National College of Ireland

Detection of Polycystic Ovarian Syndrome using Convolutional Neural Network in conjunction with Transfer Learning Models

MSc Research Project
Data Analytics

Nithya Sathiadhas Puvaneswari
Student ID:x21116270

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Nithya Sathiadhas Puvaneswari
Student ID:	x21116270
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Bharat Agarwal
Submission Due Date:	15/12/2022
Project Title:	Detection of Polycystic Ovarian Syndrome using Convolutional Neural Network in conjunction with Transfer Learning Models
Word Count:	1217
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Akash Parapurath Govindarajan
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of Polycystic Ovarian Syndrome using Convolutional Neural Network in conjunction with Transfer Learning Models

Nithya Sathiadhas Puvaneswari
x21116270

1 Introduction

The Polycystic Ovary Syndrome (PCOS) dataset was obtained from Kaggle [1]. The dataset contains two different picture folders for evaluating and training the algorithm. The goal of this thesis is to use transfer learning and deep learning techniques to develop a model that aids in the detection of PCOS.

2 Hardware Configuration

The implementation of the detection of PCOS is carried out on the desktop with the configurations mentioned in Table 1.

Package	Version
Processor	Apple M1 Pro chip
Memory	16GB
Storage Size	512GB
Google Colab	2vCPU @ 2.2GHz

Table 1: Hardware Configuration

3 Software Requirements

The dataset downloaded from Kaggle is uploaded to google drive and retrieved during the data processing stages. Different python packages are used to build the model, and to build a model; Google Colab is used. In order to test the model, a standalone application is created using the Tkinter and Pillow packages from Python. Visual Studio Code is used to create the graphical user interface. The following are some software and packages used to build our project.

[1] Dataset:`https://www.kaggle.com/datasets/anaghachoudhari/pcos-detection-using-ultrasound-images`

Package	Version
Python	3.10.8
keras	2.10.0
tensorflow	2.10.0
h5py	3.7.0
pandas	1.5.1
pillow	9.3.0
scipy	1.9.3
scikit-learn	1.1.3
pathlib	1.0.1
numpy	1.23.4
Tkinter	3.9

Table 2: Python Packages and Version

3.1 Tensorflow Installation

Tensorflow is developed by Google and is an open-source library developed by Google. It can be used to investigate the effects of various activation functions on categorization outcomes Ertam and Aydın (2017). Installing TensorFlow in MAC M1 was a tedious process. For the implementation part, TensorFlow was installed using a thirdparty package, miniforge3. The installation was done by following the steps mentioned in tutorial [2].

3.2 Google Colaboratory

Google Colaboratory, often known as 'Google Colab,' is a cloud-based service that provides a notebook for Python execution by configuring GPU and TPU settings Carneiro et al. (2018). The dataset present in the google drive can be retrieved directly from colab using python code.

3.3 VisualStudio Code

Visual Studio Code is a code editing tool used in the development process to provide a rapid code-build-debug cycle. VS Code is used in the implementation part as a presentation layer to show the results of input images.
Software Download Link: `https://code.visualstudio.com/download`

4 Code Explanation

4.1 Establish a Connection with Google Drive

The dataset used for implementing PCOS detection is stored in google drive. So the first step in the coding is to connect google drive with the google colab, as shown in Figure 1.

A dialog box appears, as shown in Figure 2, requesting permission to connect to the

[2]https://www.youtube.com/watch?v=WFIZn6titnct=780s

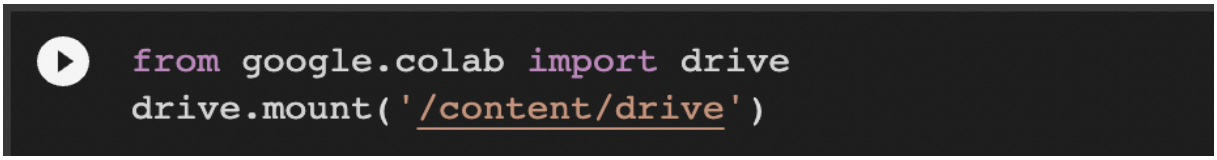

Figure 1: Google Colab Connection

google drive associated with the colab account. Click "Connect to Google Drive" to establish a successful connection.

