MINI PROJECT REPORT

on

SKIN DISEASE DETECTION

Submitted in partial fulfilment for the completion of

BE-VI Semester

In

INFORMATION TECHNOLOGY

By

Mahitha Kothapally (160118737009)
Swetha Valakonda(160118737020)

Under the guidance of

Smt. T. Prathima
Professor,
Dept. of IT, CBIT.

DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)
GANDIPET, HYDERABAD – 500 075
Website: www.cbit.ac.in

2020-2021

CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY (A)
Kokapet (Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

ISO Certified
9001:2015

COMMITTED TO
RESEARCH,
INNOVATION AND
EDUCATION
42 years

CERTIFICATE

This is to certify that the project work entitled "**SKIN DISEASE DETECTION**" submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY,** in partial fulfilment of the requirements for the award of the completion of VI semester of B.E in Information Technology, during the academic year 2020-2021, is a record of original work done by **Mahitha Kothapally (160118737009), Swetha Valakonda (160118737020)** during the period of study in Department of IT, CBIT, HYDERABAD, under our supervision and guidance.

Project Guide
Smt. T. Prathima
Professor, Dept. of IT,
CBIT, Hyderabad.

Head of the Department
Dr. K. Radhika
Professor, Dept
CBIT, Hyderabad.

CONTENTS

DECLARATION

This is to certify that the work reported in the present report titled “SKIN DISEASE DETECTION**”** is a record of work done by is in the department of information technology, Chaitanya Bharathi Institute of Technology, Hyderabad.

No part of the report is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred. The reported results are based on the project work done entirely by us and not copied from any other source.

Mahitha Kothapally (160118737009)

Swetha Valakonda(160118737020)

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Smt. T. Prathima**, our project guide, for his invaluable guidance and constant support, along with his capable instruction and persistent encouragement.

We are grateful to our Head of Department, **Dr. K. Radhika**, for her steady support and the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, **Prof. GPS Varma**, as well as the management, for having designed an excellent learning atmosphere.

Our thanks to all members of the staff and our lab assistants for helping us to carry out the groundwork of this project.

We also take this opportunity to thank our parents for their support to complete the project.

ABSTRACT

Skin disease is the most common health issue worldwide. Diagnosis of skin disease is extremely difficult because of its complexities of skin tone, color, and presence of hair. It is necessary to develop automatic methods to increase the accuracy of the diagnosis of multiple types of skin disease. The advancement in technology has made it possible to diagnose skin diseases more accurately, but these diagnoses are expensive. This project aims to detect the type of skin disease using deep learning techniques. This model will provide a screening test to diagnose the initial stage of the disease. It can be developed using convolutional neural networks (CNN). Convolutional Neural Network (CNN) is a deep learning approach that is broadly used for solving complex problems.

The dataset for this project is taken from Kaggle. In this project, different skin diseases like Acne, Bullous, Atopic Dermatitis, etc. can be identified. Firstly, skin images were pre-processed to remove noise to get an enhanced image. Later feature extraction is done using convolutional neural networks (CNN), giving a diagnosis report as output. This method gives more accuracy and gives results faster than any normal method. Then the image will be sent into the model and will predict the type of skin disease

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation

Skin is the largest organ of the body. It has great exposure to the outside and can be affected easily. skin diseases vary greatly in symptoms and severity. Some skin conditions are often temporary or permanent and painless or painful and have situational causes while others could also be genetic. Most of them are minor but some are life-threatening which is why it is necessary to have a proper diagnosis.

Skin diseases have caused vast economic burdens not only in high economic countries but also in low-income countries. In the United States, it was reported that the economical burden of skin diseases surpassed past estimates severalfold. The amount of estimated resources dedicated to skin disease management is far more than required to treat conditions.

Based on a survey in 2018, One hundred and eighty-one participants were found to have a skin disease (53%). Diagnosis of skin disease is extremely difficult because of its complexities of skin tone, color, and presence of hair, etc. Most of the time people ignore if they see any symptom, it may sometimes lead to a serious disease.

We have to take good care of our skin because if skin disease is not treated earlier it may spread to different parts of the body and sometimes it can spread to other people. Hence there should be methods to diagnose the disease at its initial stage. But it is difficult to build a system to identify the disease since there are huge similarities between different diseases.

So, it is necessary to develop automatic methods to increase the accuracy of the diagnosis of multiple types of skin disease. The advancement in technology has made it possible to diagnose skin diseases more accurately, but these diagnoses are expensive.

1.2. Basic Definitions

1.2.1 NumPy: It is a python library that provides a simple data structure that is a one-dimensional array. NumPy means Numerical Python which consists of tons of mathematical and logical operations that can be performed on arrays. NumPy aims to supply an array object that's up to 50x faster than traditional Python lists.

1.2.2 pandas: It is a software library written for the Python programming language. It is used for data analysis and data manipulation. Pandas offer various operations such as merging, reshaping, selecting, and also data cleaning, and data wrangling features.

1.2.3 OpenCV: It is a library in python which is designed to solve problems in computer vision. It mainly focuses on image pre-processing and analysis including features like face detection and object detection.

