AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA USING IBM WATSON

INDUSTRIAL ORIENTED MINI PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,

HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

KANKANALA SRI DIVYA	**20UK5A0512**
MOHAMMAD HANIF	**20UK5A0516**
JALIGAMA SAIKUMAR	**19UK1A05P4**
KURAPATI UMAMAHESHWAR	**18UK1A05B5**

Under the esteemed guidance of

Mr. N.RAVI

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
(Affiliated to JNTUH, Hyderabad)
Bollikunta, Warangal – 506005

2019 – 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
BOLLIKUNTA, WARANGAL – 506005
2019 – 2022

CERTIFICATE OF COMPLETION

INDUSTRIAL ORIENTED MINI PROJECT

This is to certify that the Industrial Oriented Mini Project entitled **“AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA USING IBM WATSON”** is being submitted by **K.SRIDIVYA (H.NO:20UK5A0512), MD.HANIF (H.NO:20UK5A0516), J.SAIKUMAR (H.NO:19UK1A05P4), K.UMAMAHESHWAR (H.NO:18UK1A05B5),** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year **2022-23**, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
Mr. N.Ravi	**Dr. R. Naveen Kumar**
(Assistant Professor)	(Professor)

External

TABLE OF CONTENTS

1. INTRODUCTION

Overview:

Skin is the largest and most sensitive part of the human body which protects our inner vital parts and organs from the outside environment, hence avoiding contact with bacteria and viruses. Skin also helps in body temperature regulation. The skin consists of cells, pigmentation, blood vessels, and other components. It is comprised of 3 main layers, namely, the epidermis, the dermis, and the hypodermis.

Skin diseases occur commonly among humans. They are usually caused by factors like different organism’s cells, a different diet, and internal and external factors, such as the hierarchical genetic group of cells, hormones, and immune system of conditions. These factors may act together or in a sequence of skin disease. There are chronic and incurable diseases, like eczema and psoriasis, and malignant diseases like malignant melanoma.

A patient can recover from skin diseases if it is detected and treated in the early stages and this can achieve cure ratios of over 95%. Hence, it is important to identify these diseases at their initial stage to control them from spreading.

Purpose:

Skin Disease Detection at edge predicts the disease of skin from the image of that infected part in less than one second. This system simply takes a skin disease image and gives the disease name with accuracy and time taken for prediction.

Skin diseases are primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis. Such a system is often prone to errors.

The main purpose of this project is to improve the accuracy of diagnostic systems by using Image Processing and classification techniques.

2.LITERATURE SURVEY

Existing Problem:

Skin diseases are primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis. To ascertain what type of skin disease a person has, they must visit a dermatologist. The dermatologist then performs visual analysis using various tests, some of which include:

1. Patch test: Known allergens are applied to a patch of skin and left for some time. The skin is then tested for a reaction.
2. Biopsy: Skin is removed using a scalpel, a blade or a biopsy tool and taken to a laboratory for analysis.

Culture: Skin of affected area or hair or nails are cultured to determine which microorganism is causing the infection.

Such a system is often time consuming and requires a number of expert professionals. Since there are people involved in this process, it is prone to human errors. This system is also quite expensive as laboratories charge a lot of fees for the tests.

Proposed Solution:

