AI-based localization and classification of skin disease with erythema

NALAIYA THIRAN PROJECT REPORT

IBM-Project TEAMID:PNT2022TMID37911

Submitted by

KARKUVEL PRABAKARAN P	**(411419104006)**
KISHORE S	**(411419104009)**
MUTHUKUMAR S	**(411419104014)**
RANJITH KUMAR V	**(411419104018)**

Of

BATCHELOR OF ENGINNERING IN

COMPUTER SCIENCE AND ENGINEERING

NEW PRINCE SHRI BHAVANI COLLEGE OF ENGINEERING AND TECHNOLOGY

TABLE OF CONTENTS

INTRODUCTION

1.1 Project Overview

Malignant melanoma is the leading cause of death from diseases of skin Malignant melanoma is considered to be the most dangerous form of skin cancer. This type of skin cancer occurs when the human skin is exposed to the ultraviolet radiation (UV) emitted from sunshine or tanning beds, which caused the damage to skin cells. Image-based computer aided diagnosis systems have significant potential for screening and early detection of malignant melanoma. In this paper, we propose a new skin melanoma

CAD system using texture analysis methods. The proposed CAD system consists of four steps: hair removal, filtering, feature extraction and classification. Before working on the image we should remove hair from it to facilitate infected part detection. In the feature extraction step a histogram of oriented gradients (HOG) used to extract features, Our CAD system classifies between non-melanoma skin lesions (represented as common nevi or dysplastic nevi) and melanoma. The experimental results show that extracting HOG features after hair removal yields the best classification results. python software is used for skin cancer detection. Machine learning based model is implemented for detect and classify the skin disease detection and classification, flask based design is provide the user interface provide the result.

1.2 Purpose

The purpose of the project is design and implementation of deep learning model deployed for detection of image processing based skin disease.

2. LITERATURE SURVEY

2.1 Existing Problem

Kunio Doi et al [1] Computer-aided diagnosis (CAD) is a computer-based system that is used in the medical imaging field to aid healthcare workers in their

diagnoses. CAD has become a mainstream tool in several medical fields such as mammography and colonography.

Trabelsi et al [2] experimented with various clustering algorithms, such as fuzzy c-means, improved fuzzy c-means, and K-means, achieving approximately 83% true positive rates in segmenting a skin disease.

Rajab et al [3] implemented an ISODATA clustering algorithm to find the optimal threshold for the segmentation of skin lesions. An inherent disadvantage of clustering a skin disease is its lack of robustness against noise. Clustering algorithms rely on the identification of a centroid that can generalize a cluster of data. Noisy data, or the presence of outliers, can significantly degrade the performance of these algorithms.

Keke et al [4] implemented an improved version of the fuzzy clustering algorithm using the RGB, HSV, and LAB colour spaces to create a model that is more robust to noisy data.

Lu et al [5] segmented erythema in the skin using the radial basis kernel function that allows SVMs to separate nonlinear hyperplanes.

Sumithra et al [6] combined a linear SVM with a k-NN classifier to segment and classify five different classes of skin lesions.

Maglogiannis et al [7] implemented a threshold on the RGB value for segmentation and used an SVM for classification. Although more robust than clustering algorithms, SVMs are more reliant on the preprocessing of data for feature extraction. Without preprocessing that allows a clear definition of hyperplanes, SVMs may also underperform.

Albawi et al [8] Owing to the disadvantages of these traditional approaches, convolution neural networks (CNNs) have gained popularity because of their ability to extract high-level features with minimal