1.2.4 TensorFlow: It is an open-source software library developed by Google for fast numerical computations. It is used for deep learning applications and also supports machine learning techniques. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML.

1.2.5 Keras: It acts as an interface for the TensorFlow library. Keras is a neural network library that only provides high-level APIs. It was developed in python which is easy to debug and allows ease for extensibility. Keras is user-friendly, easy to extend, and easy to use.

1.2.6 Matplotlib: It is a plotting library in python which is used for the visualization of data. It is a multi-platform library built on NumPy arrays and visual access to huge amounts of data in easily digestible visuals. Matplotlib provides different ways of constructing a figure and can interact with a handful of backends.

1.2.7 Scikit-learn: It is a machine learning library for the Python programming language. sklearn is perhaps the foremost useful library and consists of tons of efficient tools for machine learning and statistical modelings like classification, regression, and clustering.

1.3. Problem Statement

Nowadays skin diseases have become a more common problem in human life. Most of these diseases are dangerous and harmful, particularly if not treated at an initial stage. People do not treat skin diseases seriously. Sometimes, most of the people treat these infections of the skin using their household methods. However, if these household treatments are not suitable for that particular skin problem then it would affect the skin. Also, they may not be aware of the severe problem of skin diseases. Skin diseases have a tendency to pass from one person to another person easily. Hence it is very important to control it at an earlier stage to prevent it from spreading to people. The damage done to the skin due to skin diseases also could damage the self-confidence, mental confidence as well as well-being of people. Therefore skin diseases are becoming a huge problem among people. It has become an important thing to treat these skin diseases properly at the earlier stages itself to prevent serious skin damage. This system would help to solve this problem to a great extent. This system would allow users to determine the skin diseases by making use of images of skin infected with the disease. These images are sent to the model and it predicts the type of skin disease.

2. LITERATURE SURVEY

Skin Disease Recognition using Texture Analysis:

This research describes skin disease recognition by using neural networks which are based on texture analysis. Computer algorithms that contain a few steps that involve image processing, image feature extraction and classification of data have been implemented with the help of classifiers such as artificial neural networks (ANN). The ANN can learn patterns of symptoms of particular diseases and provides faster diagnosis and recognition than a human physician. Thus, the patients can do the treatment for the skin disease faced immediately based on the symptoms detected.

Classification of skin disease using data mining Techniques:

In this research paper, they used a new method, which applies five different data mining techniques, and then developed an ensemble approach that consists of all the five different data mining techniques as a single unit. They use informative Dermatology data to analyze different data mining techniques to classify the skin disease and then, an ensemble machine learning method is applied.

Studies on different CNN algorithms for Face Skin disease classification based on clinical Image:

This paper studied different CNN algorithms for face skin disease classification based on clinical images. They performed studies using five mainstream network algorithms to classify these diseases in the dataset and compared the results. Then, performed studies using an independent dataset of the same disease types, but from other body parts, to perform transfer learning on the models. Comparing the performances, the models that used transfer learning achieved a higher average precision and recall for almost all structures.

Detection of skin disease using metaheuristic supported artificial neural networks:

In this work, a technique based on the meta-heuristic supported artificial neural network has been proposed to classify images. Here 3 common skin diseases have been considered namely angioma, basal cell carcinoma, and lentigo simplex. Images have been obtained from the International Skin Imaging Collaboration (ISIC) dataset. A popular multi-objective optimization method called Non-dominated Sorting Genetic Algorithm - II is employed to train the ANN (NNNSGAII). Different features have been extracted to train the classifier. A comparison has been made with the proposed model and two other popular meta-heuristic based classifiers namely NN-PSO (ANN trained with Particle Swarm Optimization) and NN-GA (ANN trained with Genetic algorithm). The results have been evaluated using various performance measuring metrics such as accuracy, precision, recall, and F-measure. Experimental results clearly show the superiority of the proposed NN-NSGA-II model with different features.

Implementation of Nearest Neighbor using HSV to Identify Skin Disease:

Nearest Neighbor is a classification method or algorithm that can be used to calculate the nearest value to the training data set. In the process, Euclidean Distance was calculated to search a similar range of one record data testing to each record of the training data set. The method can be implemented in a classification model that uses HSV as data training and data testing. Based on the statement, the paper will try to create a classification model using Nearest Neighbor as a classification method, HSV as a technique to process images, an android application as a device that runs the application to identify skin disease.

Discriminative Feature Learning for Skin Disease Classification Using Deep Convolutional Neural Network:

They proposed a new framework by fine-tuning layers of ResNet152 and InceptionResNet-V2 models with a triplet loss function. In the proposed framework, first, learning the embedding from input images into Euclidean space using deep CNN ResNet152 and InceptionResNet-V2 models is done. Second, we compute L-2 distance among corresponding images from Euclidean space to learn discriminative features of skin disease images by using the triplet loss function. Finally, classify the input images using these L-2 distances. Experiment results and their analysis shows the effectiveness of the proposed framework which achieves better accuracy than many existing works in skin disease tasks.