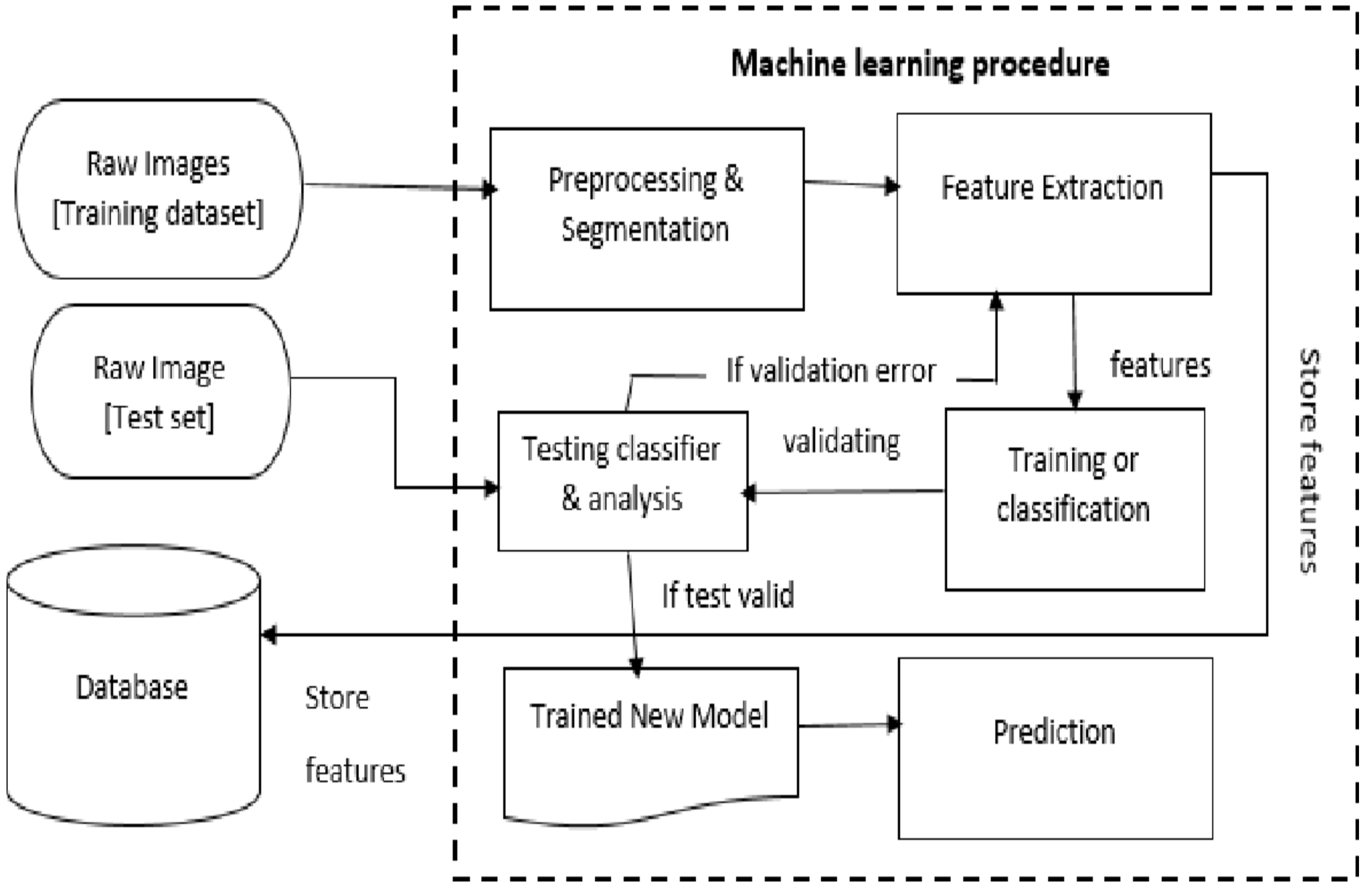
A skin disease detection and classification system is a system used for detecting whether a disease is present or not, and then classifying the type of disease, if present.

The classification is based on decisions taken using the features extracted through the feature extraction methods. In order to identify whether a disease is present or not, the system must be trained to recognize normal conditions of system activity.

There are two main phases for this purpose: training phase (building a profile using data about a particular disease) and testing phase (comparing the current image data with the trained image data)

3. THEORETICAL ANALYSIS

Block Diagram:



Hardware / Software designing:

Hardware Requirements:

- Operating system- Windows 7,8,10.
- Processor- dual-core 2.4 GHz (i5 or i7 series Intel processor or equivalent AMD).
- RAM-8GB.

Software Requirements:

- Python IDE (IDLE / Spyder / PyCharm) (Python 3.7)
- Microsoft's Visual Object Tagging Tool (VoTT)
- Python Packages need to be installed.

4.EXPERIMENTAL INVESTIGATIONS

Skin diseases are the fourth leading cause of non-fatal disease burden affecting almost 900 million people worldwide which can cause serious psychological problem including depression, frustration and even suicidal ideation. The shortage of sophisticated diagnostic devices and dermatologists and even general practitioners in developing countries make things worse and hurdle the service delivery. Moreover, the common diagnostic techniques including visual inspection, laboratory test, imaging and biopsy tests are tedious and require
experience and excellent visual perception. Computer-aided diagnosis systems have a potential to revolutionize the current disease diagnosis techniques enabling optimal treatment planning.

The aim of this study was to design and develop a smartphone based automatic skin disease diagnosis method using skin images and patient information, anatomical site of the disease and symptom list.

Artificial Intelligence is taking over automation in all fields of application even within the healthcare field. In the past years these diseases have been a matter of concern due to the sudden arrival and the complexities which have increased life risks. These Skin abnormalities are very infectious and they require to be treated at earlier stages to avoid it from spreading. The majority of diseases are caused by unprotected exposure to excessive Ultraviolet Radiation(UR). Among all, benign type is considered to be less dangerous than malignant melanoma and can be cured with proper treatment, whereas the deadliest form of skin lesion is malignant Melanoma. The investigation results indicate that the back and lower extremity, trunk and upper extremity are heavily compromised regions of skin cancer. There are large instances of patients with age ranging from 30 to 60. Also, MelanocyticNevi, Carcinoma and Dermatofibroma are not prevalent below the age of 20 years.