preprocessing10. CNNs can expand the advantages of SVMs, such as robustness in noisy datasets without the need for optimal preprocessing, by capturing image context and extracting high-level features through down-sampling. CNNs can interpret the pixels of an image within its own image-level context, as opposed to viewing each pixel in a dataset-level context. However, although down-sampling allows CNNs to view an image in its own context, it degrades the resolution of the image. Although context is gained, the location of a target is lost through down-sampling. This is not a problem for classification, but causes some difficulty for segmentation, as both the context and location of the target are essential for optimal performance. To solve this, up-sampling is needed, which works in a manner opposite to that of down-sampling, in the sense that it increases the resolution of the image. While down-sampling takes a matrix and decreases it to a smaller feature map, up-sampling takes a feature map and increases it to a larger matrix. By learning to accurately create a higher-resolution image, CNNs can determine the location of the targets to segment. Tus, for segmentation, we use a combination of down-sampling and up-sampling, whereas for classification, we use only down-sampling. To further leverage the advantages of CNNs, skip-connections were introduced, which provided a solution to the degradation problem that occurs when CNN models become too large and complex. We implement skip-connections in both segmentation and classification models. In the segmentation model, blocks of equal feature numbers are connected between the down and up-sampling sections. In the classification model, these skip-connections exist in the form of inverted residual blocks. This allows our models to grow in complexity without any performance degradation.

2.2 References

1. Doi, K. Computer-aided diagnosis in medical imaging: Historical review, current status and future potential. Comput. Med. Imaging Graph. 31, 198–211 (2007).
2. 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 (2013).

3. Rajab, M. I., Woolfson, M. S. & Morgan, S. P. Application of region-based segmentation and neural network edge detection to skin lesions. Comput. Med. Imaging Graph. 28, 61–68 (2004).
4. 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 (2010).
5. 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 (2010).
6. Sumithra, R., Suhil, M. & Guru, D. S. Segmentation and classification of skin lesions for disease diagnosis. Proced. Comput. Sci. 45, 76–85. (2015).
7. 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 Lecture Notes in Computer Science Vol. 3955 (eds Antoniou, G. et al.) 214–223 (Springer, Berlin, 2006).
8. 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 (2017).

2.3 Problem Statement Definition

Cardiologists by using various values which occurred during the ECG recording can decide whether the heartbeat is normal or not. Since observation of these values are not always clear, existence of automatic ECG detection system is required.

Luz, Eduardo José da S., et al. “ECG-based heartbeat classification for arrhythmia detection: A survey. Computer methods and programs in bio medicine 127(2016):144-164”.

Romdhane, Taissir Fekih, and Mohamed Atri Pr. "Electrocardiogram heartbeat classification based on a deep convolutional neural network and focal loss. Computers in Biology and Medicine 123 (2020):103866”.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

EMAPATHY MAP

THINK

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. The skin diseases can be prevented by investigating the infected region at an early stage. 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. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

DOES

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.