Diagnosis of skin diseases using Convolutional Neural Networks:

Here the proposed paper provides an approach to detect various kinds of these diseases. The user gives input of the skin disease image, which then the system processes, does feature extraction using CNN algorithm, and uses softmax image classifier to diagnose diseases. If no disease is found, the system provides a negative result. Thus, in this paper, a novel dermoscopy detection and classification method based on a Convolutional Neural network (CNN) is proposed.

3. METHODOLOGY

3.1. Methodology

The dataset for this project is taken from Kaggle. In this project, different skin diseases like Acne, Eczema, Vascular Tumors, and Rosacea, etc. can be identified. Firstly, skin images were pre-processed to remove noise to get an enhanced image. Later feature extraction is done using convolutional neural networks (CNN), giving a diagnosis report as output. This method gives more accuracy and gives results faster than any normal method. The image will be sent into the model and will predict the type of skin disease.

3.2. Architecture of Proposed System

Input acquisition: Input is in the form of an image that is taken from the dataset.

Data pre-processing: Pre-processing helps to improve the quality of an image which makes analyzing better. The input image may not be the same size as required by the algorithm, so resizing the image is required.

Data augmentation: Data augmentation is used to increase the number of images in the training dataset through the existing image. Data augmentation is done on the train and test set like rescale, zoom range, rotation range, horizontal flip, etc. We can reduce overfitting in the future because of data augmentation. The augmented images are stored in the train generator and test generator.

Visualization: Few images from the train generator are displayed with its disease name above it.

Adding layers: A sequential model is taken and layers are added to it. The layers which are added to the model are convolutional layer, max-pooling layer, flatten, dense, etc. The activation function used is the ReLU function and for the last dense layer, we used the softmax function.

Training the model: The model is then trained using the model.fit function which takes the train data, test data, epochs, and batch size as its parameters.

Prediction: An image is sent to the model predict function which gives a vector of size as the number of classes, the index of the array which has the maximum value will help us in predicting the disease.

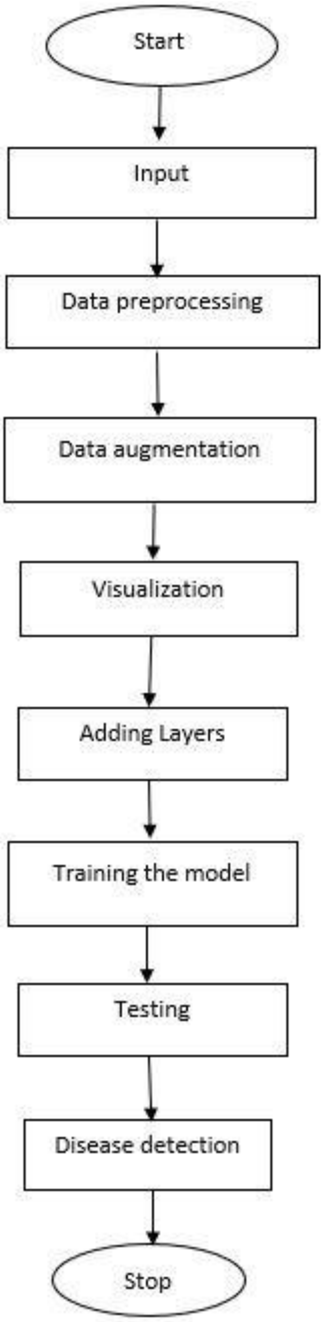


Fig.3.1 Flowchart of Skin disease detection model

There are 5 different disease images and these images are taken as input. The training and test images are pre-processed using OpenCV and are stored in a list. Data augmentation is done on the train and test set like rescale, zoom range, rotation range, horizontal flip, etc. later a few images from the train generator are plotted. A sequential model is taken and CNN layers are added to it then training the data is done. It classifies the images into different diseases and predicts the disease of the given image.

3.3 SOFTWARE & HARDWARE REQUIREMENTS

Software:

Python IDE

Google colab

Hardware:

Processors: Intel Atom® processor or Intel® Core™ i3 processor.