5.FLOWCHART DIAGRAM

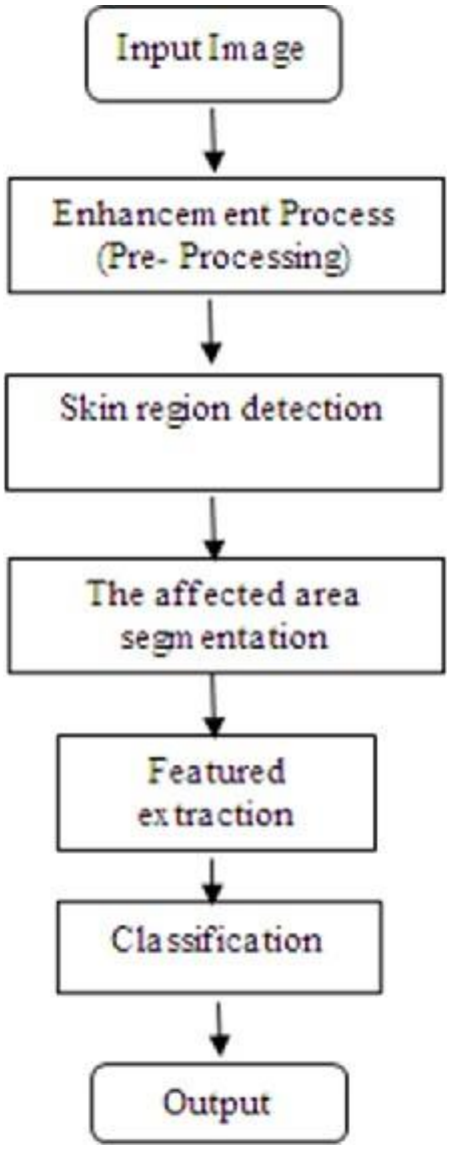


Fig: Flowchart Diagram

6.RESULTS

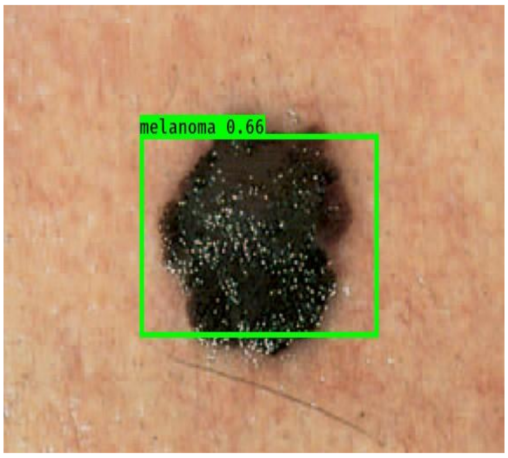


Fig: Melanoma

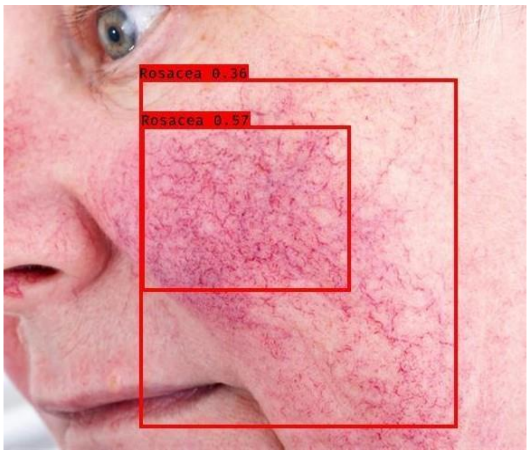


