AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

Submitted

by:

Training Branch : B5-5M1E

Team ID : PNT2022TMID22242

MANUEL MANASSEH G

BENNY THOMAS A

BHARATH S

JA.JEBAROCK

RANJITH S

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in Information Technology

VEL TECH HIGH TECH DR. RANGARAJAN DR. SAKUNTHALA ENGINEERING COLLEGE (AN AUTONOMOUS INSTITUTION)

BONAFIDE CERTIFICATE

Certified that this project report

"AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA"

is the bonafide work of

MANUEL MANASSEH G, BENNY THOMAS A,BHARATH S, JA.JEBAROCK, RANJITH S

who carried out the project work under my supervision.

VEL TECH HIGH TECH DR.RANGARAJAN DR.SAKUNTHALA

ENGINEERING COLLEGE (AN AUTONOMOUS INSTITUTION)

AFFILIATED TO ANNA UNIVERSITY CHENNAI - 600 062

***Sri Tulasi** (INDUSTRY MENTOR)* ***R.Rajasekaran** (FACULTYMENTOR)*

ABSTRACT

Skin diseases are more common than other diseases. Skin diseases can also be the result of fungal, bacterial, allergic or viral infections, etc. A skin condition can also change the feel or color of the skin. In general, skin diseases are chronic, contagious and can sometimes progress to skin cancer. Therefore, pore and skin diseases need to be detected early to limit their improvement and spread.

The analysis and treatment of a skin disease is time consuming and economically and physically expensive for the patient. In general, most normal people no longer recognize the type and extent of skin disease. Some skin diseases show signs and symptoms after several months, causing the disease to increase and develop. This is due to a lack of clinical information in the community. It can sometimes be difficult for a dermatologist (skin specialist) to diagnose a skin condition and may order a professional laboratory test to know the full type and extent of the condition. skin condition.

We propose an image processing-based method for the diagnosis of skin diseases. This method takes digital images or video images of the affected skin and then uses image analysis to determine the type of disease. Our proposed method is simple, fast, and requires no expensive equipment other than a camera and computer.

1.INTRODUCTION:

Project Overview:

This study shows that CAD may also be a viable option in dermatology by presenting a novel method to sequentially combine accurate segmentation and classification models. Given an image of the skin, we decompose the image to normalize and extract high-level features.Using a neural network-based segmentation model to create a segmented map of the image,we then cluster sections of abnormal skin and pass this information to a classification model.We classify each cluster into different common skin diseases using an other neural network model.Our segmentation model achieves better performance compared to previous studies, and also achieves a near-perfect sensitivity score in unfavorable conditions.

Purpose :

Although computer-aided diagnosis(CAD) is used to improve the quality of diagnosis in various medica field ssuch as mammography and colonography, it is not used indermatology, where non invasive screening tests are performed only with the naked eye, and avoidable inaccuracies may exist. Our classification model is more accurate than a base line model trained without segmentation,while also being able to classify multiple diseases within a single image.This improved performance may be sufficient to use CAD in the field of dermatology.

2.LITERATURE SURVEY:

EXISTING PROBLEM:

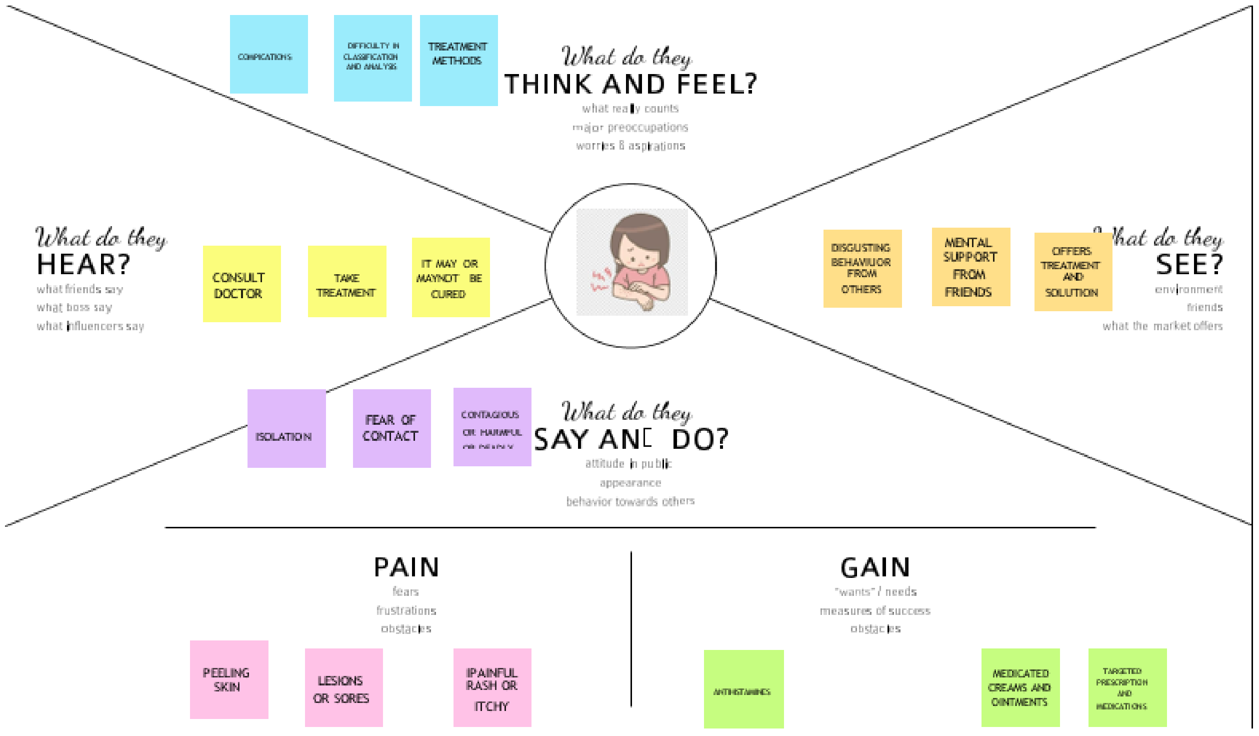
Skin diseases are the 4th common cause of skin burden worldwide. Robust and Automated system have been developed to lessen this burden and to help the patients to conduct the early assessment of the skin lesion. Mostly this system available in the literature only provide skin cancer classification. Treatments for skin are more effective and less disfiguring when found early and it is a challenging research due to similar characteristics of skin diseases. In this project we attempt to detect skin diseases. A novel system is presented in this research work for the diagnosis of the most common skin lesions (Melanocyticnevi,Melanoma, Benign keratosis-likelesions, Basalcellcarcinoma, Actinic keratoses,Vascular lesion, Dermatofibroma). The proposed approach is based on the pre-processing, Deep learning algorithm, training the model, validation and classification phase. Experiments were performed on 10010imagesand 93% accuracy is achieved for seven-class classification using Convolution Neural Networks(CNN) with the Keras Application