Input device (mouse / trackpad) to select options

32 – 64 bit processor

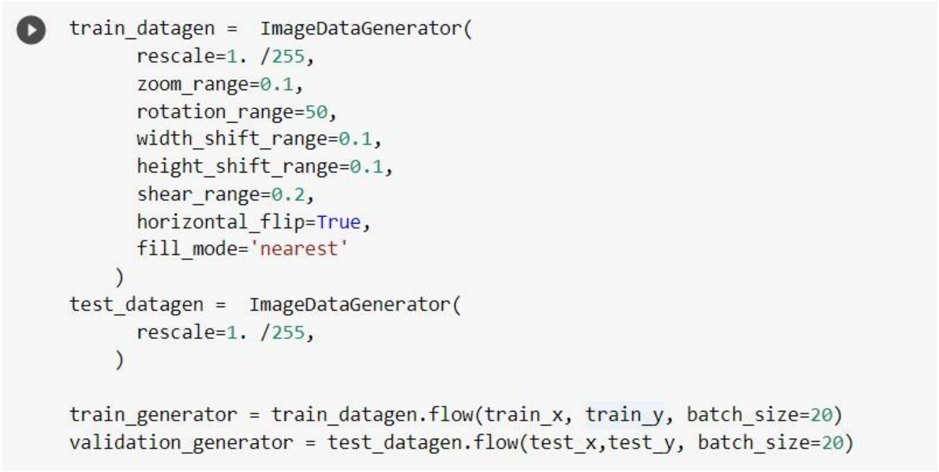
Sufficient RAM to run the program

\

4. IMPLEMENTATION OF PROJECT

4.1. Implementation

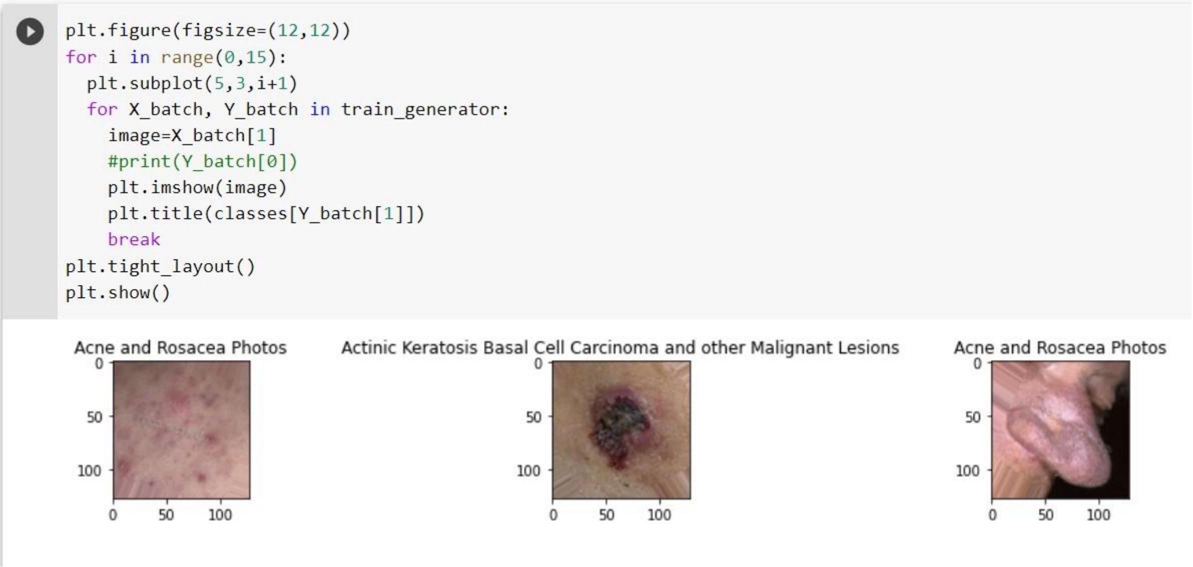
Firstly, all the required libraries are imported then the images are taken from the dataset then the images are pre-processed where they are converted into standard size i.e., 128X128. After pre-processing the images are augmented using an image data generator function where rescale, zoom range, rotation range, width shift range, height shift range, horizontal flip, etc. are used as parameters. The data generated is then stored in the train generator and validation generator. Few images from the train generator are shown using matplotlib. Now a sequential model is created and layers are added to it, the first layer is the convolution layer, and then the max-pooling layer, etc are added to it. Then we add a flatten layer which converts the input to a one-dimensional vector. Later a dense layer is added where the output of this layer is the value specified in the dense layer which has a relu as its activation function. A dropout layer is also added which reduces the overfitting. At last, a dense layer whose value is the number of classes of the training set is given and a softmax classifier is used where it assigns decimal probabilities to each class. We compile the model that uses Adam as its optimizer and has sparse categorical cross-entropy as a loss function. For the given test and training set the model is fit which has the batch size as 128 and the number of epochs used are 40. After this, the accuracy is plotted and the confusion matrix is drawn. The test images are used to predict the type of skin disease.



```
train_datagen =  ImageDataGenerator(
      rescale=1. /255,
      zoom_range=0.1,
      rotation_range=50,
      width_shift_range=0.1,
      height_shift_range=0.1,
      shear_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest'
    )
test_datagen =  ImageDataGenerator(
      rescale=1. /255,
    )

train_generator = train_datagen.flow(train_x, train_y, batch_size=20)
validation_generator = test_datagen.flow(test_x,test_y, batch_size=20)
```

Fig.4.1 Data augmentation

Data augmentation is used to increase the number of images in the training dataset through the existing image. Data augmentation is done on the train and test set like rescale, zoom range, rotation range, horizontal flip, etc. We can reduce overfitting in the future because of data augmentation. The augmented images are stored in the train generator and test generator.