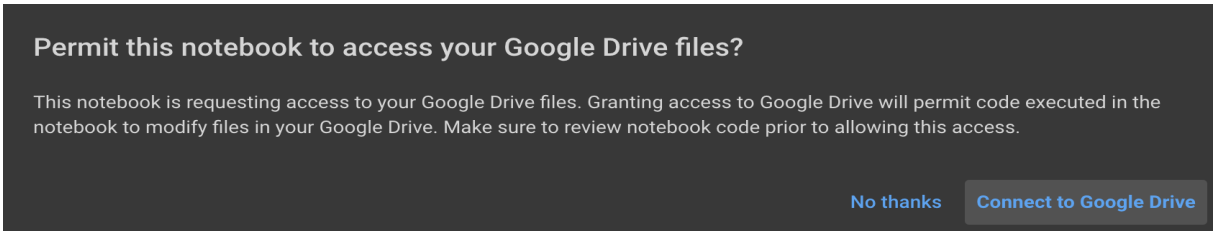

Figure 2: Permission Dialog

4.2 Import Packages

The required packages are imported into the next cell, including pandas, NumPy, Keras, and TensorFlow, as shown in Figure 3.

4.3 Data Loading

Once the connection is established, the next stage is to load the data. As the dataset has two folders separately for testing and training, both are loaded to different variables, as shown in Figure 4.

The original image is resized to an image of size 150 as the models accept this dimension of images as input.

4.4 Data Visualization

In the data visualization part, a few images are displayed randomly using the for loop in Python, as shown in Figure 5. In addition, the boxplot is used to display the number of infected and non-infected images in both test and train folders, as shown in Figure 6. It is quite clear from Figure 7 that the training data has over 1000 non-infected cases, whereas the infected cases are above 700.

4.5 Label Encoding

Datasets were divided into train and test in a ratio of 90:10 with a random state of 28, as shown in Figure 8.

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
import cv2
from sklearn.model_selection import train_test_split
from tqdm import tqdm
from PIL import Image
import io
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB2
from keras.layers import GlobalAveragePooling2D, Dropout, Dense
from keras.models import Model
from tensorflow.keras import layers
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, TensorBoard, ModelCheckpoint
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization, AveragePooling2D, GlobalAveragePooling2D
from sklearn.metrics import classification_report, confusion_matrix
from IPython.display import display, clear_output
import ipywidgets as widgets
from tensorflow.keras.utils import plot_model
from sklearn.metrics import confusion_matrix, classification_report
import itertools
from tensorflow.keras import layers ,models,optimizers
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.applications.vgg19 import VGG19
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import RMSprop,Adam,SGD
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import EarlyStopping
```

Figure 3: Imports

```python
labels = ['infected', 'notinfected']
X_train = []
Y_train  = []
X_test = []
Y_test = []
image_size=150 #sizeof image

for label in labels:
    trainPath = os.path.join('/content/drive/MyDrive/pcos_classification/Data/train',label)
    for file in tqdm(os.listdir(trainPath)):
        image = cv2.imread(os.path.join(trainPath, file))
        image = cv2.resize(image, (image_size, image_size))
        X_train.append(image)
        Y_train.append(label)

    testPath = os.path.join('/content/drive/MyDrive/pcos_classification/Data/test',label)
    for file in tqdm(os.listdir(testPath)):
        image = cv2.imread(os.path.join(testPath, file))
        image = cv2.resize(image, (image_size, image_size))
        X_test.append(image)
        Y_test.append(label)

X_train = np.array(X_train)
X_test = np.array(X_test)
```

Figure 4: Data Loading

4.6 Data Splitting

The dataset is split into test and train in the ratio of 90:10 with a random state of 28, as shown in Figure 9.

```
plt.figure(figsize=(15, 15))
class_names = Y_train
for i in range(0,10):
    for i in range(10):
        ax = plt.subplot(6, 6, i + 1)
        plt.imshow(X_train[i])
        plt.title(Y_train[i])
        plt.axis("off")
```

Figure 5: Visualization