Fig: Rosacea

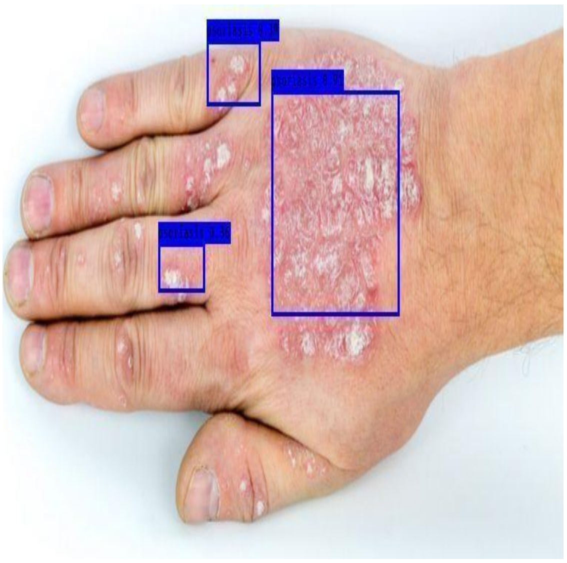


Fig: Psoriasis

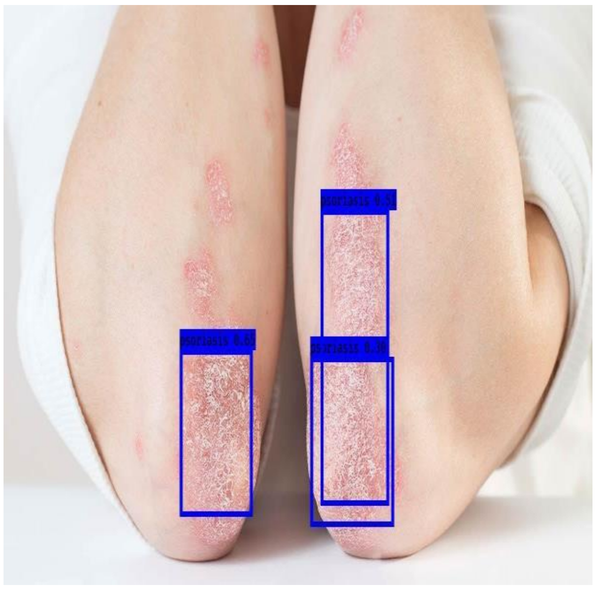
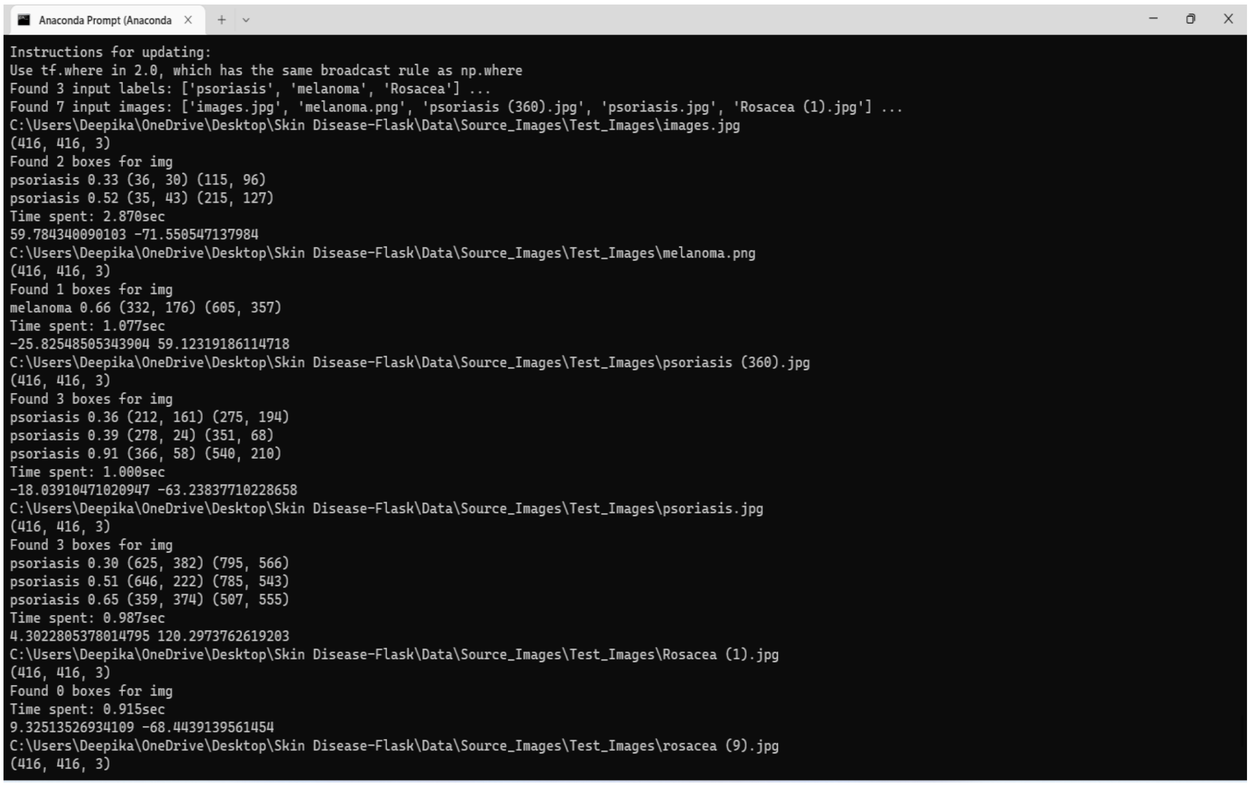
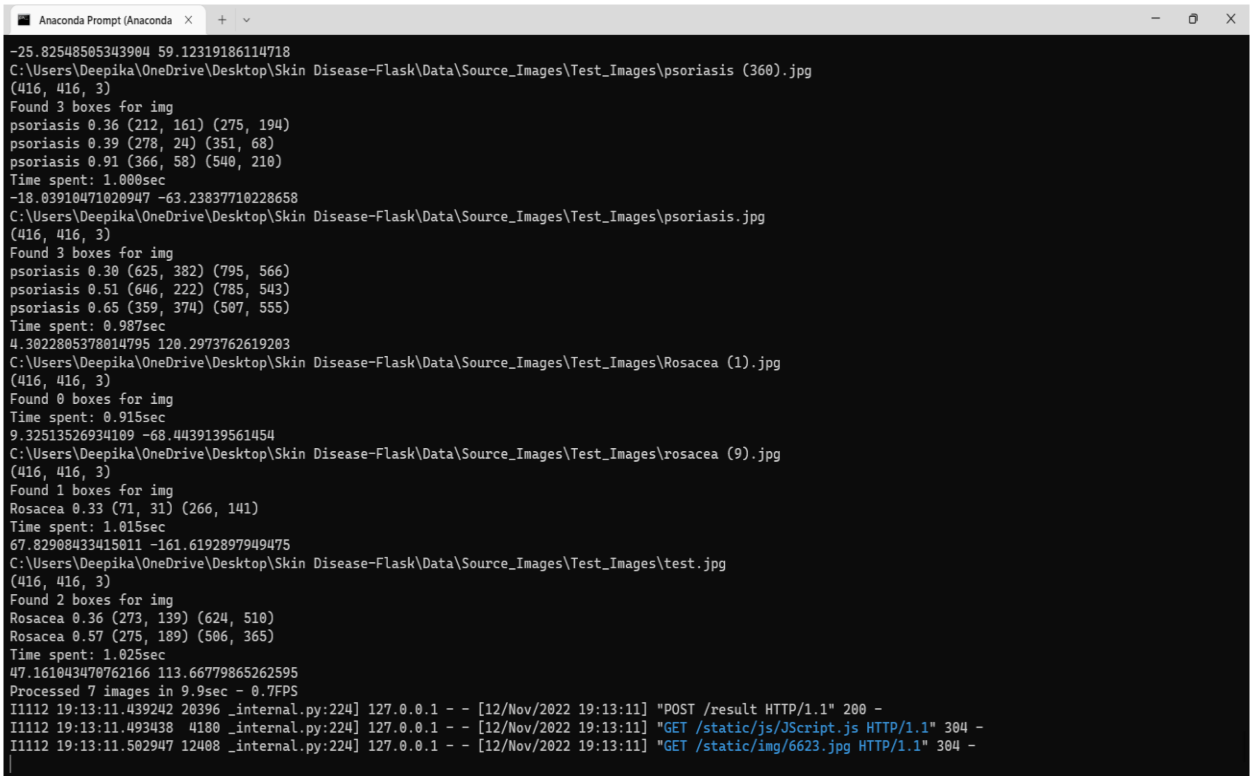