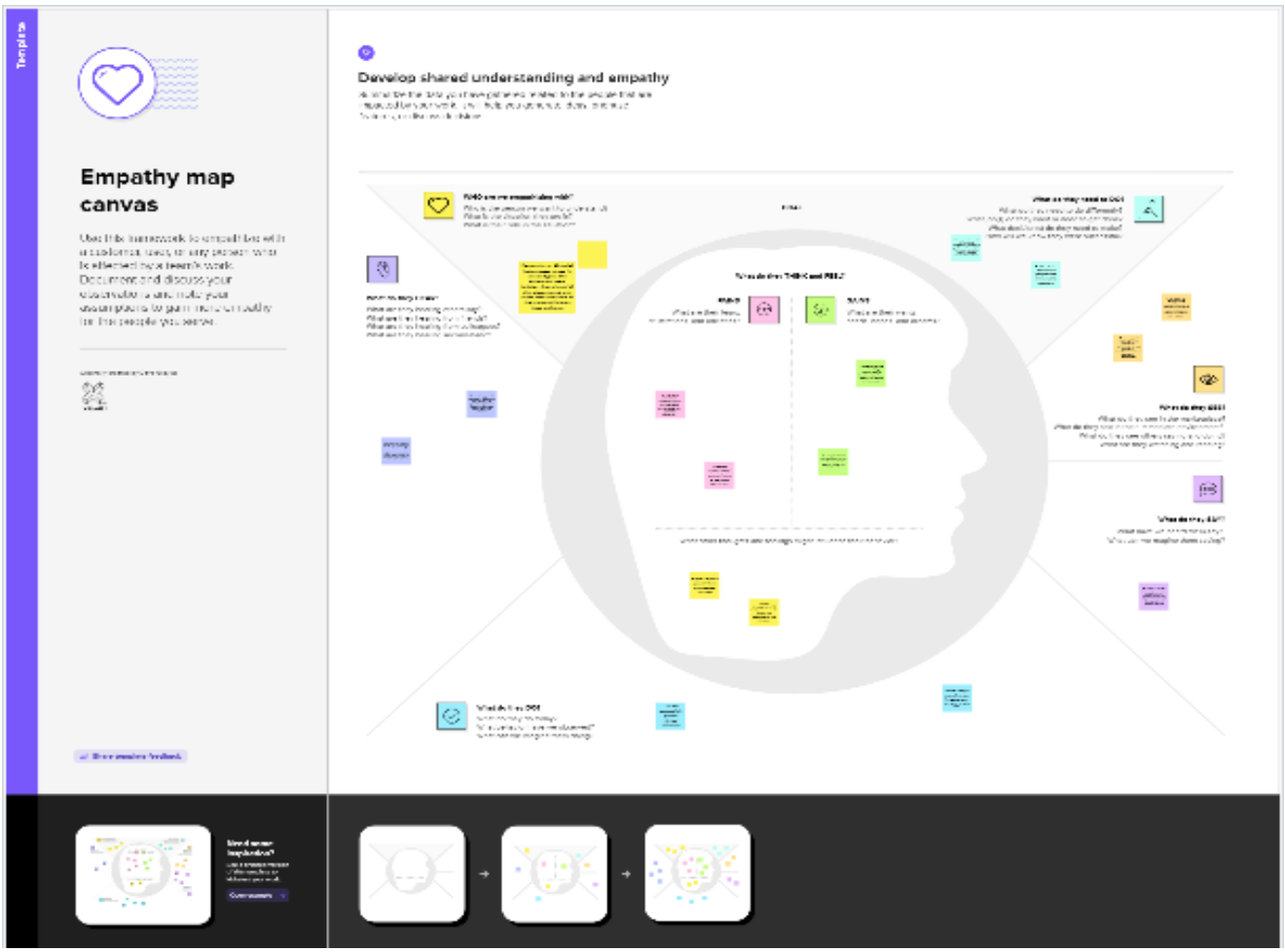
SAYS

we present a method to sequentially combine two separate models to solve a larger problem. In the past, skin disease models have been applied to either segmentation or classification. In this study, we sequentially combine both models by using the output of a segmentation model as input to a classification model. In addition, although past studies of non-CNN segmentation models used innovative preprocessing methods, recent CNN developments have focused more on the architecture of the model than on the preprocessing of data. As such, we apply an innovative preprocessing method to the data of our CNN segmentation model. The methods described above lack the ability to localize and classify multiple diseases within one image; however, we have developed a method to address this problem. Our objective is two-fold. First, we show that CAD can be used in the field of dermatology. Second, we show that state-of-the-art models can be used with current computing power to solve a wider range of complex problems than previously imagined.

FEELS

AI-Based Localization And Classification Of Skin Disease With Erythema method is help to prevent the skin cancer in the early stage by detecting and classifying the type of skin cancer.

3.2 Ideation And Brain Storming



3.3 Proposed Solution

Two-phases 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. Development Model is still under training.

3.4 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 by CNN model a focus on particular subsections of the images.