REFERENCES:

1. Doi, K. Computer-aided diagnosisinmedicalimaging: Historicalreview, currentstatusand futurepotential. *Comput.Med.Imaging Graph.* **31**, 198–211.

2. Yoshida, H. & Dachman, A. H. Computer-aided diagnosisfor CTcolonography. *Semin.Ultrasound CT MRI* **25**, 419–431

3. Trabelsi, O.,Tlig, L., Sayadi, M. & Fnaiech, F., Skindisease analysisandtrackingbasedonimagesegmentation. *2013International ConferenceonElectricalEngineering andSoftwareApplications*, Hammamet, 1–7.



3. Proposed Solution

CAD(ComputerAided Diagnosis)hasbeen aviableoption in dermatology bypresentinganovelmethod to sequentiallycombineaccurate segmentationandclassification models.

Problem Solution fit :

Skin disease canappearin virtually anypart of body and there is alackofdatarequired to formanassociationbetweentheprobabilityof a skindisease based on the bodypart .It is shownthat current state-of-the-art CNNmodelscanoutperform models created by previous research ,through proper data preprocessing, self-supervisedlearning, transferlearning, and specialCNNarchitecturetechniques. Furthermore, withaccurate segmentation, wegainknowledgeof thelocation of the disease, whichis usefulin the preprocessing ofdatausedin classification, as it allowsthe CNNmodelto focus onthearea ofinterest. Lastly,unlikepreviousstudies, our method provides asolution toclassify multiple diseaseswithin 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.

4.Requirement Analysis :

4.1 Functional Requirements :

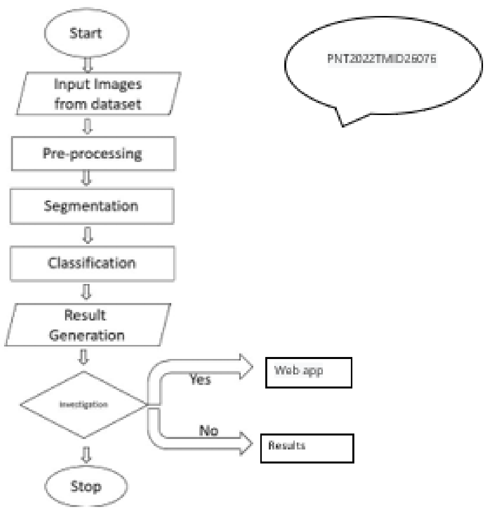
Temperature:If the temperature level exceeds theroom temperature then thealert messagewillbe sent usingGSM, **Pulse sensor**tomeasure the pulseamplitudeand width signals,**GPS** whichisused toused totrackthe livelocationof theskin disease.**GSM** results inpartialorwhole-bodyexposuresto electromagneticfield(EMF)communications,**Webcamera** collectsmedicalimagesor imagesfrom WebCam are feed into the system,,**Raspberry pimicroprocessor** in whichall othersensors, GPS and GSMareintegrated.