Fig: Psoriasis



```
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Found 3 input labels: ['psoriasis', 'melanoma', 'Rosacea'] ...
Found 7 input images: ['images.jpg', 'melanoma.png', 'psoriasis (360).jpg', 'psoriasis.jpg', 'Rosacea (1).jpg'] ...
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\images.jpg
(416, 416, 3)
Found 2 boxes for img
psoriasis 0.33 (36, 30) (115, 96)
psoriasis 0.52 (35, 43) (215, 127)
Time spent: 2.870sec
59.784340090103 -71.550547137984
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\melanoma.png
(416, 416, 3)
Found 1 boxes for img
melanoma 0.66 (332, 176) (605, 357)
Time spent: 1.077sec
-25.82548505343904 59.12319186114718
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\psoriasis (360).jpg
(416, 416, 3)
Found 3 boxes for img
psoriasis 0.36 (212, 161) (275, 194)
psoriasis 0.39 (278, 24) (351, 68)
psoriasis 0.91 (366, 58) (540, 210)
Time spent: 1.000sec
-18.03910471020947 -63.23837710228658
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\psoriasis.jpg
(416, 416, 3)
Found 3 boxes for img
psoriasis 0.30 (625, 382) (795, 566)
psoriasis 0.51 (646, 222) (785, 543)
psoriasis 0.65 (359, 374) (507, 555)
Time spent: 0.987sec
4.3022805378014795 120.2973762619203
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\Rosacea (1).jpg
(416, 416, 3)
Found 0 boxes for img
Time spent: 0.915sec
9.32513526934109 -68.4439139561454
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\rosacea (9).jpg
(416, 416, 3)
```

