Project Report

AI- Based Localization And Classification Of Skin Disease With Erythema

Team Id	PNT2022TMID34423
Project Name	AI- Based Localization And Classification Of Skin Disease With Erythema
Team Members	Aysha Farhana K 961819104023 Rithika Mejole X M 961819104071 Fathima Arshatha S 961819104031 Anclin Jeni I 961819104013

CONTENTS

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. Literature Survey

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. **REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

5.PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6 . **PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

7 **7.CODING & SOLUTIONING**

8. RESULTS

9.ADVANTAGES & DISADVANTAGES

9.1 Advantage

9.2 Disadvantage

10.CONCLUSION

11.FUTURE SCOPE

12.APPENDIX

Source Code

GitHub & Project Demo Link

1. Introduction

1.1 Project Overview

Now a day's people are suffering from skin diseases, more than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other.
To overcome the above problem, we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.

1.2 Purpose

The diseases are not considered skin diseases, and skin tone is majorly suffered from the ultraviolet rays from the sun. However, dermatologists perform the majority of non-invasive screening tests simply with the naked eye, even though skin illness is a frequent disease for which early detection and classification are essential for patient success and recovery. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

2. Literature Survey

2.1 Existing problem

A neglected public health problem Skin diseases are among the most common health problems in humans. Considering their significant impact on the individual, the family, the social life of patients, and their heavy economic burden, the public health importance of these diseases is underappreciated.

2.2 References

[1] J. Kawahara and G. Hamarneh, "Multi-resolution-tract CNN with hybrid pretrained and skin-lesion trained layers," in International Workshop on Machine Learning in Medical Imaging, pp. 164–171, Springer, New York, NY, USA, 2016.

[2] S. Verma, M. A. Razzaque, U. Sangtongdee, C. Arpnikanondt, B. Tassaneetrithep, and A. Hossain, “Digital diagnosis of Hand, Foot, and mouth disease using hybrid deep neural networks,” IEEE Access, vol. 9, pp. 143481–143494, 2021.
[3] P. P. Rebouças Filho, S. A. Peixoto, R. V. Medeiros da Nobrega´ et al., “Automatic histologically-closer classification of skin lesions,” Computerized Medical Imaging and Graphics, vol. 68, pp. 40–54, 2018.

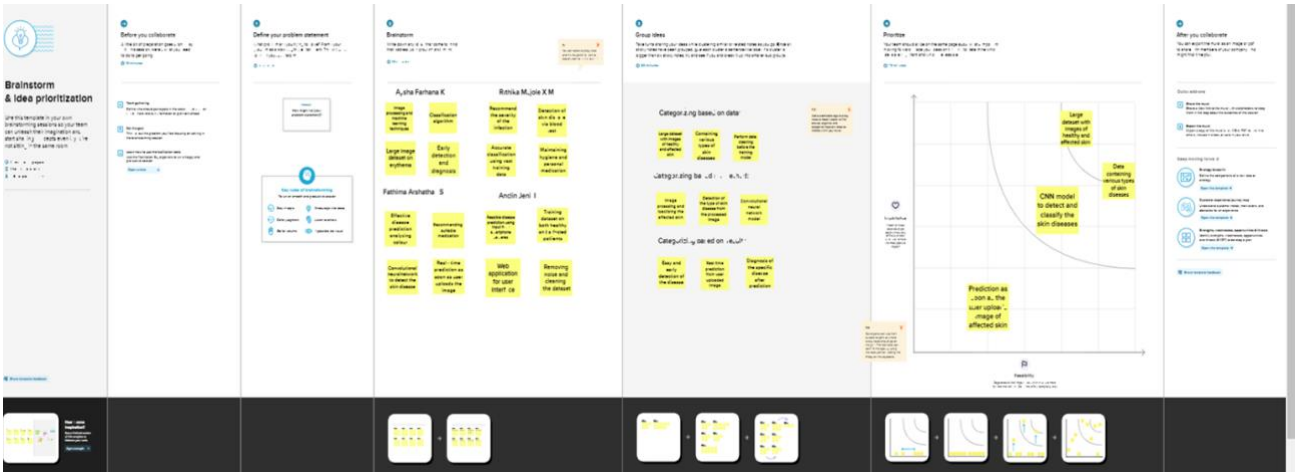
2.4 **Problem Statement Definition**

We're trying to find a solution to identify Skin Disease but Developed model is under training because given an image of skin, we can decompose, segment, and classify in a sequential manner which takes to Early detection of skin cancer, psoriasis.