```
sns.countplot(Y_train)
```

Figure 6: countplot

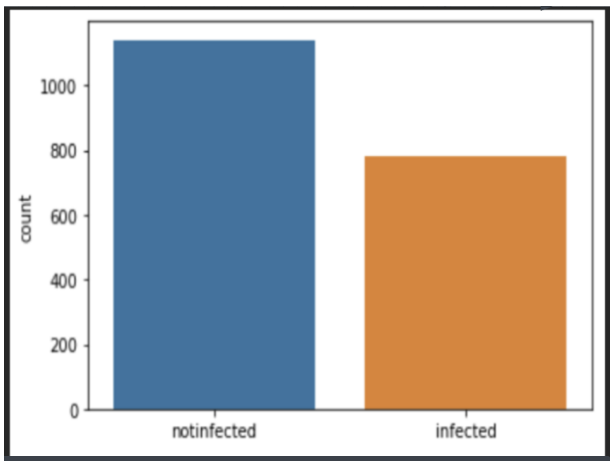

Figure 7: encoding

4.7 Data Augmentation

Data augmentation approaches are especially beneficial for building a robust picture classifier from limited training data. As shown in Figure 10, image augmentation is performed only on the training images with a rotation range of 30, and the fill mode was " nearest".

5 ResNet-50

The resnet-50 model was built starting with two convolutional 2D layers followed by a batch normalization and a max pooling 2D layer. 'Relu' activation is used in the modelbuilding process except for the "softmax" activation implemented in the last dense

```
[ ]  #label encoding
     y_train_ = []
     for i in Y_train:
         y_train_.append(labels.index(i))
     Y_train = y_train_

     Y_train = tf.keras.utils.to_categorical(Y_train)

     y_test_ = []
     for i in Y_test:
         y_test_.append(labels.index(i))
     Y_test = y_test_

     Y_test = tf.keras.utils.to_categorical(Y_test)
```

Figure 8: encoding

```
x_train, x_val, y_train, y_val = train_test_split(X_train, Y_train, test_size=0.1, random_state=28)
```

Figure 9: Splitting the data

```
[ ]  batch_size = 32
     nb_train_samples = x_train.shape[0]
     nb_validation_samples = x_val.shape[0]
     print(nb_train_samples)
     print(nb_validation_samples)
     training_generator = datagen.flow(x_train, y_train, batch_size=batch_size)
     validation_generator = datagen.flow(x_val, y_val, batch_size=batch_size)
```

Figure 10: Data Augmentation for Training Data

layer of the model. The input size of the image accepted is (150,150,3). The model is compiled using the SGD optimizer with the loss function as "categorical cross entropy" and the metrics used to validate its accuracy. The model is fitted with five epochs and 32 batches, Figure 11. It is evident that the model has seven convolutional layers with two max-pooling layers, and batch normalization has been applied at seven different stages.

The model is fitted using the "fit generator" method, which takes the training and validation data as input, the number of epochs, and the verbose value, as shown in Figure 12.

6 Efficient-B3

EfficientNet-B2 is an upgraded and improved version of the deep learning architecture's convolutional neural network (CNN). In addition, it has a global average pooling layer which is two-dimensional. The model was implemented using the imagenet weights, and softmax was used for the activation. In order to compile the loss, categorical cross-entropy was used with Adam as the optimizer, as shown in Figure 13.

```
[ ] from tensorflow.keras.applications import resnet50
    from tensorflow.keras.applications.imagenet_utils import preprocess_input
```

```
[ ] conv_model = resnet50.ResNet50(weights='imagenet',include_top=False,input_shape = (150,150,3))

    for layer in conv_model.layers[:-3]:
        layer.trainable=False  #The role of the embedding layer is to map a category into a dense space in a way that is useful for the task

    resnet_model = models.Sequential()
    resnet_model.add(layers.Conv2D(32,(3,3),activation = 'relu',name = 'Conv_',input_shape = (150,150,3)))
    resnet_model.add(layers.Conv2D(32,(3,3),activation = 'relu',name = 'Conv_2',padding = 'same'))
    resnet_model.add(layers.Conv2D(32,(3,3),activation = 'relu',name = 'Conv_3',padding = 'same'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.MaxPooling2D((2,2),name = 'max_1'))
    resnet_model.add(layers.Conv2D(64,(3,3),activation = 'relu',name = 'Conv_4',padding='same'))
    resnet_model.add(layers.Conv2D(64,(3,3),activation = 'relu',name = 'Conv_5',padding='same'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.MaxPooling2D((2,2),name = 'max_2'))

    resnet_model.add(layers.Conv2D(128,(3,3),activation='relu'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.MaxPooling2D((2,2)))
    resnet_model.add(layers.Conv2D(128,(3,3),activation='relu'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.Flatten())
    resnet_model.add(layers.Dense(512,activation = 'relu',name = 'L1',))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.Dense(256,activation = 'relu',name = 'L2'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.Dense(256,activation = 'relu',name = 'L3'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.Dense(128,activation = 'relu' ,name ='L4'))
    resnet_model.add(layers.BatchNormalization())
    resnet_model.add(layers.Dense(2,activation = 'softmax',name = 'output'))
```

Figure 11: Resnet Model Building

```
history1 = resnet_model.fit_generator(training_generator,
                            steps_per_epoch = nb_train_samples // batch_size,
                            epochs = 5,
                            verbose = 1,
                            validation_data = validation_generator,
                            validation_steps = nb_validation_samples // batch_size)
```

Figure 12: Resnet Model Fitting

7 CNN

CNN has become the most widely accepted deep-learning algorithm for generating high accuracy levels that machine-learning algorithms cannot achieve. A CNN implementation is built on hidden layers that mimic the working mechanism of a neuron. The python code for building the CNN model can be seen in Figure 14.