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

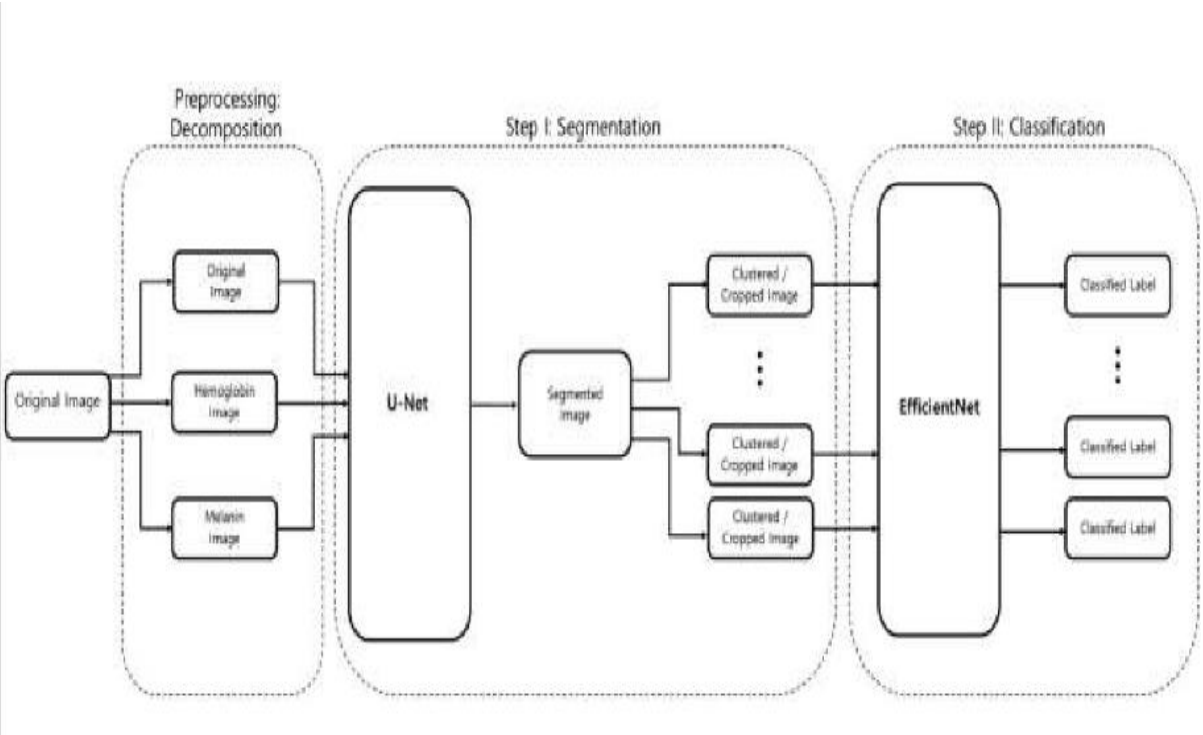
Image Acquisition, Pre-processing Steps such as Colour gradient generator on an image, Cropping and isolating region of interest and Thresholding and clustering on image, Visual feature extraction, system Training YOLO Model for Skin disease classification with deep learning and CNN, Separate access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

4.2 Non-Functional Requirements

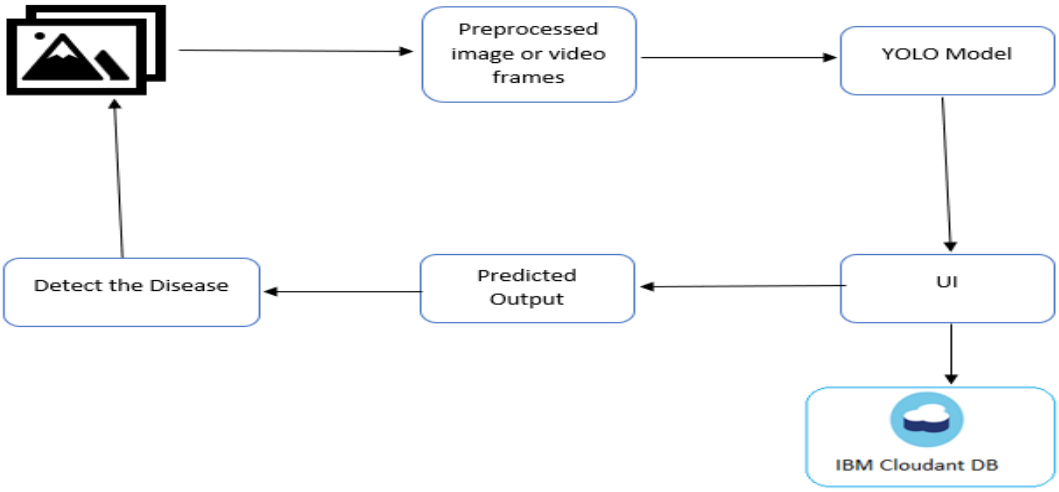
Software Quality Attributes, Prediction, Accuracy, usability, security, Reliability, Performance, Availability, Scalability.