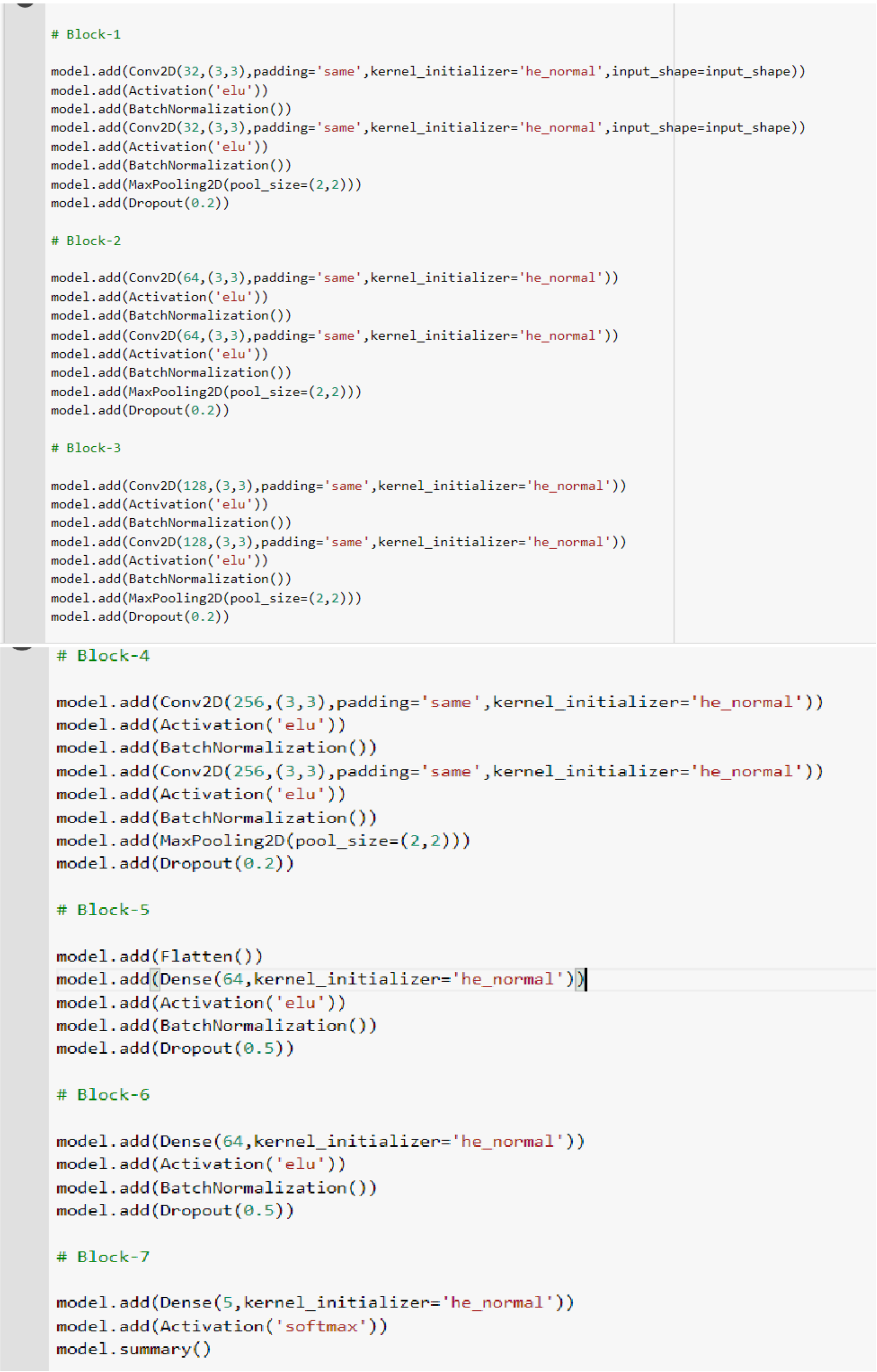


```
plt.figure(figsize=(12,12))
for i in range(0,15):
  plt.subplot(5,3,i+1)
  for X_batch, Y_batch in train_generator:
    image=X_batch[1]
    #print(Y_batch[0])
    plt.imshow(image)
    plt.title(classes[Y_batch[1]])
    break
plt.tight_layout()
plt.show()
```

Acne and Rosacea Photos
Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions
Acne and Rosacea Photos
0
50
100
0
50
100

Fig.4.2 Visualization

Few images from the train generator are displayed using matplotlib. Its disease name is shown above each plot.

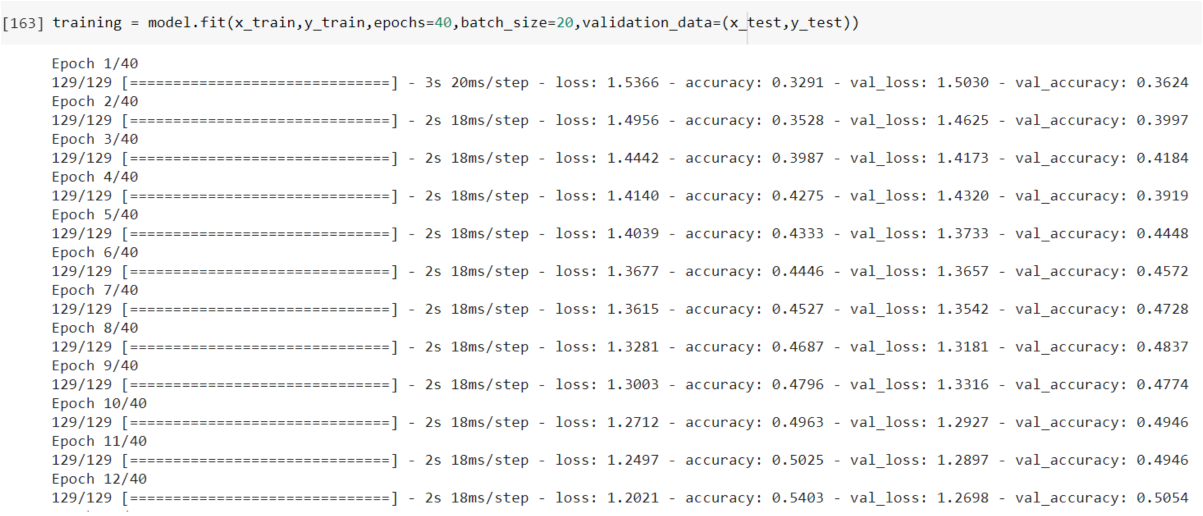


```
# Block-1

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',input_shape=input_shape))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',input_shape=input_shape))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-2

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-3

model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```