```
[ ] model2 = base_model.output
    model2 = GlobalAveragePooling2D()(model2)
    model2 = Dropout(0.4)(model2)
    model2 = Dense(2, activation='softmax')(model2)
    model2 = Model(inputs = base_model.input, outputs=model2)

    model2.compile(loss='categorical_crossentropy', optimizer=Adam(0.1), metrics=['accuracy'])
```

Figure 13: EfficientNet-B2 Model Building

```
    model = Sequential()
    model.add(Conv2D(15, (5,5),padding='valid',activation='relu',input_shape=(150,150,3)))
    model.add(MaxPooling2D(pool_size=(5,5)))
    model.add(Conv2D(12, (4,4),padding='valid',activation='relu'))
    model.add(MaxPooling2D(pool_size=(4,4)))
    model.add(Dropout(0.65))
    model.add(Conv2D(8, (1,1),padding='valid',activation='relu'))
    model.add(MaxPooling2D(pool_size=(1,1)))
    model.add(Dropout(0.55))
    model.add(Flatten())
    model.add(Dense(2,activation='softmax'))

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

Figure 14: CNN Model Building

8 Evaluation Metrics

The trained models are evaluated using the accuracy and the loss plot, as shown in Figure 15. The plots determine whether the model provides valuable insights if implemented. Sensitivity and Specificity values are used to determine if the model can be used in real-time based on the values obtained. The code snippet for this evaluation metrics can be seen in Figure 17. Another evaluation metric used was the Confusion matrix, and its code can be seen in Figure 16. It determines the number of cases the model could predict correctly and the number of cases predicted wrongly.

```
#accuracy and loss plot
plt.plot(history1.history['acc'])
plt.plot(history1.history['val_acc'])
plt.title('Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','val'], loc='upper left')
plt.show()

#loss plot
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','val'], loc='upper left')
plt.show()
```

Figure 15: Code for Accuracy and Loss Plot

```
pred = resnet_model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(Y_test,axis=1)

cmat = confusion_matrix(y_test_new,pred)
plt.figure(figsize=(6,6))
sns.heatmap(cmat, annot = True, cbar = False);
```

Figure 16: Confusion Matrix Code

9 Graphical User Interface

Visual Studio Code application is used to create a standalone application using the python packages. The project structure consists of the dataset used for model training and the CNN model, which outperformed the other transfer learning models. In addition, it

```
from sklearn.metrics import precision_recall_fscore_support
res = []
for l in range(2):
     prec,recall,_,_ = precision_recall_fscore_support(y_test_new==l,
                                                       pred==l,
                                                       pos_label=True,average=None)
     res.append([l,recall[0],recall[1]])

pd.DataFrame(res,columns = ['class','sensitivity','specificity'])
```