5. PROJECT DESIGN

5.1 Data Flow Diagram



5.2 Solution And Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register &access dashboard through Gmail Id	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register &access dashboard through email	High	Sprint-1
	Dashboard	USN-6	As a user I can see my profile, medical history, uploaded images, getting report services	I can choose anyone of the service and use	Medium	Sprint-2
	Data input	USN-7	As a user I can upload the images of skin disease affected area	I can submit it to the application	High	Sprint-2
Administrator	Train model(Yolo)	USN-8	As a administrator I can train a model to compare the images uploaded with the images in the database to detect the disease	I can test the model whether it meets the criteria	High	Sprint-3
Trained model	Image Processing	USN-9	By comparing the images uploaded in the dashboard the disease will be detected	All the necessary operation performed and information extracted	High	Sprint-3
	Report Generation	USN-10	Based on the detection of disease report is generated	Result will be displayed on the screen	High	Sprint-4

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	Login	USN-1	As a user, I can login to the dashboard by entering my email, password, and confirming my password.	7	High	Karkuvel Prabakaran P, Ranjith Kumar V
Sprint-1		USN-2	As a user, I will give the correct details about my medical report.	3	High	Muthukumar S, Kishore S
Sprint-2	Screening	USN-3	As a user, I can find the method more efficient and accurate.	5	Medium	Karkuvel Prabakaran P
Sprint-1		USN-4	As a user, I can use it with minimal physical interaction with the device.	3	Medium	Ranjith Kumar V
Sprint-	Physical	USN-5	As a user, I can	5	High	Muthukumar S

4	Features		use the database and software installed in a particular system			Karkuvel Prabakaran P
Sprint-2		USN-6	As a user, I can find it portable and light weight	10	Low	Ranjith Kumar v, Kishore S, Muthukumar S
Sprint-3	Safety	USN-7	As a user, I can be safe as the detection methodis free from radiations	5	Medium	Karkuvel Prabakaran P, Ranjith Kumar V
Sprint-3	Testing	USN-8	As a user, I can undergo testing without any fear of pain as this method is pain free.	5	High	Muthukumar S, Kishore S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-9	As a user, I also suggest others to use this software.	5	High	Ranjith Kumar V, Kishore S
Sprint-2	Cost Effectiveness	USN-10	As a user, I can reach many people affected from skin disease	5	Low	Kishore S, Karkuvel Prabakaran P
Sprint-3		USN-11	As a user, I can create awareness among people to undergo frequent medical checkup.	5	Medium	Karkuvel Prabakaran P, Muthukumar S

Sprint-4	Results	USN-12	As a user I can rely on the results without any suspicion	5	Medium	Karkuvel Prabakaran P, Ranjith Kumar V, Kishore P

Sprint-4		USN-13	As a user, I can benefit from the result as it will help me know whether treatment is necessary or not.	3	High	Muthukumar S
Sprint-1			As a user I can complete the screening process within minutes for a single patient.	7	Medium	Karkuvel Prabakaran P
Sprint-4			As a user I can get the results immediately after screening process.	7	Medium	Kishore P Ranjith Kumar V Muthukumar 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 Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	03 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	08 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

7. CODING AND SOLUTION

Final Code

```
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"]
```

Preparing the Dataset

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

```
# 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})
```

```
# 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
```

Copy

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])
```

```
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
```

```
# training parameters
batch_size = 64
optimizer = "rmsprop"


