GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

SKIN DISEASE DETECTION SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

HIMANSHU

(1613101291/16SCSE101298)

in partial fulfillment for the award of the degree of

Bachelor of Technology

IN

Computer Science and Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Ms. Kamakshi Gupta, Assistant Professor (SCSE)

APRIL / MAY- 2020

TABLE OF CONTENTS

ABSTRACT

The practical increase of interest in intelligent technologies has caused a rapid development of all activities in terms of sensors and automatic mechanisms for smart operations. The implementations concentrate on technologies which avoid unnecessary actions on user side while examining health conditions. One of important aspects is the constant inspection of the skin health due to possible diseases such as melanomas that can develop under excessive influence of the sunlight. Our proposed system will help in identifying skin diseases and also help us in providing proper precautions.

Skin diseases are among the most common health problems worldwide. In this article we proposed a method that uses computer vision based techniques to detect various kinds of dermatological skin diseases. We have used different types of image processing algorithms for feature extraction and feed forward artificial neural network for training and testing purpose. The system works on two phases- first pre-process the colour skin images to extract significant features and later identifies the diseases.

INTRODUCTION

Skin is the largest organ present in the human body. Skin disease doesn’t just affect the skin. It can have a huge impact on a person’s day-to-day life, crush self- confidence, restrict their movement, and lead to depression and even ruin relationships. So it is needed to take skin disease seriously. Skin diseases are very common among people of all age groups. In 2019 with prevalence rate of 10 percent, the population affected across India from skin disease was estimated at nearly 15.1 crore.

(i) Overall Description:

Skin disease diagnosis is very essential in earlier stage in order to prevent and control them. The proposed software takes input a skin image and based on CNN algorithm it classifies skin into (1) Healthy (2) Bacterial Disease And also provide Remedies based on the classified disease.

(ii) Purpose:

The skin infections impact the people directly. The skin disease become the important factor which causes significant reduction in the health and self-confidence. The detection and classification of disease is important task in skin disease detection, here we proposed a novel method of the using the CNN algorithm for the detection and classification of the disease.

(iii) Motivations and Scope:

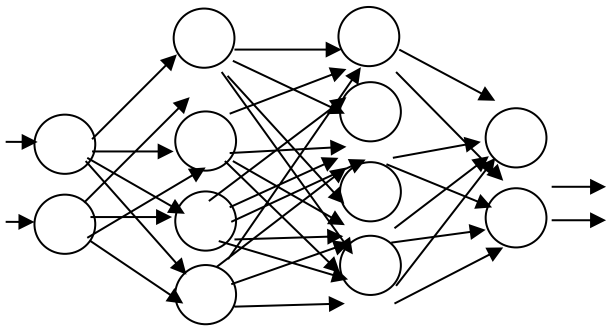
Skin diseases are most common form of infections occurring in people of all ages. As the costs of dermatologists to monitor every patient is very high, there is a need for a computerized system to evaluate patient’s risk of skin disease using images of their skin lesions. Skin is the largest organ present in the human body. Skin disease doesn’t just affect the skin. It can have a huge impact on a person’s day-to-day life, crush self- confidence, restrict their movement, and lead to depression.

EXISTING SYSTEM

1. In the paper "Survey of Texture Based Feature Extraction for Skin Disease Detection" by Seema Kolkur, D.R. Kalbande, an approach of texture based feature extraction for detection of skin diseases has been presented to resolve issues. In statistical texture analysis, texture features are computed from the statistical distribution of observed combinations of intensities at specified positions relative to each other in the image. According to the number of intensity points (pixels) in each combination, statistics are classified into first-order, second-order and higher-order statistics. GLCM method is a way of extracting second order statistical texture features. Third and higher order textures consider the relationships among three or more pixels. These are theoretically possible but not commonly implemented due to calculation time and interpretation difficulty. In this paper, work on texture based features derived from GLCM matrix used for the detection of skin diseases is discussed and consolidated.
2. In the paper "Detection of Malignant Skin Diseases Based on the Lesion Segmentation" by Jyothilakshmi K. K, Jeeva J. B, a novel technique to automatically detect the malignancy of skin diseases using conventional camera images is proposed. The procedure used would be of great advantage to the dermatologists as a pre-screening system for early diagnosis in situations where the dermoscopes are not accessible. This algorithm mainly aims at an early diagnosis of the malignant diseases since they can be cured if detected early. The proposed method works on color images by taking the HSV component and preprocessing was performed. A robust segmentation procedure is performed for the accurate detection of the lesion. For detection purpose, the morphological features like asymmetry, border irregularity, color variation and diameter are used. These extracted features help to identify the malignant lesions from the non-malignant