4.2 Non-Functional Requirements :

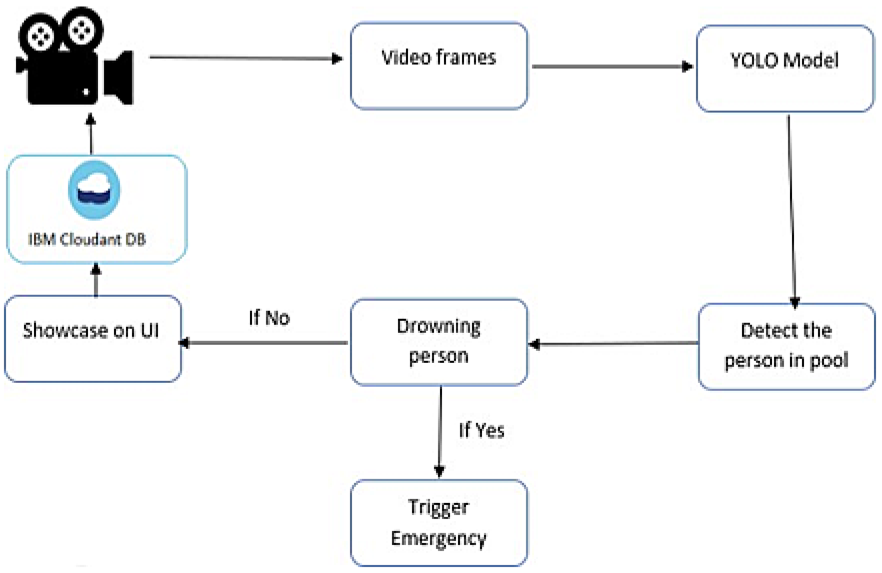
Usability, Security, Reliability, Performance,Availability, Scalability.

5. Project Design

5.1 Data Flow Diagram:



5.2 Solutionand Technical Architecture :



5.3 Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application IBM Watson Assistant	BM Watson Assistant
5.	Database	Database Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud IBM DB2, IBM Cloudant etc.	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.

11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

6.Project Planningand Scheduling

6.1 Sprint Planning and Estimation

Sprint	FunctionalRequirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint1	Login	USN-1	As a user, I can login to the dashboard by entering my email, password, and confirming my password.	7	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint1		USN-2	As a user, I will give the correctdetails about my medical report.	3	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
					Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint2	Screening	USN-3	As a user, I can find the method more efficient and accurate.	5		
Sprint1		USN-4	As a user, I can use itwith minimalphysical interactionwith the device.	3	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S

Sprint4	Physical Features	USN-5	As a user, I can use the database and software installed in a particular system	5	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint2		USN-6	As a user, I can find itportable and light weight	10	Low	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint3	Safety	USN-7	As a user, I can be safe as the detection method is free from radiations	5	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S

Sprint3	Testing	USN-8	As a user, I can undergo testing without any fearof pain as this method ispain free.	5	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S

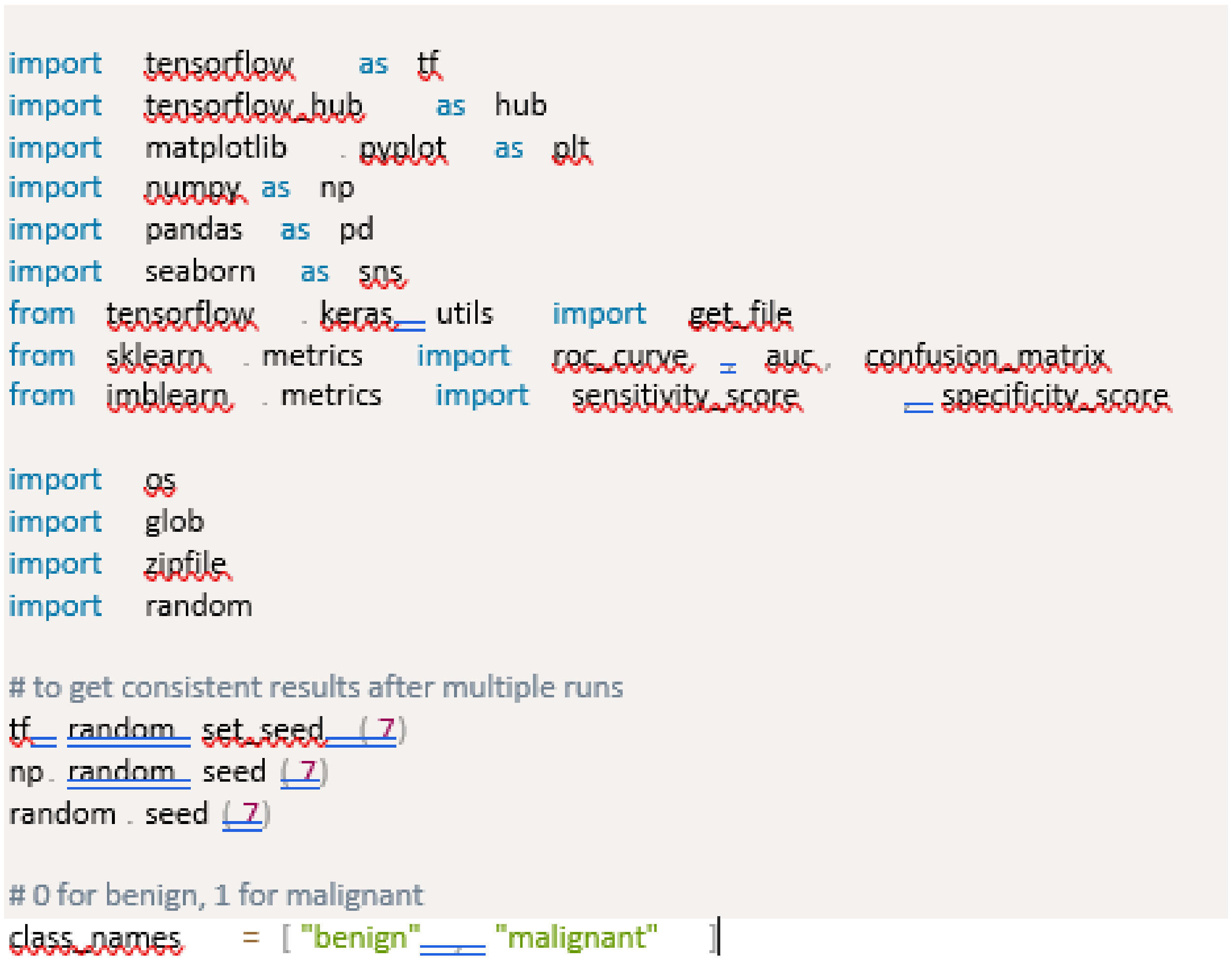
Sprint	**FunctionalRequirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-3		USN-9	As a user, Ialso suggestothers to use this software.	5	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint-2	CostEffectiveness	USN-10	As a user, I can reach many people affected from skin disease	5	Low	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S