def prepare_for_training(ds, cache=True, batch_size=64,
shuffle_buffer_size=1000):
  if cache:
    if isinstance(cache, str):
      ds = ds.cache(cache)
    else:
      ds = ds.cache()
  # shuffle the dataset
  ds = ds.shuffle(buffer_size=shuffle_buffer_size)
  # Repeat forever
  ds = ds.repeat()
```

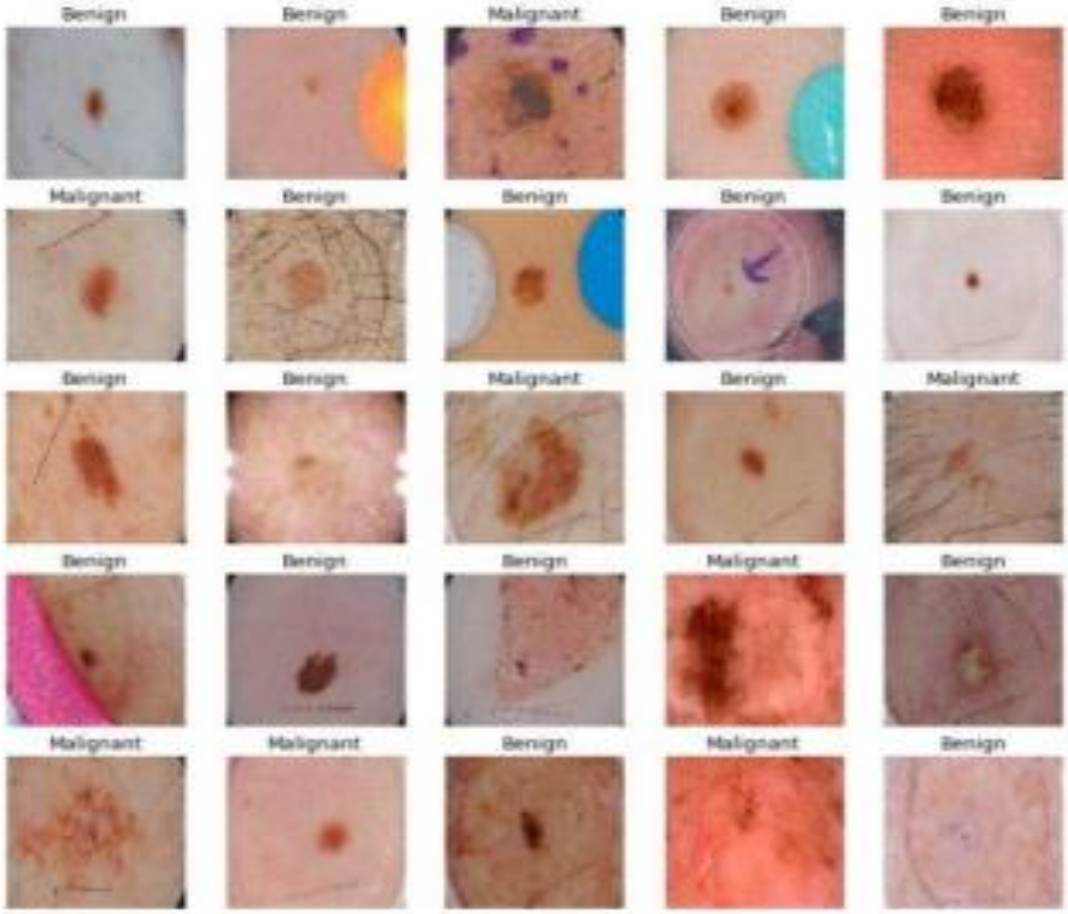
```
  # split to batches
  ds = ds.batch(batch_size)
  # `prefetch` lets the dataset fetch batches in the background while
the model
  # is training.
  ds = ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
  return ds


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)
```

Output:




```
# building the model
# InceptionV3 model & pre-trained weights
module_url = "https://tfhub.dev/google/tf2-
preview/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
_________________________________________________________________
```

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 our test 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)
    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")
```

The above code loads our test data and prepares it for testing:

```
Number of testing samples: 600
```

`600` images of the shape `(299, 299, 3)` can fit our 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)
```

Copy

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)
```

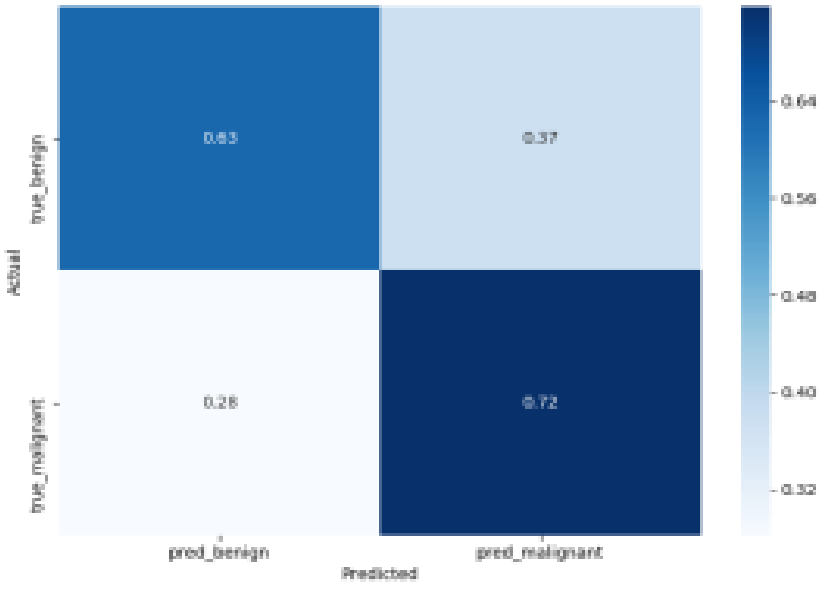
Now let's draw our confusion matrix and interpret it:

```
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 it
  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)
```