Fig: Prediction results of psoriasis and melanoma disease.



```
-25.82548505343904 59.12319186114718
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\psoriasis (360).jpg
(416, 416, 3)
Found 3 boxes for img
psoriasis 0.36 (212, 161) (275, 194)
psoriasis 0.39 (278, 24) (351, 68)
psoriasis 0.91 (366, 58) (540, 210)
Time spent: 1.000sec
-18.03910471020947 -63.23837710228658
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\psoriasis.jpg
(416, 416, 3)
Found 3 boxes for img
psoriasis 0.30 (625, 382) (795, 566)
psoriasis 0.51 (646, 222) (785, 543)
psoriasis 0.65 (359, 374) (507, 555)
Time spent: 0.987sec
4.3022805378014795 120.2973762619203
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\Rosacea (1).jpg
(416, 416, 3)
Found 0 boxes for img
Time spent: 0.915sec
9.32513526934109 -68.4439139561454
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\rosacea (9).jpg
(416, 416, 3)
Found 1 boxes for img
Rosacea 0.33 (71, 31) (266, 141)
Time spent: 1.015sec
67.82908433415011 -161.6192897949475
C:\Users\Deepika\OneDrive\Desktop\Skin Disease-Flask\Data\Source_Images\Test_Images\test.jpg
(416, 416, 3)
Found 2 boxes for img
Rosacea 0.36 (273, 139) (624, 510)
Rosacea 0.57 (275, 189) (506, 365)
Time spent: 1.025sec
47.161043470762166 113.66779865262595
Processed 7 images in 9.9sec - 0.7FPS
I1112 19:13:11.439242 20396 _internal.py:224] 127.0.0.1 - - [12/Nov/2022 19:13:11] "POST /result HTTP/1.1" 200 -
I1112 19:13:11.493438  4180 _internal.py:224] 127.0.0.1 - - [12/Nov/2022 19:13:11] "GET /static/js/JScript.js HTTP/1.1" 304 -
I1112 19:13:11.502947 12408 _internal.py:224] 127.0.0.1 - - [12/Nov/2022 19:13:11] "GET /static/img/6623.jpg HTTP/1.1" 304 -
```

Fig: Prediction results of Rosacea & Psoriasis disease.

7. ADVANTAGES AND DISADVANTAGES

Advantages:

- It Provides a free basic dermoscopic tutorial for the general public.
- This system can be used by dermatologists to give a better diagnosis and treatment to the patients.
- The system can be used to diagnose skin diseases at a lower cost.
- This system can be used to detect and classify diseases as well as their severity.

Disadvantages:

- Internet connectivity is the major issue.
- In case of low Quality images it may show inaccurate information.
- Everyone is not aware of this application

8. APPLICATIONS

In dermatology, a study published in 2017 found 526 apps corresponding to an 80.8 percent growth in smartphone-based platforms since 2014. The boom is understandable: as you can easily detect if you have a skin problem, and smartphones coupled with super-fast internet connection make it easy to check images against databases, send pictures or footage anywhere, self-surveillance solutions, disease guides, educational apps as well as telehealth platforms appeared naturally in dermatology. Here’s our take on the most useful and efficient dermatology apps for the health of your skin.

The options of teledermatology services, as well as self-care platforms, are soaring. FirstDerm, Spruce, Direct Dermatology, SkinMDnow, Zwivel, iDoc24, SkinVision all work based on the same principle: they promise patients the option to self-check their symptoms and then to connect to a dermatologist online for consultation within a very short period of time. Usually, people can load up their photos to a certain platform, and smart algorithms and/or dermatologists give advice based on them.

Few Applications are :-

➢ SkinVision

➢ CureSkin

➢ iDoc24

➢ FirstDerm

9.CONCLUSION

The system proposed is a Skin Disease Detection System. This system uses test images to detect if it is healthy or not; if not, then classified as Melanoma, Psoriasis or Rosacea.

This system can be used by dermatologists to give a better diagnosis and treatment to the patients. The system can be used to diagnose skin diseases at a lower cost.

In future, this system can be improved to detect and classify more diseases as well as their severity.

10.FUTURE SCOPE

In future, this machine learning model may bind with various websites which can provide real time data for skin disease prediction. Also, we may add large historical data on skin disease which can help to improve the accuracy of the machine learning model. We can build an android app as a user interface for interacting with the user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates, and train it on clusters of data rather than the whole dataset.

11.BIBLIOGRAPHY

1. Doi k. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. Comput. Med. Imaging graph. 2007;31:198–211. Doi: 10.1016/j.Compmedimag.2007.02.002. [pmc free article] [pubmed] [crossref] [google scholar]
2. Yoshida h, dachman ah. Computer-aided diagnosis for ct colonography. Semin. Ultrasound ct mri. 2004;25:419–431. Doi: 10.1053/j.Sult.2004.07.002. [pubmed] [crossref] [google scholar]
3. Trabelsi, o., tlig, l., sayadi, m. & fnaiech, f., skin disease analysis and tracking based on image segmentation. 2013 international conference on electrical engineering and software applications, hammamet, 1–7. 10.1109/iceesa.2013.6578486 (2013).
4. Rajab mi, woolfson ms, morgan sp. Application of region-based segmentation and neural network edge detection to skin lesions. Comput. Med. Imaging graph. 2004;28:61–68. Doi: 10.1016/s0895-6111(03)00054- [pubmed] [crossref] [google scholar]
5. Keke, s., peng, z. & guohui, l., study on skin color image segmentation used by fuzzy-c- means arithmetic. In 2010 seventh international conference on fuzzy systems and knowledge discovery, yantai, 612–615.10.1109/fskd.2010.5569451 (2010).
6. Hongmao s. Quantitative structure-activity relationships: promise, validations, and pitfalls in a practical guide to rational drug design. Sawston: woodhead publishing; 2016. Pp. 163–192. [google scholar]

7.Lu, j., manton, j. H., kazmierczak e. & sinclair, r., erythema detection in digital skin images. In 2010 ieee international conference on image processing, hong kong, 2545–2548. 10.1109/icip.2010.5653524 (2010).