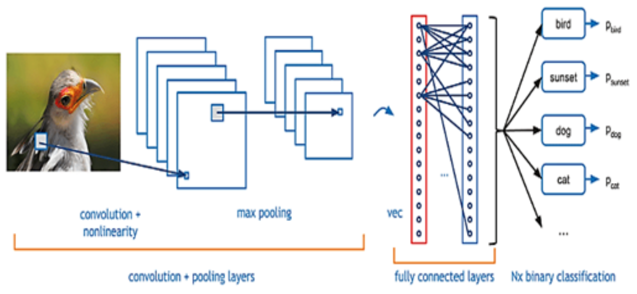
PROPOSED SYSTEM

Skin Disease Detection System is a computer based technology that use computer vision for taking input of various skin diseases like eczema, melanoma etc. Here we have used technologies like image processing for enhancing our input image by eliminating unwanted or noisy data. Our proposed system will work on the CNN based classification for classifying the input. This system contributes in describing and understanding a CNN (Convolutional Neural Network) classification based technique through which we can cure and detect skin diseases efficiently at lower cost. We are using a CNN based classification as it takes less number of data nodes and weights for transferring data.



This is how a simple artificial neural net-work looks.

This is how a CNN looks.



Implementation or architecture diagrams

Code for the user interface:

```
import tkinter as tk

from tkinter.filedialog import askopenfilename

import shutil

import os

import sys

from PIL import Image, ImageTk


window = tk.Tk()


window.title("Skin Disease Detection System")


window.geometry("500x510")

window.configure(background ="lightgreen")


title = tk.Label(text="Click below to choose picture for testing disease....", background =
"lightgreen", fg="Brown", font=("", 15))

title.grid()

def acne():

    window.destroy()

    window1 = tk.Tk()


    window1.title("Skin Disease Detection System")
    window1.geometry("500x510")
    window1.configure(background="lightgreen")


    def exit():
        window1.destroy()
    rem = "The remedies for acne are:\n\n "
    remedies = tk.Label(text=rem, background="lightgreen",
                        fg="Brown", font=("", 15))
    remedies.grid(column=0, row=7, padx=10, pady=10)
    rem1 = "Take vitamin A derivative. \n Apply Benzoyl Peroxide on the infected part
regularly."
    remedies1 = tk.Label(text=rem1, background="lightgreen",
                         fg="Black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)


    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)


    window1.mainloop()



def melanoma():
    window.destroy()
    window1 = tk.Tk()


    window1.title("Skin Disease Detection System")
    window1.geometry("650x510")
    window1.configure(background="lightgreen")

    def exit():
    window1.destroy()
    rem = "The remedies for melanoma are: "
    remedies = tk.Label(text=rem, background="lightgreen",
                        fg="Brown", font=("", 15))
    remedies.grid(column=0, row=7, padx=10, pady=10)
    rem1 = " Take anti virul drug. \n  Immunotherapy. \n  Consult to a dermatologist"
    remedies1 = tk.Label(text=rem1, background="lightgreen",
                         fg="Black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)

    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

    window1.mainloop()

def eczema():
    window.destroy()
    window1 = tk.Tk()

    window1.title("Skin Disease Detection System")

    window1.geometry("520x510")
    window1.configure(background="lightgreen")
    def exit():
        window1.destroy()
    rem = "The remedies for eczema are: "
    remedies = tk.Label(text=rem, background="lightgreen",
                   fg="Brown", font=("", 15))
    remedies.grid(column=0, row=7, padx=10, pady=10)

    rem1 = " Apply Coal tar extract for preventing irritation. \n  Consult to a 
dermatologist."
    remedies1 = tk.Label(text=rem1, background="lightgreen",
                    fg="Black", font=("", 12))
    remedies1.grid(column=0, row=8, padx=10, pady=10)

    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

    window1.mainloop()



def analysis():
    import cv2
    import numpy as np
    import os
    from random import shuffle
    from tqdm import
    verify_dir = 'testpicture'
```

```
IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'healthyvsunhealthy-{}-{}.model'.format(LR, '2conv-basic')

def process_verify_data():
    verifying_data = []
    for img in tqdm(os.listdir(verify_dir)):
        path = os.path.join(verify_dir, img)
        img_num = img.split('.')[0]
        img = cv2.imread(path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        verifying_data.append([np.array(img), img_num])
    np.save('verify_data.npy', verifying_data)
    return verifying_data

verify_data = process_verify_data()