Output:

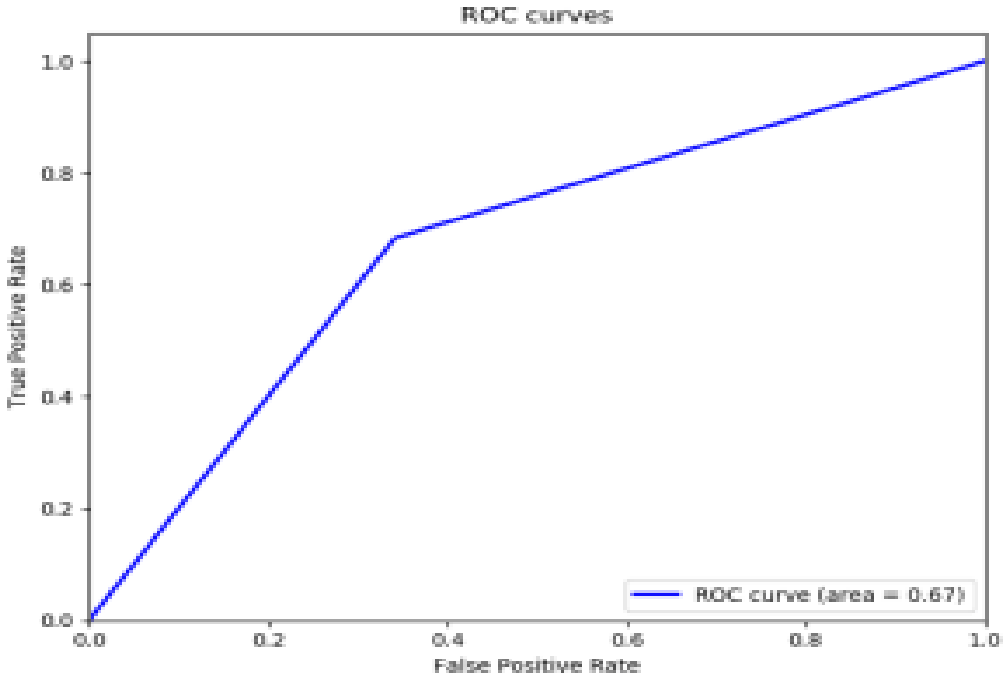


```
def plot_roc_auc(y_true, y_pred):
    """
    This function plots the ROC curves and provides the scores.
    """
    # prepare for figure
    plt.figure()
    fpr, tpr, _ = roc_curve(y_true, y_pred)
    # obtain ROC AUC
    roc_auc = auc(fpr, tpr)
    # print score
    print(f"ROC AUC: {roc_auc:.3f}")
    # plot ROC curve
    plt.plot(fpr, tpr, color="blue", lw=2,
                label='ROC curve (area = {f:.2f})'.format(d=1,
f=roc_auc))
    plt.xlin([0.0, 1.0])
    plt.ylin([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)
```

Copy

Output:


ROC curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.67)

```
ROC AUC: 0.671
```

8. CONCLUSION

This project designed the deep learning based CNN (Convolution Neural Network) based algorithm with consists in put layer, convolution layer, Re layer, Max pooling layer for extract the feature for training of images.

9. APPENDIX

Github Link:

IBM-Project-42187-1660655001

TEAM ID: PNT2022TMID37911