3. Ideation and Proposed Solution

3.1 Empathy Map Canvas

3.2 Ideation and Brainstorming



3.4 Proposed Solution

Two-phase analysis model. The original image primarily enters a pre-processing stage, where normalization and decomposition occur. Afterwards, the first step is segmentation, where cluster of abnormal skin are segmented and cropped. The second step is classification, where each cluster is classified into its corresponding class. Developed Model is Still under training.

3.5 Problem Solution fit

Skin disease can appear in virtually any part of body and there is a lack of data required to form an association between the probability of a skin disease based on the body part. A Solution model used for the prevention and early detection of skin cancer and psoriasis by image analyses to detect whether the person is having skin disease or not. The location of the disease that is present in an image and improved performance by CNN model to focus on particular subsections of the images.

4. Requirement Analysis

4.1 Functional requirements

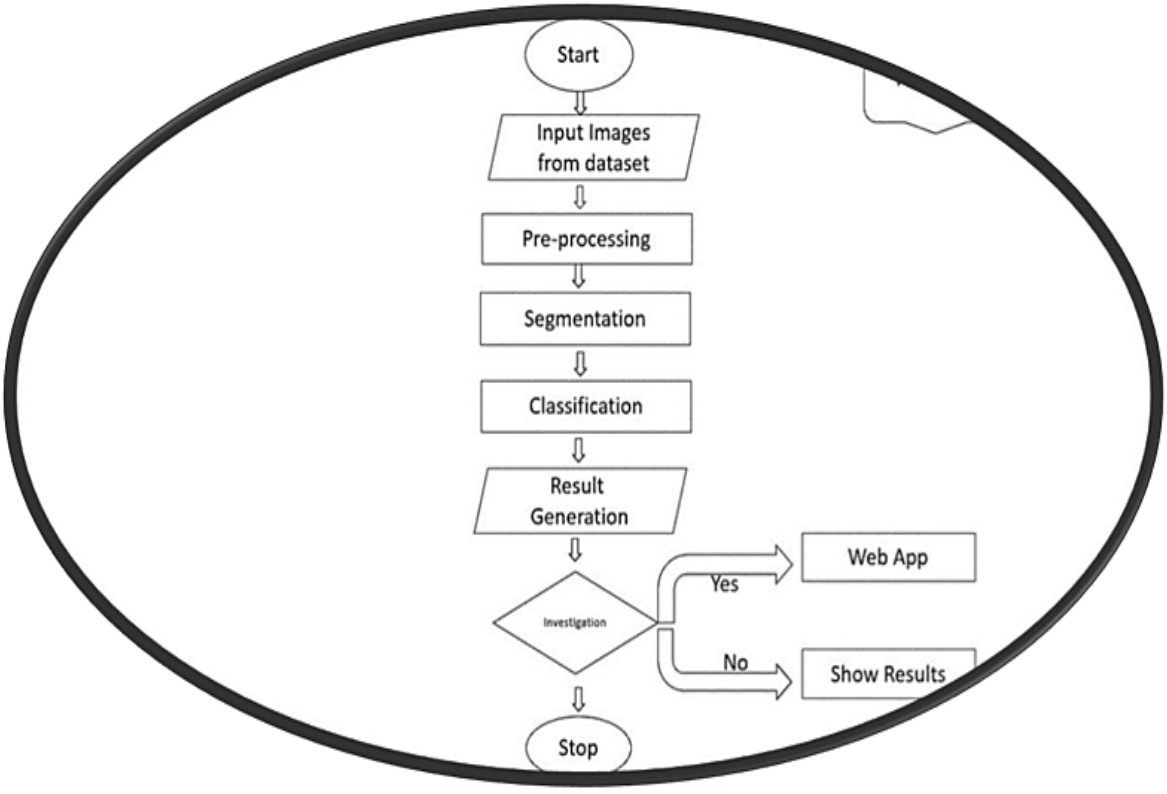
Image Acquisition, Pre-processing Steps such as Colour gradient generator on an image , Cropping and isolating region of interest and Thresholding and Clustering on image, Visual feature extraction, System Training YOLO Model for Skin disease classification with deep learning and CNN, Separate access of application for admin, Diagnosis of Skin disease and Data retrieval and Data manipulation.