```
# Block-4

model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-5

model.add(Flatten())
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-6

model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-7

model.add(Dense(5,kernel_initializer='he_normal'))
model.add(Activation('softmax'))
model.summary()
```

Fig.4.3 Adding Layers

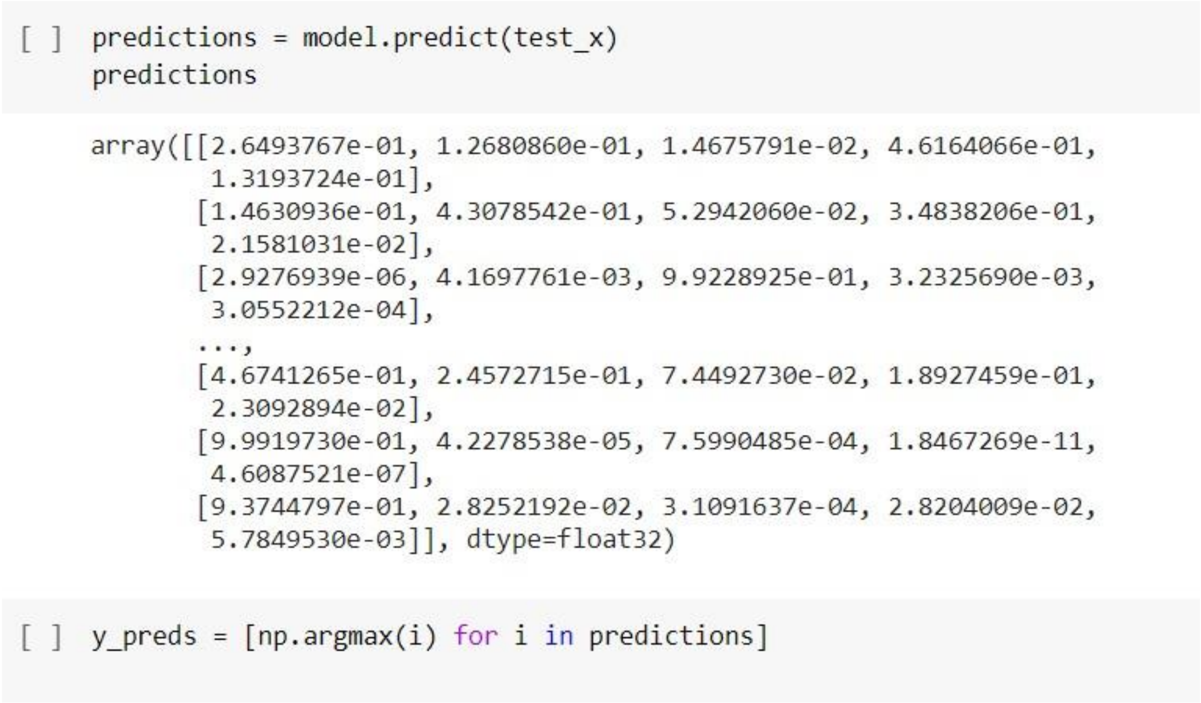
Adding layers: A sequential model is taken and layers are added to it. The layers which are added to the model are convolutional layer, max-pooling layer, flatten, dense, etc. The activation function used is the ReLU function and for the last dense layer, we used the softmax function.



```
[163] training = model.fit(x_train,y_train,epochs=40,batch_size=20,validation_data=(x_test,y_test))

      Epoch 1/40
      129/129 [==============================] - 3s 20ms/step - loss: 1.5366 - accuracy: 0.3291 - val_loss: 1.5030 - val_accuracy: 0.3624
      Epoch 2/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.4956 - accuracy: 0.3528 - val_loss: 1.4625 - val_accuracy: 0.3997
      Epoch 3/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.4442 - accuracy: 0.3987 - val_loss: 1.4173 - val_accuracy: 0.4184
      Epoch 4/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.4140 - accuracy: 0.4275 - val_loss: 1.4320 - val_accuracy: 0.3919
      Epoch 5/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.4039 - accuracy: 0.4333 - val_loss: 1.3733 - val_accuracy: 0.4448
      Epoch 6/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.3677 - accuracy: 0.4446 - val_loss: 1.3657 - val_accuracy: 0.4572
      Epoch 7/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.3615 - accuracy: 0.4527 - val_loss: 1.3542 - val_accuracy: 0.4728
      Epoch 8/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.3281 - accuracy: 0.4687 - val_loss: 1.3181 - val_accuracy: 0.4837
      Epoch 9/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.3003 - accuracy: 0.4796 - val_loss: 1.3316 - val_accuracy: 0.4774
      Epoch 10/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.2712 - accuracy: 0.4963 - val_loss: 1.2927 - val_accuracy: 0.4946
      Epoch 11/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.2497 - accuracy: 0.5025 - val_loss: 1.2897 - val_accuracy: 0.4946
      Epoch 12/40
      129/129 [==============================] - 2s 18ms/step - loss: 1.2021 - accuracy: 0.5403 - val_loss: 1.2698 - val_accuracy: 0.5054
```

Fig.4.4 Fitting the dataset

Training the model: The model is then trained using the model.fit function which takes the train data, test data, epochs, and batch size as its parameters.