Sprint-3		USN-11	As a user, I can create awareness among people to undergo frequentmedical check up.	5	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint-4	Results	USN-12	As a user I can rely on the resultswithout any suspicion	5	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint-4		USN-13	As a user, I can benefit from the result as itwill help me knowwhether treatmentisnecessary or not.	3	High	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint-1			As a user I can complete the screeningprocess within minutes for a single patient.	7	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S
Sprint-4			As a user I can getthe results immediately after screeningprocess.	7	Medium	Manuel.G Bharath.S Ja.Jebarock Benny.A Ranjith.S

6.2 sprint delivery schedule:

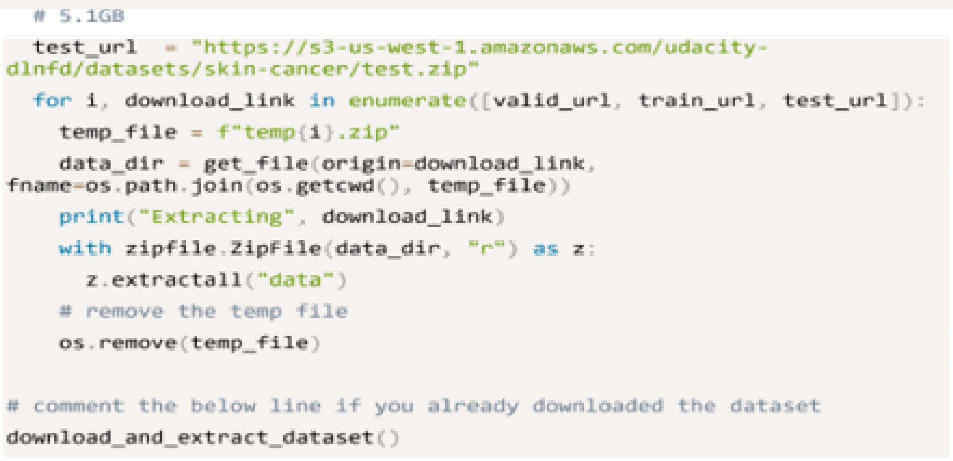
Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint ReleaseDate (Actual)**
Sprint-1	20	6 Days	24Oct2022	29Oct2022	20	29Oct2022
Sprint-2	20	6 Days	31Oct2022	05Nov 2022	20	05Nov 2022
Sprint-3	20	6 Days	07Nov 2022	12Nov 2022	20	12Nov 2022
Sprint-4	20	6 Days	14Nov 2022	19Nov 2022	20	19Nov 2022