4.2 Non-Functional requirements

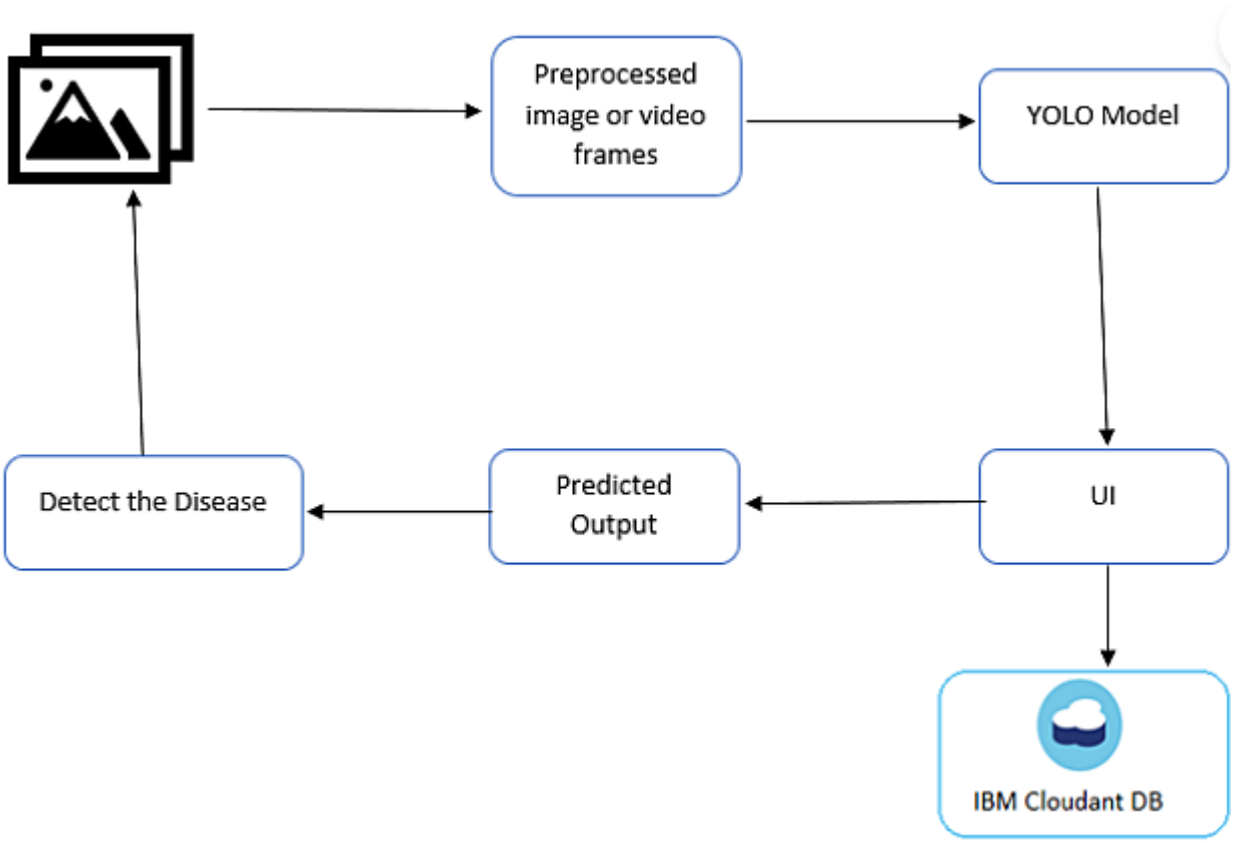
Software Quality Attributes, Prediction, Accuracy.

5. Project Design

5.1 Data Flow Diagram



5.2 Solution and Technical Architecture



5.3 User Stories

Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Prerequisites	USN-1	Install Python IDE, Python packages, Microsoft Visual Object Tagging Tool, Yolo Structure	3	High
Data Collection	USN-2	Dataset should be collected from google or using a Chrome extension such as Fatkun Batch Downloader	3	High
Annotate Images	USN-3	Create A Project in VOTT (Microsoft's Visual Object Tagging Tool)	2	Medium
Training YOLO	USN-4	train our model using YOLO weights	2	Medium

	USN-5	To Download and Convert Pre-Trained Weights	3	High
	USN-6	To Train YOLOv3 Detector	3	High
Cloudant DB	USN-7	Register & Login to IBM Cloud	3	High
	USN-8	Create Service Instant and Credentials	2	Medium
	USN-9	Launch DB and Create database	3	High
Development Phase	USN-10	To build a web application	3	High
	USN-11	Building HTML pages with python code	2	Medium
	USN-12	To run the application	3	High
Testing Phase	USN-13	As a user login to dashboard	2	Medium
	USN-14	As a user import the images with skin diseases to the software application	2	Medium
	USN-15	YOLO processes the image and give the necessary details	3	High

6.Project Planning and Scheduling

6.1 Sprint Planning and Estimation

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Prerequisites	USN-1	Install Python IDE, Python packages, Microsoft Visual Object Tagging Tool, Yolo Structure	3	High	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I