```
[ ]  predictions = model.predict(test_x)
     predictions
```

```
array([[2.6493767e-01, 1.2680860e-01, 1.4675791e-02, 4.6164066e-01,
        1.3193724e-01],
       [1.4630936e-01, 4.3078542e-01, 5.2942060e-02, 3.4838206e-01,
        2.1581031e-02],
       [2.9276939e-06, 4.1697761e-03, 9.9228925e-01, 3.2325690e-03,
        3.0552212e-04],
       ...,
       [4.6741265e-01, 2.4572715e-01, 7.4492730e-02, 1.8927459e-01,
        2.3092894e-02],
       [9.9919730e-01, 4.2278538e-05, 7.5990485e-04, 1.8467269e-11,
        4.6087521e-07],
       [9.3744797e-01, 2.8252192e-02, 3.1091637e-04, 2.8204009e-02,
        5.7849530e-03]], dtype=float32)
```

```
[ ]  y_preds = [np.argmax(i) for i in predictions]
```

Fig.4.5 Predictions

Prediction: An image is sent to the model predict function which gives a vector of size as the number of classes, the index of the array which has the maximum value will help us in predicting the disease.

5. TESTING AND RESULTS

Graphs:

Training v/s Validation:

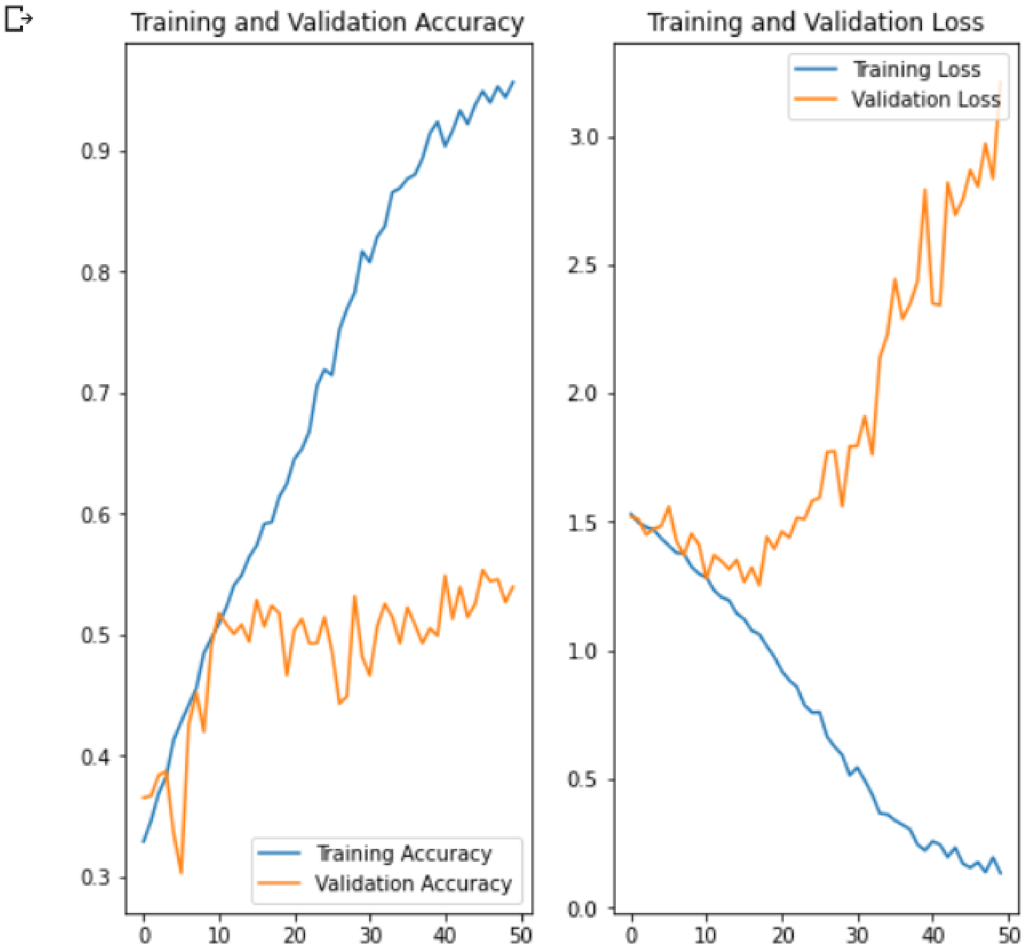
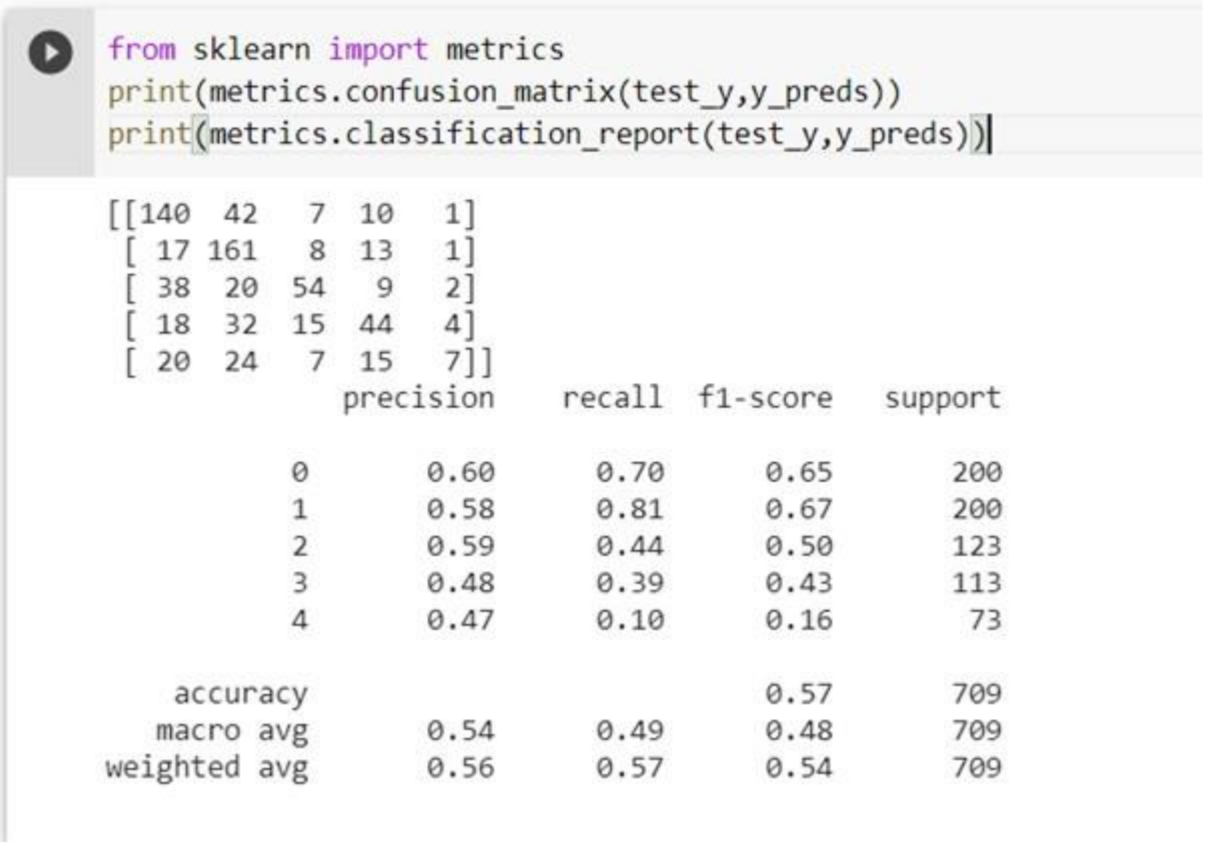


Fig.5.1 Training v/s validation

We plot a graph to get maximum accuracy that is achieved by the model in training and validation sets and to minimize the loss. The above plot shows the increase in training and validation accuracy. Similarly, the training and validation loss decreases.