7. Coding and Solutioning:

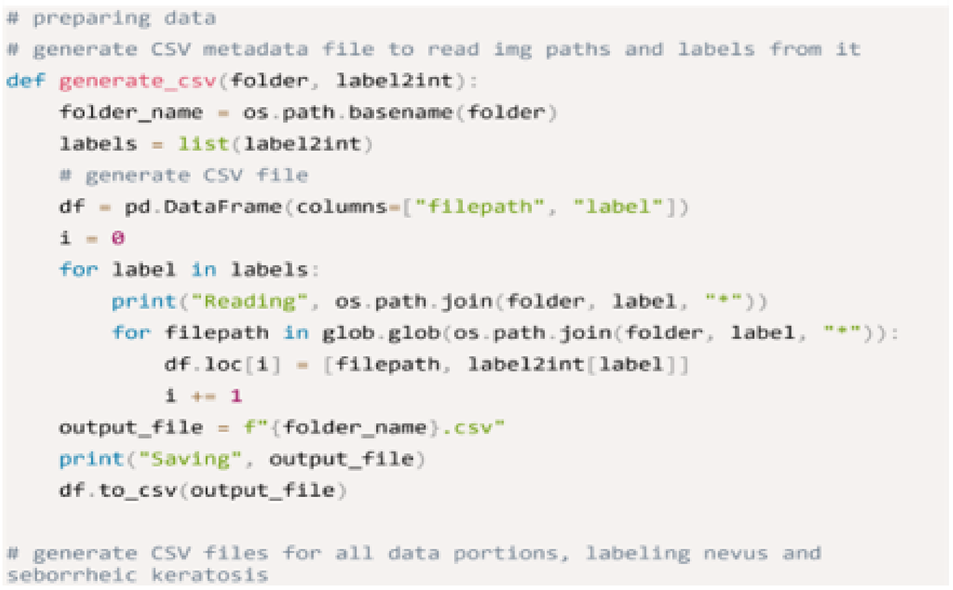


```
import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from tensorflow.keras.utils import get_file
from sklearn.metrics import roc_curve, auc, confusion_matrix
from imblearn.metrics import sensitivity_score, specificity_score

import os
import glob
import zipfile
import random

# to get consistent results after multiple runs
tf.random.set_seed(7)
np.random.seed(7)
random.seed(7)

# 0 for benign, 1 for malignant
class_names = ["benign", "malignant"]
```

```
def download_and_extract_dataset():
 # dataset from https://github.com/udacity/dermatologist-ai
 # 5.3GB
 train_url = "https://s3-us-west-1.amazonaws.com/udacitydlnfd/datasets/skin-cancer/train.zip"
 # 824.5MB   valid_url = "https://s3-us-west-
1.amazonaws.com/udacitydlnfd/datasets/skin-cancer/valid.zip"
```



```
  # 5.1GB
  test_url   = "https://s3-us-west-1.amazonaws.com/udacity-
dlnfd/datasets/skin-cancer/test.zip"
  for i, download_link in enumerate([valid_url, train_url, test_url]):
    temp_file = f"temp{i}.zip"
    data_dir = get_file(origin=download_link,
fname=os.path.join(os.getcwd(), temp_file))
    print("Extracting", download_link)
    with zipfile.ZipFile(data_dir, "r") as z:
      z.extractall("data")
    # remove the temp file
    os.remove(temp_file)

# comment the below line if you already downloaded the dataset
download_and_extract_dataset()
```



```
# preparing data
# generate CSV metadata file to read img paths and labels from it
def generate_csv(folder, label2int):
    folder_name = os.path.basename(folder)
    labels = list(label2int)
    # generate CSV file
    df = pd.DataFrame(columns=["filepath", "label"])
    i = 0
    for label in labels:
        print("Reading", os.path.join(folder, label, "*"))
        for filepath in glob.glob(os.path.join(folder, label, "*")):
            df.loc[i] = [filepath, label2int[label]]
            i += 1
    output_file = f"{folder_name}.csv"
    print("Saving", output_file)
    df.to_csv(output_file)

# generate CSV files for all data portions, labeling nevus and
seborrheic keratosis
```

```
0# as 0 (benign), and melanoma as 1 (malignant)
# you should replace "data" path to your extracted dataset path # don't replace if you used
download_and_extract_dataset() function generate_csv("data/train", {"nevus": 0, "seborrheic_keratosis": 0,
"melanoma": 1}) generate_csv("data/valid", {"nevus": 0, "seborrheic_keratosis": 0,
"melanoma": 1}) generate_csv("data/test", {"nevus": 0, "seborrheic_keratosis": 0, "melanoma": 1})
.0# loading data train_metadata_filename = "train.csv" valid_metadata_filename = "valid.csv" # load CSV files as
DataFrames df_train = pd.read_csv(train_metadata_filename) df_valid = pd.read_csv(valid_metadata_filename)
n_training_samples = len(df_train) n_validation_samples = len(df_valid) print("Number of training samples:",
n_training_samples) print("Number of validation samples:", n_validation_samples) train_ds =
tf.data.Dataset.from_tensor_slices((df_train["filepath"], df_train["label"])) valid_ds =
tf.data.Dataset.from_tensor_slices((df_valid["filepath"], df_valid["label"]))
```

```
Number of training samples: 2000
Number of validation samples: 150
```

Let's load the images:

```
# preprocess data def
decode_img(img):
  # convert the compressed string to a 3D uint8 tensor   img =
tf.image.decode_jpeg(img, channels=3)
  # Use `convert_image_dtype` to convert to floats in the [0,1] range.   img =
tf.image.convert_image_dtype(img, tf.float32)   # resize the image to the desired size.   return
tf.image.resize(img, [299, 299])
```

```
    ds = ds.batch(batch_size)
    # `prefetch` lets the dataset fetch batches in the background while
the model
    # is training.
    ds = ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
    return ds


valid_ds = prepare_for_training(valid_ds, batch_size=batch_size,
cache="valid-cached-data")
train_ds = prepare_for_training(train_ds, batch_size=batch_size,
cache="train-cached-data")
batch = next(iter(valid_ds))


def show_batch(batch):
    plt.figure(figsize=(12, 12))
    for n in range(25):
        ax = plt.subplot(5, 5, n+1)
        plt.imshow(batch[0][n])
        plt.title(class_names[batch[1][n].numpy()].title())
        plt.axis('off')


show_batch(batch)
```

```
def process_path (filepath , label ):
    # load the raw data from the file as a string
    img = tf . io . read_file ( filepath )
    img = decode_img ( img )
    return img, label


valid_ds = valid_ds . map( process_path )
train_ds = train_ds . map( process_path )
# test_ds = test_ds
for image, label in train_ds . take ( 1 ):
        print ( "Image shape:" , image . shape )
        print ( "Label:" , label . numpy())
Image shape : ( 299 , 299 , 3 )
Label : 0
```

```
# building the model
# InceptionV3 model & pre-trained weights module_url =
"https://tfhub.dev/google/tf2preview/inception_v3/feature_vector/4" m = tf.keras.Sequential([
hub.KerasLayer(module_url, output_shape=[2048], trainable=False),     tf.keras.layers.Dense(1, activation="sigmoid")
])


m.build([None, 299, 299, 3])
 m.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"]) m.summary()
Model: "sequential"
```

```
Layer (type)            Output Shape          Param #
================================================================= keras_layer (KerasLayer)     multiple
21802784  _________________________________________________________________ dense (Dense)
multiple             2049      =================================================================
Total params: 21,804,833
Trainable params: 2,049
Non-trainable params: 21,802,784
```