Sprint-1	Data Collection	USN-2	Dataset should be collected from google or using a Chrome extension such as Fatkun Batch Downloader	3	High	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I
Sprint-1	Annotate Images	USN-3	Create A Project in VOTT (Microsoft's Visual Object Tagging Tool)	2	Medium	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I
Sprint-2	Training YOLO	USN-4	train our model using YOLO weights	2	Medium	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I
Sprint-2		USN-5	To Download and Convert PreTrained Weights	3	High	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I
Sprint-2		USN-6	To Train YOLOv3 Detector	3	High	Aysha Farhana K Rithika Mejole X M Fathima Arshatha S Anclin Jeni I
Sprint-3	Cloudant DB	USN-7	Register & Login to IBM Cloud	3	High	Aysha Farhana K Rithika Mejole X M
Sprint-3		USN-8	Create Service Instant and Credentials	2	Medium	Fathima Arshatha S Anclin Jeni I
Sprint-3		USN-9	Launch DB and Create database	3	High	Aysha Farhana K Rithika Mejole X M

Sprint-3	Development Phase	USN-10	To build a web application	3	High	Fathima Arshatha S Anclin Jeni I
Sprint-3		USN-11	Building HTML pages with python code	2	Medium	Aysha Farhana K Rithika Mejole X M
Sprint-3		USN-12	To run the application	3	High	Fathima Arshatha S Anclin Jeni I
Sprint-4	Testing Phase	USN-13	As a user login to dashboard	2	Medium	Aysha Farhana K Rithika Mejole X M
Sprint-4		USN-14	As a user import the images with skin diseases to the software application	2	Medium	Fathima Arshatha S Anclin Jeni I
Sprint-4		USN-15	YOLO processes the image and give the necessary details	3	High	Aysha Farhana K Rithika Mejole X M

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022

6. Coding and Solutioning

Main code

```
from flask import Flask, app,request,render_template
from flask import Flask, request, render_template, redirect, url_for
from cloudant.client import Cloudant
```

```
# Authenticate using an IAM API key
client = Cloudant.iam('b696605d-6526-4485-bdfa-1c7577fa7d8f-bluemix','uGQKAW_PQI6MXV1f96UyMvEt24wx0IT4lw6ptz-T86sZ', connect=True)
# Create a database using an initialized client
my_database = client.create_database('erythema')
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('/prediction1')
def prediction1():
    return render_template('prediction1.html')
    #return render_template('prediction.html')
    """ Running our application """
if __name__ == "__main__":
    app.run(debug=True)
```

Training code

STEP1: WE ARE IMPLEMENTING YOLO IN KERAS TO TRY TO UNDERSTAND WHAT WE ARE DOING

Because our business case is quite unique, we might not need all the layers that the original yolo model offers

In [3]:

```
#Let's first load all the libraries we need

from keras.models import Sequential, Model
from keras.layers import Reshape, Activation, Conv2D, Input, MaxPooling2D,
BatchNormalization, Flatten, Dense, Lambda, Dropout
from keras.layers.advanced_activations import LeakyReLU
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
from keras.optimizers import SGD, Adam, RMSprop, Adamax
from keras.layers.merge import concatenate
import matplotlib.pyplot as plt
import keras.backend as K
import tensorflow as tf
#import imgaug as ia
from tqdm import tqdm
```

```
#from imgaug import augmenters as iaa
import numpy as np
import pandas as pd
import pickle
import os, cv2
#from preprocessing import parse_annotation, BatchGenerator
```

```
/anaconda/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [4]:

```
#custom to us are the labels and the image size
LABELS = ['melanoma', 'notmelanoma']

    #I originally tried 200 * 200 and gave me an error - this is because the input layer is a 32 neuron so we need multiples of 32
#I'm going to use 32*7 = 224

IMAGE_H, IMAGE_W = 192, 192
#Grids are used when you are facing problems with more than one object to detect and the fact they alow 2 (in the original)
#overlapping bouding boxes. In our case, we have only 1 very well defined object to detect so we don't need more than 1 grid

#GRID_H,  GRID_W  = 7 , 7

#Let's leave the rest as is

BOX              = 5
CLASS            = len(LABELS)
CLASS_WEIGHTS    = np.ones(CLASS, dtype='float32')
OBJ_THRESHOLD    = 0.3#0.5
NMS_THRESHOLD    = 0.3#0.45
ANCHORS          = [0.57273, 0.677385, 1.87446, 2.06253, 3.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.16828]

NO_OBJECT_SCALE  = 1.0
OBJECT_SCALE     = 5.0
COORD_SCALE      = 1.0
CLASS_SCALE      = 1.0
```

```
BATCH_SIZE      = 16
WARM_UP_BATCHES  = 0
TRUE_BOX_BUFFER  = 50
```

STEP 1.1: LET'S BUILD THE NETWORK

In [6]:

```
# the function to implement the orgnization layer (thanks to github.com/allanzelener/YAD2K)
def space_to_depth_x2(x):
    return tf.space_to_depth(x, block_size=2)
```

In [8]:

```
input_image = Input(shape=(IMAGE_H, IMAGE_W, 3))
#true_boxes  = Input(shape=(1, 1, 1, TRUE_BOX_BUFFER , 4))

#NOTE ON THE SINTAX = This isn't using Sequential(), it's building a pipeline of layers applied to
#the input_image. So what this is doing x = fn...(f1(x)). Nested functions. This is very useful when you need to do skip
#connections or you need to do something a bit more complex to the output, it isn't a clear sequence

# Layer 1
x = Conv2D(32, (3,3), strides=(1,1), padding='same', name='conv_1',
use_bias=False)(input_image)
x = BatchNormalization(name='norm_1')(x)
x = LeakyReLU(alpha=0.1)(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
```

Freeze layers

Freezing layers - this are needed because when we printed the original summary we have 50million+ trainable parameters and even if we have prelearnt weights, it will optimise them again

Total params: 50,695,396 Trainable params: 50,674,724 Non-trainable params: 20,672

In [ ]:

```
#for i in range(1,23):
    #model.layers[i].trainable = False
```

In [9]:

```
model.summary()
```

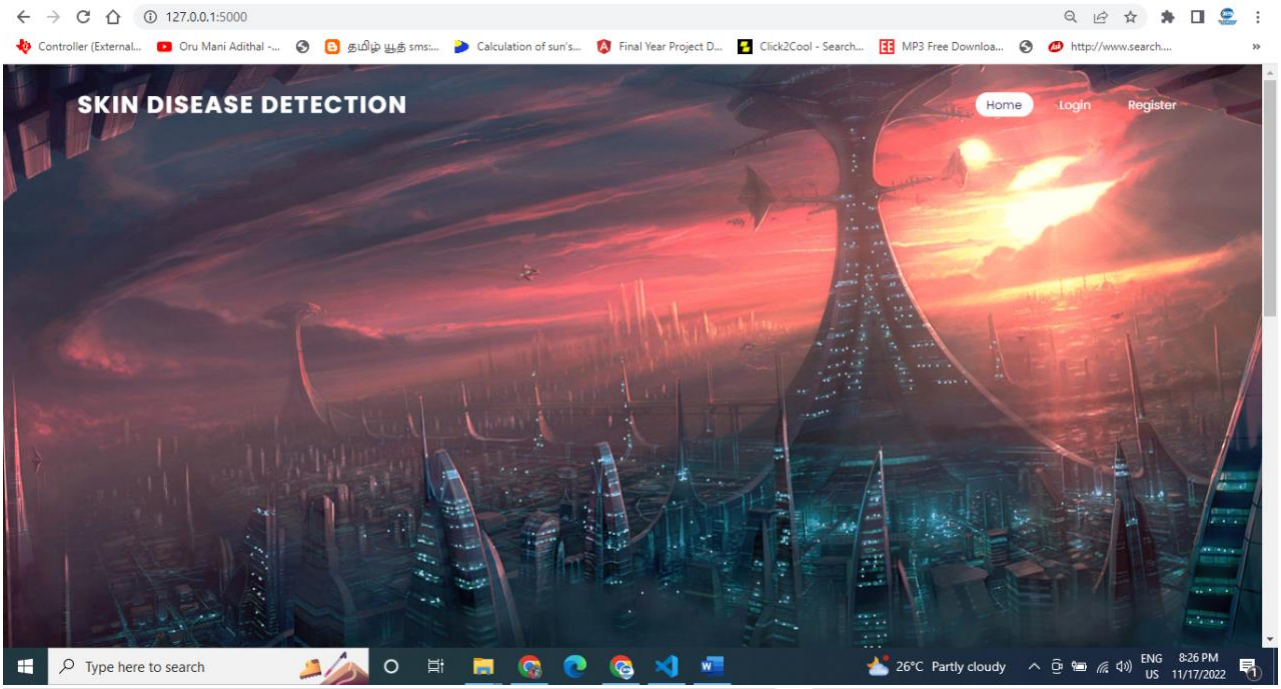
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
```