#verify_data = np.load('verify_data.npy')

import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')
```

```
    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 128, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = fully_connected(convnet, 1024, activation='relu')
    convnet = dropout(convnet, 0.8)

    convnet = fully_connected(convnet, 4, activation='softmax')
    convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')

    model = tflearn.DNN(convnet, tensorboard_dir='log')

    if os.path.exists('{}.meta'.format(MODEL_NAME)):
        model.load(MODEL_NAME)
        print('model loaded!')
```

```
import matplotlib.pyplot as plt

fig = plt.figure()

for num, data in enumerate(verify_data):

    img_num = data[1]
    img_data = data[0]

    y = fig.add_subplot(3, 4, num + 1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE, IMG_SIZE, 3)
    # model_out = model.predict([data])[0]
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 0:
        str_label = 'healthy'
    elif np.argmax(model_out) == 1:
        str_label = 'acne'
    elif np.argmax(model_out) == 2:
        str_label = 'melanoma'
    elif np.argmax(model_out) == 3:
        str_label = 'eczema'

    if str_label =='healthy':
        status ="HEALTHY"
    else:
        status = "UNHEALTHY"

    message = tk.Label(text='Status: '+status, background="lightgreen",
                       fg="Brown", font=("", 15))
    message.grid(column=0, row=3, padx=10, pady=10)
    if str_label == 'acne':
        diseasename = "Acne "
        disease = tk.Label(text='Disease Name: ' + diseasename,
background="lightgreen",
                           fg="Black", font=("", 15))
        disease.grid(column=0, row=4, padx=10, pady=10)
        r = tk.Label(text='Click below for remedies...', background="lightgreen",
fg="Brown", font=("", 15))
        r.grid(column=0, row=5, padx=10, pady=10)
        button3 = tk.Button(text="Remedies", command=bact)
        button3.grid(column=0, row=6, padx=10, pady=10)
    elif str_label == 'melanoma':
        diseasename = "Melanoma"
        disease = tk.Label(text='Disease Name: ' + diseasename,
background="lightgreen",
                           fg="Black", font=("", 15))
        disease.grid(column=0, row=4, padx=10, pady=10)
        r = tk.Label(text='Click below for remedies...', background="lightgreen",
fg="Brown", font=("", 15))
        r.grid(column=0, row=5, padx=10, pady=10)
        button3 = tk.Button(text="Remedies", command=vir)
        button3.grid(column=0, row=6, padx=10, pady=10)
    elif str_label == 'eczema':

        diseasename = "Eczema "

        disease = tk.Label(text='Disease Name: ' + diseasename,
background="lightgreen",

                           fg="Black", font=("", 15))

        disease.grid(column=0, row=4, padx=10, pady=10)

        r = tk.Label(text='Click below for remedies...', background="lightgreen",
fg="Brown", font=("", 15))

        r.grid(column=0, row=5, padx=10, pady=10)

        button3 = tk.Button(text="Remedies", command=latebl)

        button3.grid(column=0, row=6, padx=10, pady=10)

    else:

        r = tk.Label(text='skin is healthy', background="lightgreen", fg="Black",

                     font=("", 15))

        r.grid(column=0, row=4, padx=10, pady=10)

        button = tk.Button(text="Exit", command=exit)

        button.grid(column=0, row=9, padx=20, pady=20)


def openphoto():

    dirPath = "testpicture"

    fileList = os.listdir(dirPath)

    for fileName in fileList:

    os.remove(dirPath + "/" + fileName)

    fileName = askopenfilename(initialdir='C:/Users/Downloads/images', title='Select
image for analysis ',

                       filetypes=[('image files', '.jpg')])

    dst = "C:/Users/Desktop/plant_project/testpicture"

    shutil.copy(fileName, dst)
    load = Image.open(fileName)
    render = ImageTk.PhotoImage(load)
    img = tk.Label(image=render, height="250", width="500")
    img.image = render
    img.place(x=0, y=0)
    img.grid(column=0, row=1, padx=10, pady = 10)
    title.destroy()
    button1.destroy()
    button2 = tk.Button(text="Analyse Image", command=analysis)
    button2.grid(column=0, row=2, padx=10, pady = 10)
button1 = tk.Button(text="Get Photo", command = openphoto)
button1.grid(column=0, row=1, padx=10, pady = 10)



window.mainloop()
```