7.Training the Model

We now have our dataset and the model, let's get them together:

```
model_name = f"benign-vs-malignant_{batch_size}_{optimizer}" tensorboard =
tf.keras.callbacks.TensorBoard(log_dir=os.path.join("logs", model_name))

# saves model checkpoint whenever we reach better weights modelcheckpoint =
tf.keras.callbacks.ModelCheckpoint(model_name + "_{val_loss:.3f}.h5", save_best_only=True, verbose=1)


history = m.fit(train_ds, validation_data=valid_ds,          steps_per_epoch=n_training_samples // batch_size,
validation_steps=n_validation_samples // batch_size, verbose=1, epochs=100,

          callbacks=[tensorboard, modelcheckpoint])
```

Here is a part of the output during training:

```
Train for 31 steps, validate for 2 steps

Epoch 1/100

30/31 [============================>.] - ETA: 9s - loss: 0.4609 - accuracy: 0.7760

Epoch 00001: val_loss improved from inf to 0.49703, saving model to benign-vs-malignant_64_rmsprop_0.497.h5

31/31 [==============================] - 282s 9s/step - loss: 0.4646 - accuracy: 0.7722 - val_loss: 0.4970 -
val_accuracy: 0.8125

<..SNIPED..>

Epoch 27/100

30/31 [============================>.] - ETA: 0s - loss: 0.2982 - accuracy: 0.8708

Epoch 00027: val_loss improved from 0.40253 to 0.38991, saving model to benign-vs-malignant_64_rmsprop_0.390.h5

31/31 [==============================] - 21s 691ms/step - loss: 0.3025
-  accuracy: 0.8684 - val_loss: 0.3899 - val_accuracy: 0.8359

<..SNIPED..>

Epoch 41/100

30/31 [============================>.] - ETA: 0s - loss: 0.2800 - accuracy: 0.8802

Epoch 00041: val_loss did not improve from 0.38991

31/31 [==============================] - 21s 690ms/step - loss: 0.2829
-  accuracy: 0.8790 - val_loss: 0.3948 - val_accuracy: 0.8281

Epoch 42/100

30/31 [============================>.] - ETA: 0s - loss: 0.2680 - accuracy: 0.8859

Epoch 00042: val_loss did not improve from 0.38991

31/31 [==============================] - 21s 693ms/step - loss: 0.2722 - accuracy: 0.8831 - val_loss: 0.4572 -
val_accuracy: 0.8047
```

Model Evaluation :

First, let's load ourtest set, just like previously:

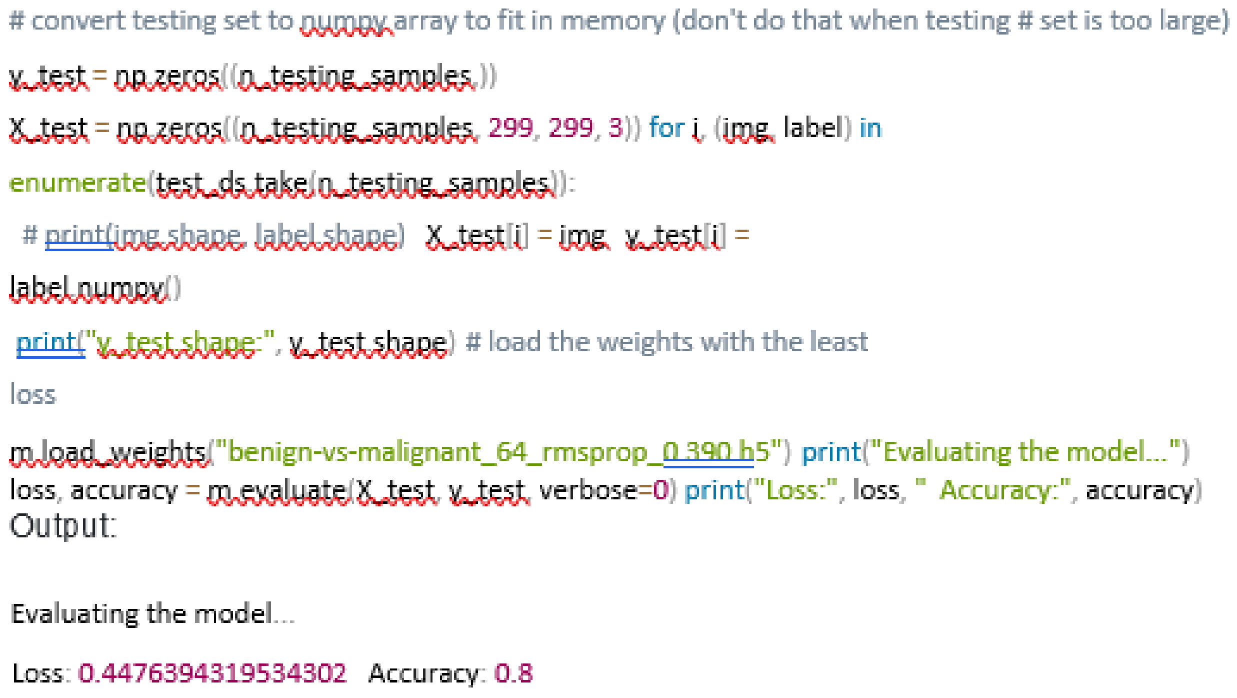
```
# evaluation # load
testing set
test_metadata_filename = "test.csv" df_test =
pd.read_csv(test_metadata_filename) n_testing_samples =
len(df_test)
print("Number of testing samples:", n_testing_samples)
test_ds = tf.data.Dataset.from_tensor_slices((df_test["filepath"], df_test["label"]))
 def prepare_for_testing(ds, cache=True, shuffle_buffer_size=1000):
```

```
if cache:    if isinstance(cache, str):      ds = ds.cache(cache)

isinstance(cache, str):      ds = ds.cache(cache)
        else   :
            ds   =  ds . cache  ()
    ds   =  ds . shuffle  ( buffer_size     =shuffle_buffer_size          )
    return     ds

test_ds     =  test_ds   . map( process_path    )
test_ds     =  prepare_for_testing     ( test_ds    ,    cache  ="test   - cached  - data"   )
```

Results :

Number of testing samples : 600

600 images of the shape(299, 299, 3) can fitour memory, let's convert our test set from `tf.data` into a NumPy array :



```
# convert testing set to numpy array to fit in memory (don't do that when testing # set is too large)
y_test = np.zeros((n_testing_samples,))
X_test = np.zeros((n_testing_samples, 299, 299, 3)) for i, (img, label) in
enumerate(test_ds.take(n_testing_samples)):
 # print(img.shape, label.shape)  X_test[i] = img  y_test[i] =
label.numpy()
 print("y_test.shape:", y_test.shape) # load the weights with the least
loss
m.load_weights("benign-vs-malignant_64_rmsprop_0.390.h5") print("Evaluating the model...")
loss, accuracy = m.evaluate(X_test, y_test, verbose=0) print("Loss:", loss, "  Accuracy:", accuracy)
```

Output:

```
Evaluating the model...
Loss: 0.4476394319534302   Accuracy: 0.8
```

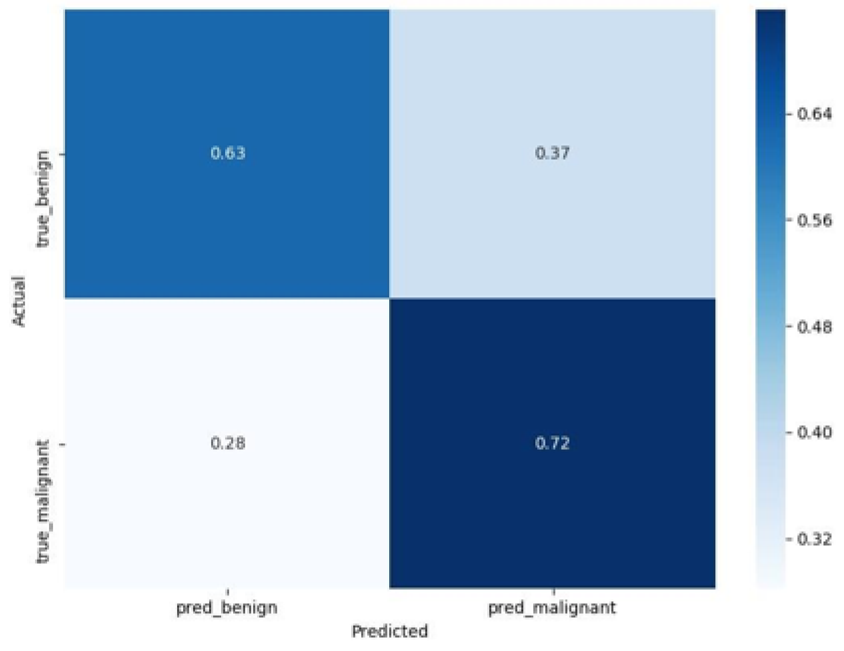
The below function does that:

```
def get_predictions(threshold=None):
    """
    Returns predictions for binary classification given `threshold`
    For instance, if threshold is 0.3, then it'll output 1 (malignant)
for that sample if
    the probability of 1 is 30% or more (instead of 50%)
    """
    y_pred = m.predict(X_test)
    if not threshold:
        threshold = 0.5
    result = np.zeros((n_testing_samples,))
    for i in range(n_testing_samples):
        # test melanoma probability
        if y_pred[i][0] >= threshold:
            result[i] = 1
        # else, it's 0 (benign)
    return result

threshold = 0.23
# get predictions with 23% threshold
# which means if the model is 23% sure or more that is malignant,
# it's assigned as malignant, otherwise it's benign
y_pred = get_predictions(threshold)
```

