx=0.1,pady=0.1)

frame4 = LabelFrame(mainframe, text="Diagnosis result",
    background="#FFF9D5", foreground="purple")
frame4.place(relx=0.4, rely=0.8,relwidth=0.365,relheight=0.18)
sign_result = Label(frame4)
sign_result.pack(padx=0.1,pady=0.1)

frame5 = LabelFrame(mainframe, text="Monitor Patients Status",
    background="#FFF9D5", foreground="purple")
frame5.place(relx=0.778, rely=0.11,relwidth=0.2,relheight=0.855 )
sign_mps = Label(frame5)
sign_mps.pack(padx=0.1,pady=0.1)
```

```
window.mainloop()
```