8.Sumithra r, suhil m, guru ds. Segmentation and classification of skin lesions for disease diagnosis. Proced. Comput. Sci. 2015;45:76–85. Doi: 10.1016/j.Procs.2015.03.090. [crossref] [google scholar]

9.Maglogiannis i, zafiropoulos e, kyranoudis c. Intelligent segmentation and classification of pigmented skin lesions in dermatological images in advances in artificial intelligence. Setn 2006. In: antoniou g, potamias g, spyropoulos c, plexousakis d, editors. Lecture notes incomputer science. Berlin: springer; 2006. Pp. 214–223. [google scholar]

10. Albawi, s., mohammed, t. A. & al-zawi, s., understanding of a convolutional neural network. In 2017 international conference on engineering and technology (icet), antalya, 1–6. 10.1109/icengtechnol.2017.8308186 (2017).

APPENDIX

A. SOURCE CODE

App.py

```
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
import sys
from flask import Flask, request, render_template, redirect, url_for
import argparse
from tensorflow import keras
from PIL import Image
from timeit import default_timer as timer
import test
import pandas as pd
import numpy as np
import random

def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path =r'C:\Users\HP\Desktop\Skin Disease-Flask\2_Training\src'
print(src_path)
utils_path = r'C:\Users\HP\Desktop\Skin Disease-Flask\Utils'
print(utils_path)

sys.path.append(src_path)
sys.path.append(utils_path)
```

```
import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
import test
import utils
import pandas as pd
import numpy as np
from Get_File_Paths import GetFileList
import random

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

# Set up folder names for default values
data_folder = os.path.join(get_parent_dir(n=1), "Skin Disease-Flask", "Data")

image_folder = os.path.join(data_folder, "Source_Images")

image_test_folder = os.path.join(image_folder, "Test_Images")

detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")

model_folder = os.path.join(data_folder, "Model_Weights")

model_weights = os.path.join(model_folder, "trained_weights_final.h5")
model_classes = os.path.join(model_folder, "data_classes.txt")

anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")

FLAGS = None


from cloudant.client import Cloudant

# Authenticate using an IAM API key
client = Cloudant.iam('2eb40045-a8d6-450d-9d24-52cc7cbb2810-
bluemix','Ud0wunTPOI_8h5ZtEqi1IXk1gIKeYLmpUsCn0EeO8T4z', connect=True)


# Create a database using an initialized client
my_database = client.create_database('my_database')


app=Flask(__name__)

#default home page or route
@app.route('/')
def index():
    return render_template('index.html')



@app.route('/index.html')
def home():
    return render_template("index.html")


#registration page
@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    '_id': x[1], # Setting _id is optional
    'name': x[0],
    'psw':x[2]
    }
    print(data)

    query = {'_id': {'$eq': data['_id']}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))

    if(len(docs.all())==0):
        url = my_database.create_document(data)
        #response = requests.get(url)
        return render_template('register.html', pred="Registration Successful, please login
using your details")
    else:
        return render_template('register.html', pred="You are already a member, please
login using your details")

#login page
@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user = request.form['_id']
    passw = request.form['psw']
    print(user,passw)

    query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))


    if(len(docs.all())==0):
        return render_template('login.html', pred="The username is not found.")
    else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')


@app.route('/logout')
def logout():
    return render_template('logout.html')

@app.route('/prediction')
def prediction():
    return render_template('prediction.html')


@app.route('/result',methods=["GET","POST"])
def res():
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--input_path",
        type=str,
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Default
is "
        + image_test_folder,
    )

    parser.add_argument(
        "--output",
        type=str,
        default=detection_results_folder,
        help="Output path for detection results. Default is "
        + detection_results_folder,
    )

    parser.add_argument(
        "--no_save_img",
        default=False,
        action="store_true",
        help="Only save bounding box coordinates but do not save output images with
annotated boxes. Default is False.",
    )

    parser.add_argument(
        "--file_types",
        "--names-list",
        nargs="*",
        default=[],
        help="Specify list of file types to include. Default is --file_types .jpg .jpeg
.png .mp4",
    )

    parser.add_argument(
        "--yolo_model",
        type=str,
        dest="model_path",
        default=model_weights,
        help="Path to pre-trained weight files. Default is " + model_weights,
    )

    parser.add_argument(
        "--anchors",
        type=str,
        dest="anchors_path",
        default=anchors_path,
        help="Path to YOLO anchors. Default is " + anchors_path,
    )

    parser.add_argument(
        "--classes",
        type=str,
        dest="classes_path",
        default=model_classes,
        help="Path to YOLO class specifications. Default is " + model_classes,
    )