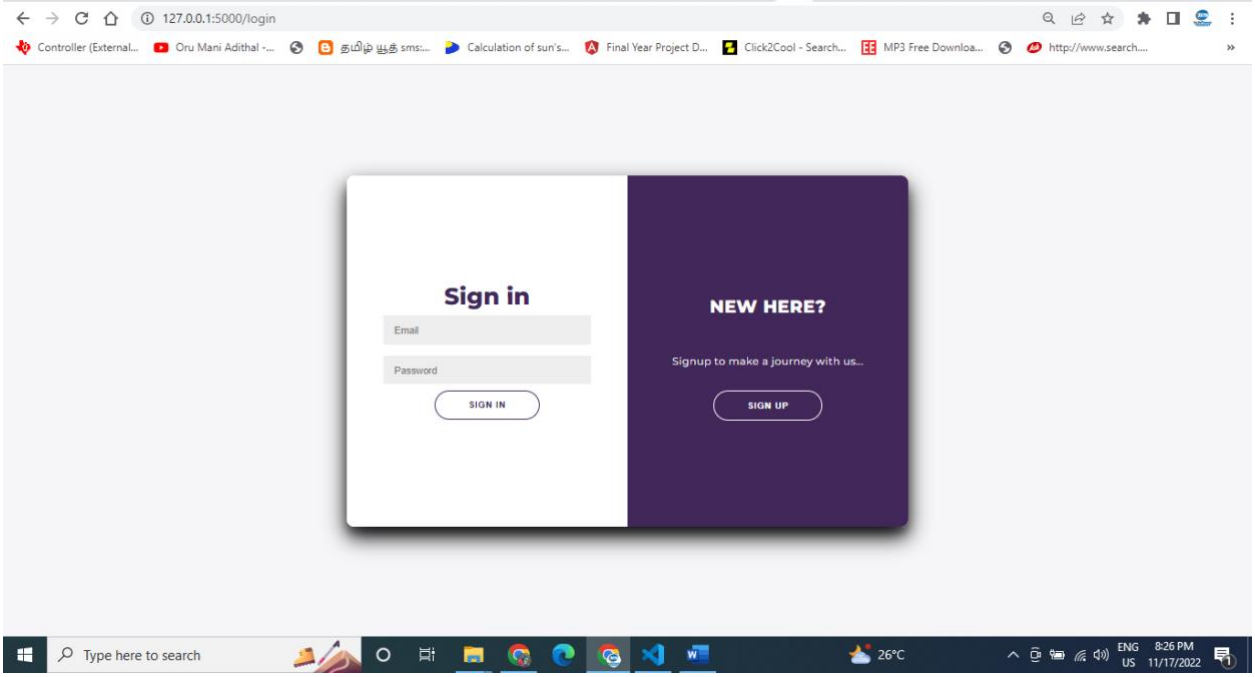
```
==================================================================================================
 input_3 (InputLayer)           (None, 192, 192, 3)  0
__________________________________________________________________________________________________
conv_1 (Conv2D)                (None, 192, 192, 32) 864         input_3[0][0]
__________________________________________________________________________________________________
norm_1 (BatchNormalization)    (None, 192, 192, 32) 128         conv_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu_44 (LeakyReLU)     (None, 192, 192, 32) 0           norm_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_11 (MaxPooling2D) (None, 96, 96, 32)   0          leaky_re_lu_44[0][0]
__________________________________________________________________________________________________
conv_2 (Conv2D)                (None, 96, 96, 64)   18432       max_pooling2d_11[0][0]
__________________________________________________________________________________________________
norm_2 (BatchNormalization)    (None, 96, 96, 64)   256         conv_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_45 (LeakyReLU)     (None, 96, 96, 64)   0           norm_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_12 (MaxPooling2D) (None, 48, 48, 64)   0          leaky_re_lu_45[0][0]
```

The other code features are submitted in GitHub refer the link
https://github.com/IBM-EPBL/IBM-Project-46399-1660746538