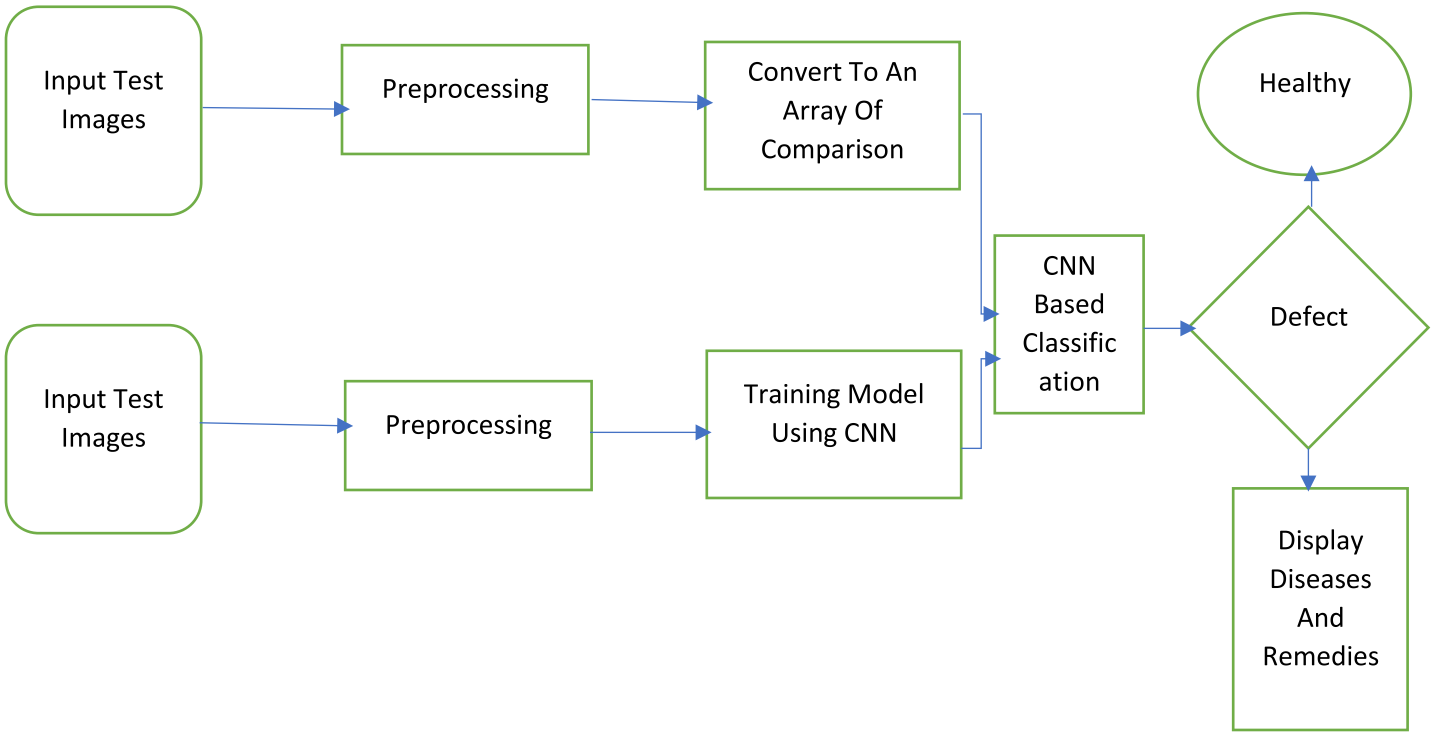
Code for the CNN classification model:

```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
TRAIN_DIR = 'train/train'
TEST_DIR = 'test/test'
IMG_SIZE = 50
LR = 1e-3
```

```
MODEL_NAME = 'healthyvsunhealthy-{}-{}.model'.format(LR, '2conv-basic')

def label_img(img):
    word_label = img[0]

    if word_label == 'h': return [1,0,0,0]

    elif word_label == 'b': return [0,1,0,0]
    elif word_label == 'v': return [0,0,1,0]
    elif word_label == 'l': return [0,0,0,1]

def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data

def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data

train_data = create_train_data()
# If you have already created the dataset:
#train_data = np.load('train_data.npy')



import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)


convnet = conv_2d(convnet, 128, 3, activation='relu')

convnet = max_pool_2d(convnet, 3)


convnet = conv_2d(convnet, 32, 3, activation='relu')

convnet = max_pool_2d(convnet, 3)


convnet = conv_2d(convnet, 64, 3, activation='relu')

convnet = max_pool_2d(convnet, 3)


convnet = fully_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)


convnet = fully_connected(convnet, 4, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')


model = tflearn.DNN(convnet, tensorboard_dir='log')


if os.path.exists('{}.meta'.format(MODEL_NAME)):

    model.load(MODEL_NAME)

    print('model loaded!')


train = train_data[:-500]

test = train_data[-500:]
```