Figure 17: Sensitivity and Specificity

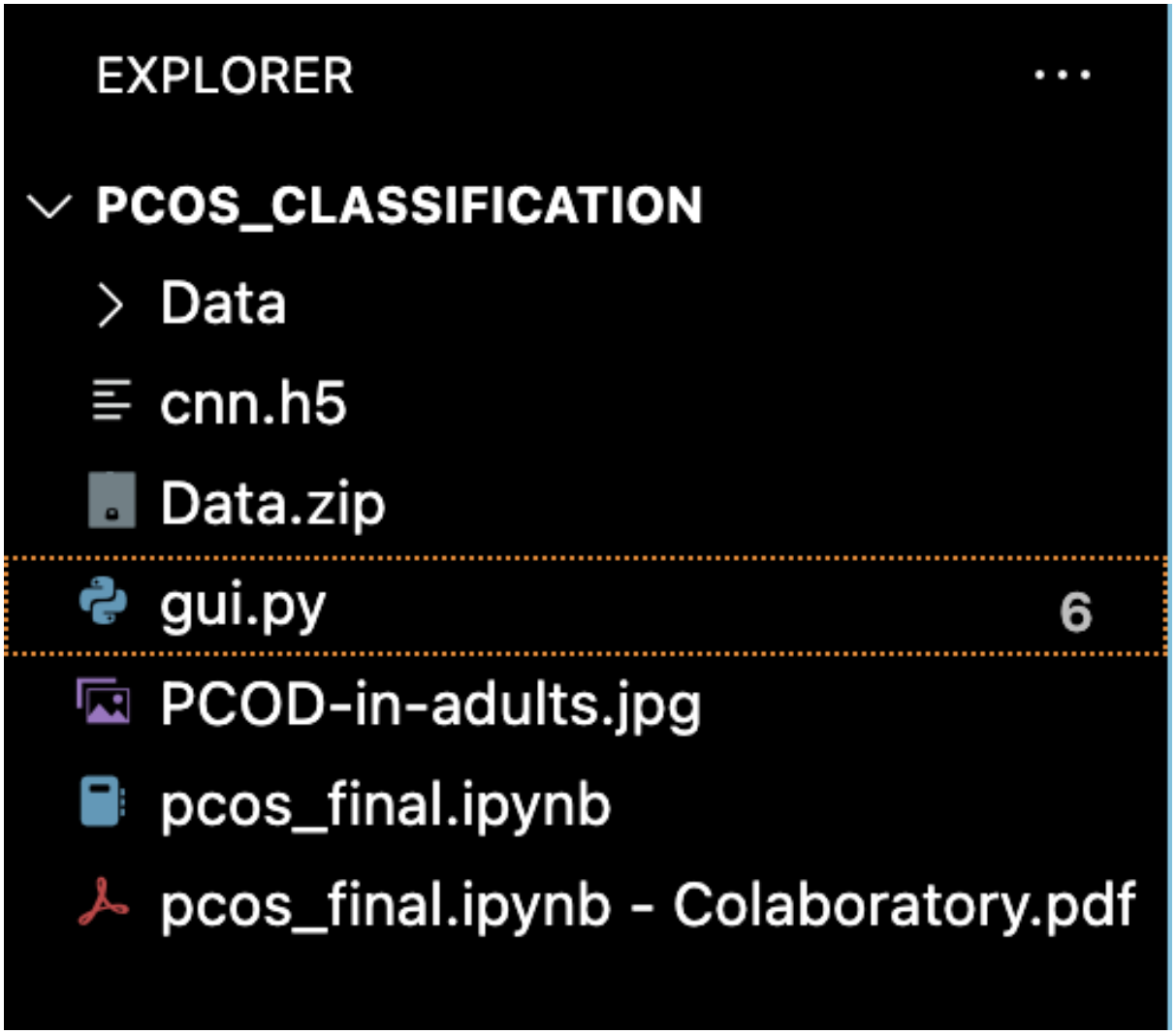

Figure 18: Project Structure

contains the "gui.py" file, which contains the python code for the graphical interface. The project structure can be seen in Figure 18.

The code for creating the dialog box using the python package can be seen in Figure 19. Initially, the frames were created, and the background image was added. The buttons to upload the image and the submit button to predict the output of the uploaded file is

```
frame2.pack_forget()
frame3.pack_forget()

e2 = Text(frame2, width=40,height=1)
e2.grid(row=1, column=2,padx=10)

e1 = Text(frame1,height=15, width=70)
e1.grid(row=1, column=2, padx=10,pady=10)


button5 = Button(frame3, text="Browse",command=clickbrowse,width=20,height=2)
button5.grid(row=12, column=1, pady=10,padx=10)

button5 = Button(frame3, text="Submit",command=click1,width=20,height=2)
button5.grid(row=13, column=1, pady=10,padx=10)


frame2.configure(background="silver")
frame2.pack(pady=10)

frame1.configure(background="red")
frame1.pack(pady=10)

frame3.configure(background="silver")
frame3.pack()
```

Figure 19: Code for Dialog Box

created. When the "Browse" button is clicked, the file manager opens, and the ultrasound image that needs to be tested is uploaded. After selecting the image, the "submit" button needs to be clicked, using the saved CNN model to predict whether the woman is infected. Different HTML and click event functionality are written to make the page responsive. The application can be started by clicking the run button in the VS code application.

References

Carneiro, T., Medeiros Da NóBrega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C. and Filho, P. P. R. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications, Vol. 6, pp. 61677–61685.

Ertam, F. and Aydın, G. (2017). Data classification with deep learning using tensorflow, *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 755–758.