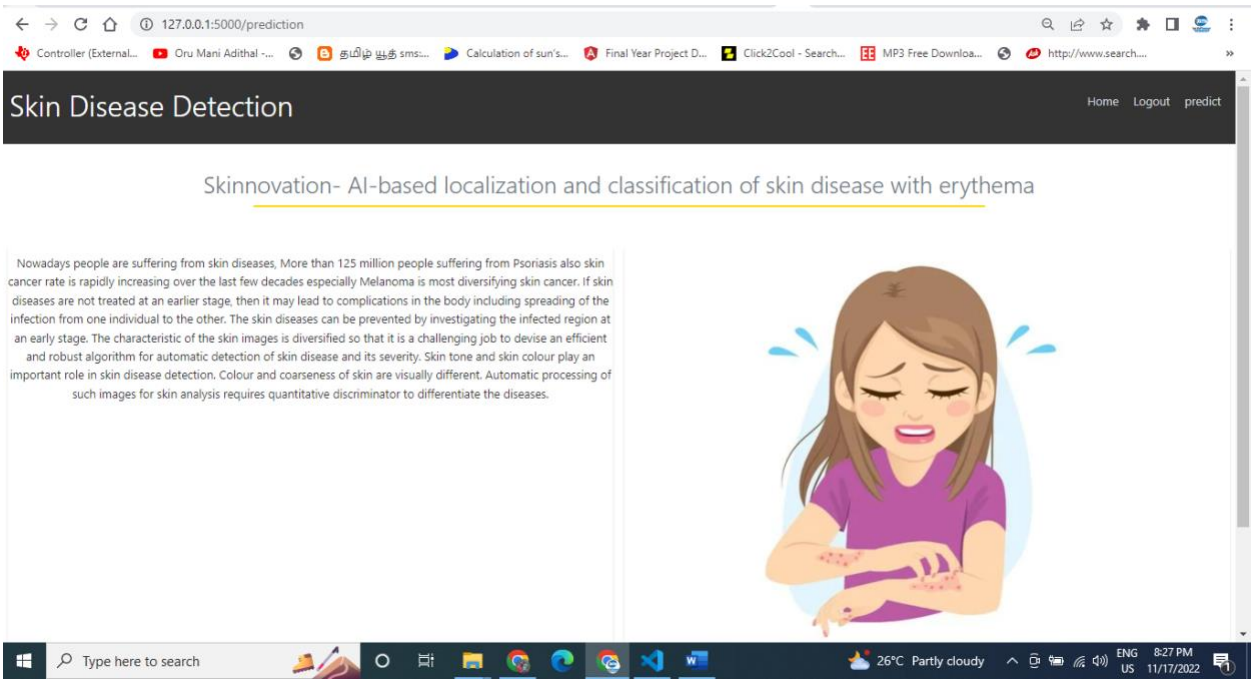
8. Results

Login Page

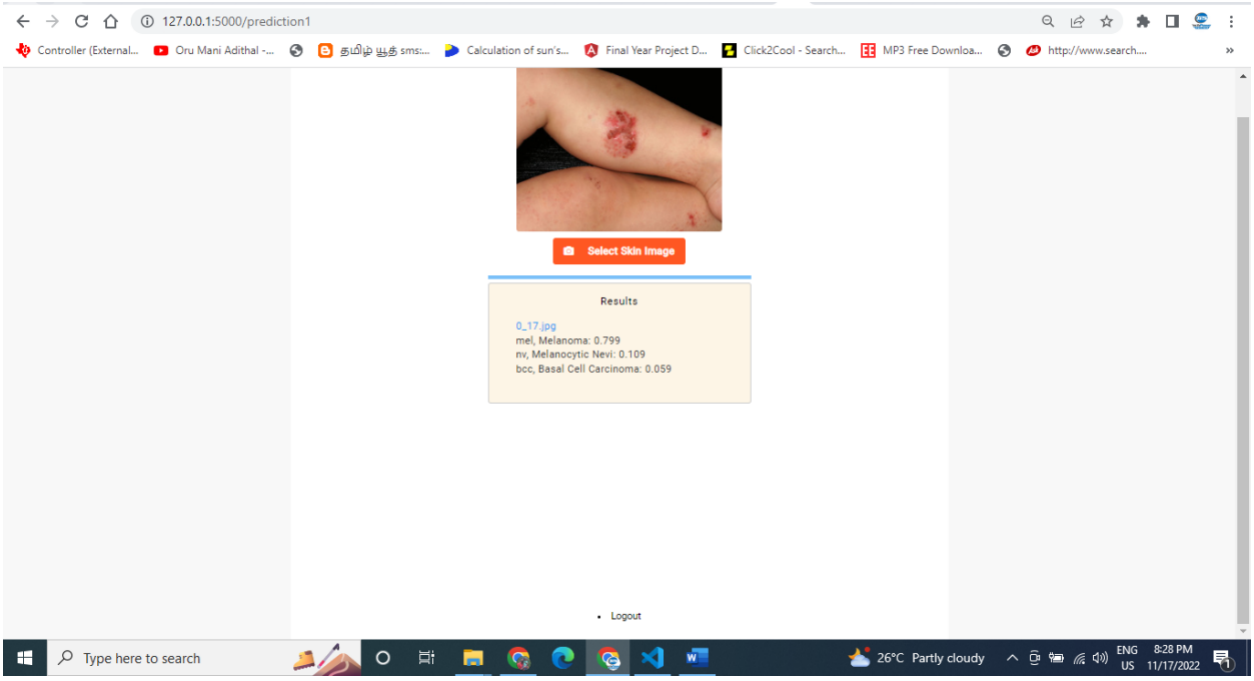

127.0.0.1:5000
SKIN DISEASE DETECTION
Home
Login
Register
Type here to search
26°C Partly cloudy
ENG US
8:26 PM 11/17/2022


127.0.0.1:5000/login
Sign in
Email
Password
SIGN IN
NEW HERE?
Signup to make a journey with us...
SIGN UP
Type here to search
26°C
ENG US
8:26 PM 11/17/2022