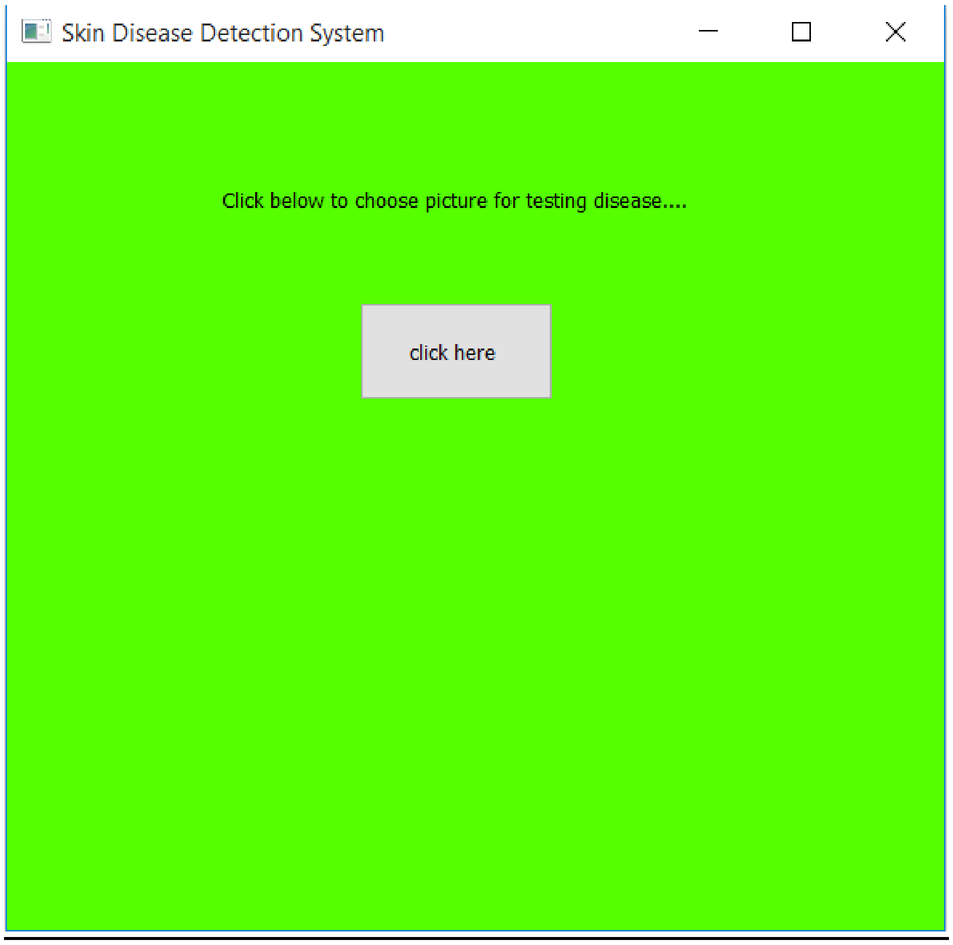
```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=8, validation_set=({'input': test_x},
{'targets': test_y}),
    snapshot_step=40, show_metric=True, run_id=MODEL_NAME)

model.save(MODEL_NAME)
```