    parser.add_argument(
        "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"
    )

    parser.add_argument(
        "--confidence",
        type=float,
        dest="score",
        default=0.25,
        help="Threshold for YOLO object confidence score to show predictions. Default is
0.25.",
    )

    parser.add_argument(
        "--box_file",
        type=str,
        dest="box",
        default=detection_results_file,
        help="File to save bounding box results to. Default is "
        + detection_results_file,
    )

    parser.add_argument(
        "--postfix",
        type=str,
        dest="postfix",
        default="_disease",
        help='Specify the postfix for images with bounding boxes. Default is "_disease"',
    )

    FLAGS = parser.parse_args()

    save_img = not FLAGS.no_save_img

    file_types = FLAGS.file_types
    #print(input_path)

    if file_types:
        input_paths = GetFileList(FLAGS.input_path, endings=file_types)
        print(input_paths)
    else:
        input_paths = GetFileList(FLAGS.input_path)
        print(input_paths)

    # Split images and videos
    img_endings = (".jpg", ".jpeg", ".png")
    vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")

    input_image_paths = []
    input_video_paths = []
    for item in input_paths:
        if item.endswith(img_endings):
            input_image_paths.append(item)
        elif item.endswith(vid_endings):
            input_video_paths.append(item)

    output_path = FLAGS.output
```

```
if not os.path.exists(output_path):
    os.makedirs(output_path)

# define YOLO detector
yolo = YOLO(
    **{
        "model_path": FLAGS.model_path,
        "anchors_path": FLAGS.anchors_path,
        "classes_path": FLAGS.classes_path,
        "score": FLAGS.score,
        "gpu_num": FLAGS.gpu_num,
        "model_image_size": (416, 416),
    }
)

# Make a dataframe for the prediction outputs
out_df = pd.DataFrame(
    columns=[
        "image",
        "image_path",
        "xmin",
        "ymin",
        "xmax",
        "ymax",
        "label",
        "confidence",
        "x_size",
        "y_size",
    ]
)

# labels to draw on images
class_file = open(FLAGS.classes_path, "r")
input_labels = [line.rstrip("\n") for line in class_file.readlines()]
print("Found {} input labels: {} ...".format(len(input_labels), input_labels))

if input_image_paths:
    print(
        "Found {} input images: {} ...".format(
            len(input_image_paths),
            [os.path.basename(f) for f in input_image_paths[:5]],
        )
    )
    start = timer()
    text_out = ""

    # This is for images
    for i, img_path in enumerate(input_image_paths):
        print(img_path)
        prediction, image,lat,lon= detect_object(
            yolo,
            img_path,
            save_img=save_img,
            save_img_path=FLAGS.output,
            postfix=FLAGS.postfix,
        )
        print(lat,lon)
        y_size, x_size, _ = np.array(image).shape
        for single_prediction in prediction:
            out_df = out_df.append(
                pd.DataFrame(
                    [
                        [
                                        os.path.basename(img_path.rstrip("\n")),
                                        img_path.rstrip("\n"),
                                    ]
                                    + single_prediction
                                    + [x_size, y_size]
                                ],
                                columns=[
                                    "image",
                                    "image_path",
                                    "xmin",
                                    "ymin",
                                    "xmax",
                                    "ymax",
                                    "label",
                                    "confidence",
                                    "x_size",
                                    "y_size",
                                ],
                            )
                        )
            end = timer()
            print(
                "Processed {} images in {:.1f}sec - {:.1f}FPS".format(
                    len(input_image_paths),
                    end - start,
                    len(input_image_paths) / (end - start),
                )
            )
            out_df.to_csv(FLAGS.box, index=False)

        # This is for videos
        if input_video_paths:
            print(
                "Found {} input videos: {} ...".format(
                    len(input_video_paths),
                    [os.path.basename(f) for f in input_video_paths[:5]],
                )
            )
            start = timer()
            for i, vid_path in enumerate(input_video_paths):
                output_path = os.path.join(
                    FLAGS.output,
                    os.path.basename(vid_path).replace(".", FLAGS.postfix + "."),
                )
                detect_video(yolo, vid_path, output_path=output_path)

            end = timer()
            print(
                "Processed {} videos in {:.1f}sec".format(
                    len(input_video_paths), end - start
                )
            )
        # Close the current yolo session
        yolo.close_session()
        return render_template('prediction.html')


""" Running our application """
if __name__ == "__main__":
    app.run(debug=True)
```