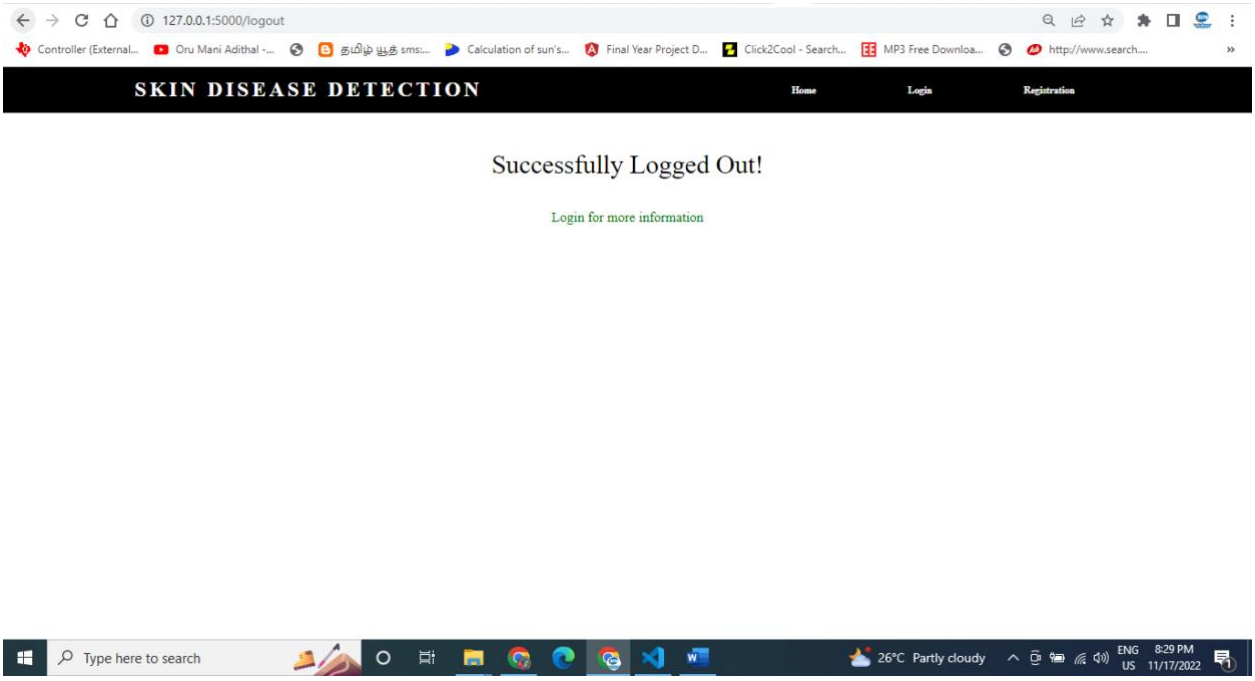
Home Page



Prediction Page



Logout Page



Trained Dataset

9. Advantage & Disadvantage

9.1 Advantages

Instant Response, improves prediction of Skin Disease, no referral needed, Saves Money and Time, and Confidential Advice.

9.2 Disadvantages

Network Connectivity and Accuracy

10. Conclusion

We have shown that even without a large dataset and high-quality images, it is possible to achieve sufficient accuracy rates. In addition, we have shown that current state-of-the-art CNN

models can outperform models created by previous research, through proper data pre-processing, self-supervised learning, transfer learning, and special CNN architecture techniques. Furthermore, with accurate segmentation, we gain knowledge of the location of the disease, which is useful in the pre-processing of data used in classification, as it allows the CNN model to focus on the area of interest. Lastly, unlike previous studies, our method provides a solution to classify multiple diseases within a single image. With higher quality and a larger quantity of data, it will be viable to use state-of-the-art models to enable the use of CAD in the field of dermatology.

11. Future Scope

This implementation of the Structural Co-Occurrence matrices for feature extraction in the skin diseases classification and the pre-processing techniques are handled by using the Median filter, this filter helps to remove the salt and pepper noise in the image processing; thus, it enhances the quality of the images, and normally, the skin diseases are considered as the risk factor in all over the world. Our proposed approach provides 97% of the classification of the accuracy results while another existing model such as FFT + SCM gives 80%, SVM + SCM gives 83%, KNN + SCM gives 85%, and SCM + CNN gives 82%. Future work is dependent on the increased support vector machine's accuracy in classifying skin illnesses, and SCM is used to manage the feature extraction technique.

12. Appendix

GitHub Link :
https://github.com/IBM-EPBL/IBM-Project-46399-1660746538
Team Id: PNT2022TMID34423
Demo link :
https://drive.google.com/file/d/1jrItt-mbfJZp857ykaucM3DZjJpGfdU9/view?usp=share_link