Below is the diagrammatical representation of our proposed system:

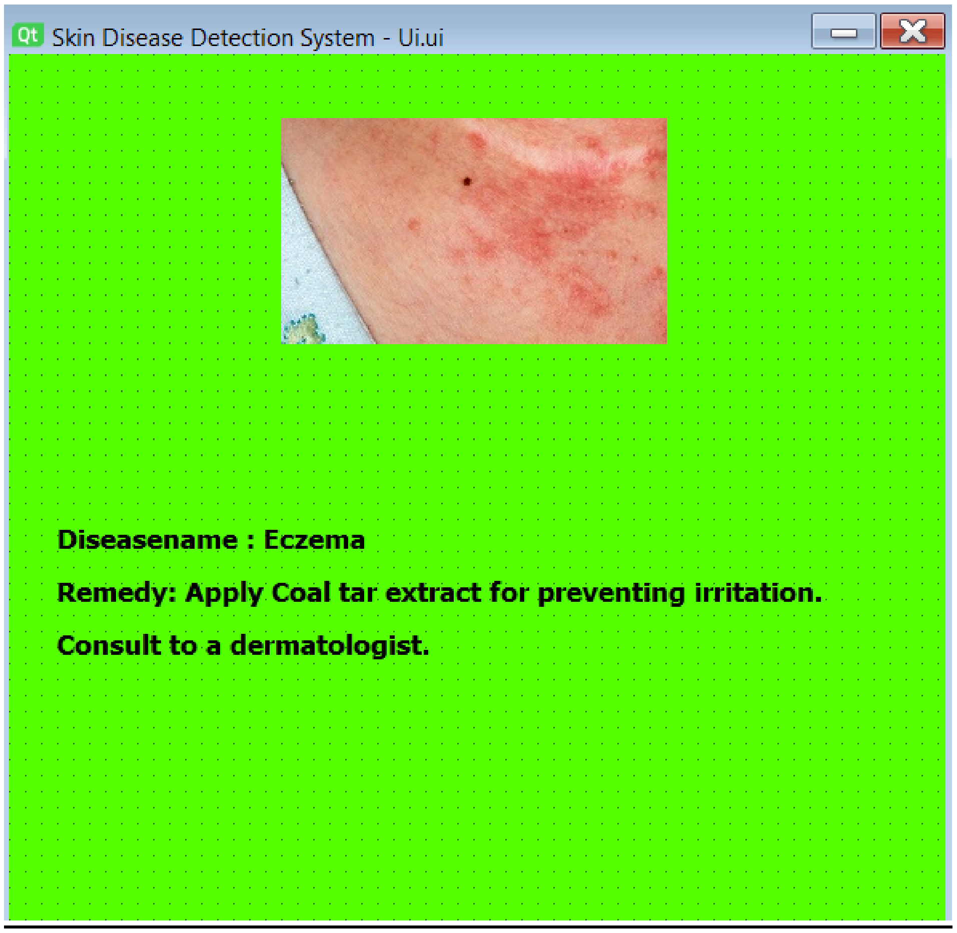


RESULT

User Interface:


Skin Disease Detection System
Click below to choose picture for testing disease....
click here

Result after the classification:



Diseasename : Eczema

Remedy: Apply Coal tar extract for preventing irritation.

Consult to a dermatologist.

Conclusion:

Accurate detection of disease is a very important part of the medication and we have taken this aspect very seriously that is why we have used CNN classification model for the skin disease detection. Though the system depends on the quality of the input image but we have tried to make the result more and more accurate. Image processing has also played a vital role in the removal of irrelevant things from input image which will help the CNN model to extract the feature from the images easily that will lead to prepare training data set quickly.

References:

- Digital Image Processing 4th Edition Pearson Book By Gonzalez (2018).
- https://www.google.com
- https://www.geeksforgeeks.org/digital-image-processing/
- https://www.elprocus.com
- https://www.analyticsvidhya.com/blog/2014/12/image-processingpython-basics/