Now let's draw our confusion matrix and interpret it:

```
def plot_confusion_matrix(y_test, y_pred):
    cmn = confusion_matrix(y_test, y_pred)
    # Normalise
    cmn = cmn.astype('float') / cmn.sum(axis=1)[:, np.newaxis]
```

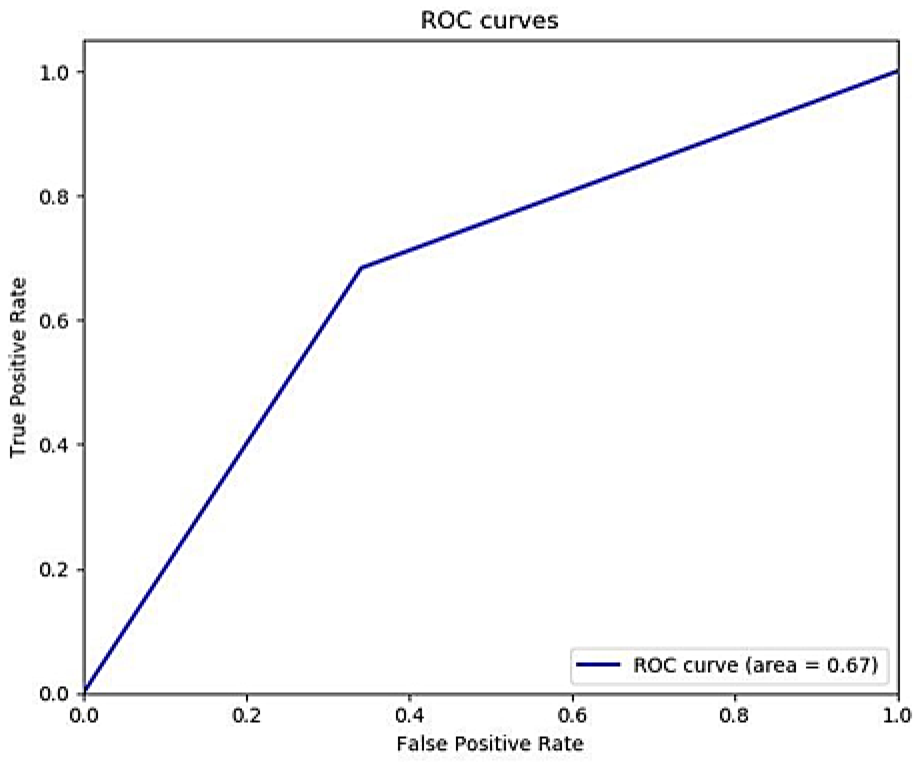
```
print (cmn)
fig , ax = plt . subplots (figsize =( 10, 10))
sns . heatmap (cmn, annot =True , fmt ='.2f' ,
                 xticklabels =[f"pred_ {c} " for c in class_names ],
                 yticklabels =[f"true_ {c} " for c in class_names ],
                 cmap="Blues"
                 )
plt. ylabel ('Actual' )
plt. xlabel ('Predicted' )
# plot the resulting confusion matrix
plt. show()

plot_confusion_matrix (y_test , y_pred )
```



```
def  plot_roc_auc  (y_true  ,  y_pred  ):
        """
        This function plots the ROC curves and provides the scores.
        """
        # prepare for figure
        plt. figure  ()
        fpr,  tpr ,  _ = roc_curve  (y_true ,  y_pred )
        # obtain ROC AUC
        roc_auc  = auc (fpr ,  tpr )
        # print score
        print  ( f"ROC AUC:   { roc_auc  : .3 f } " )
        # plot ROC curve
        plt. plot (fpr ,  tpr ,  color  ="blue"  ,  lw =2,
                          label  ='ROC curve (area = {f:.2f})'          . format  ( d=1,
f =roc_auc  ))
        plt. xlim ([ 0.0 ,  1.0 ])
        plt . ylim ([ 0.0 ,  1.05 ])
        plt. xlabel  ('False Positive Rate'       )
        plt . ylabel  ('True   Positive Rate'     )
        plt . title  ( 'ROC curves'     )
        plt. legend  (loc  ="lower right"      )
        plt . show ()

plot_roc_auc  (y_test  ,  y_pred )
```

Output:

```
ROC AUC: 0.671
```



ROCAUC: 0.671

9. Advantages and Disadvantages :

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. Inaddition, 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 inthe pre-processing of data used in classification, as it allows the CNN model tofocus 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. FutureScope :

This implementation of the Structural Co-Occurrence matricesfor 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. Ourproposed approach provides 97% of the classification of the accuracy resultswhile another existing model suchas 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 extractiontechnique.

12. Appendix :

GitHub Link : https://github.com/IBM-EPBL/IBM-Project-24564-1659944673