Confusion matrix:

The confusion matrix is a table that describes the performance of the classification model. It calculates precision, recall, f1-score, support, etc. by analyzing the true positives, true negatives, false positives, false negatives.



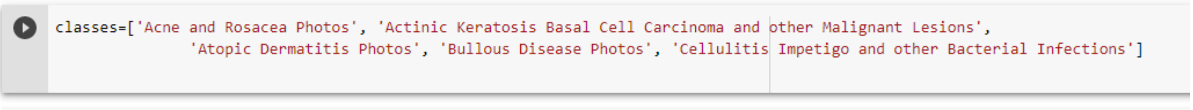
```
from sklearn import metrics
print(metrics.confusion_matrix(test_y,y_preds))
print(metrics.classification_report(test_y,y_preds))
```

```
[[140  42   7  10   1]
 [ 17 161   8  13   1]
 [ 38  20  54   9   2]
 [ 18  32  15  44   4]
 [ 20  24   7  15   7]]
              precision    recall  f1-score   support

           0       0.60      0.70      0.65       200
           1       0.58      0.81      0.67       200
           2       0.59      0.44      0.50       123
           3       0.48      0.39      0.43       113
           4       0.47      0.10      0.16        73

    accuracy                           0.57       709
   macro avg       0.54      0.49      0.48       709
weighted avg       0.56      0.57      0.54       709
```

Fig.5.2 Confusion matrix

From the above classification report, we can say that the accuracy is reported around 60 percent.

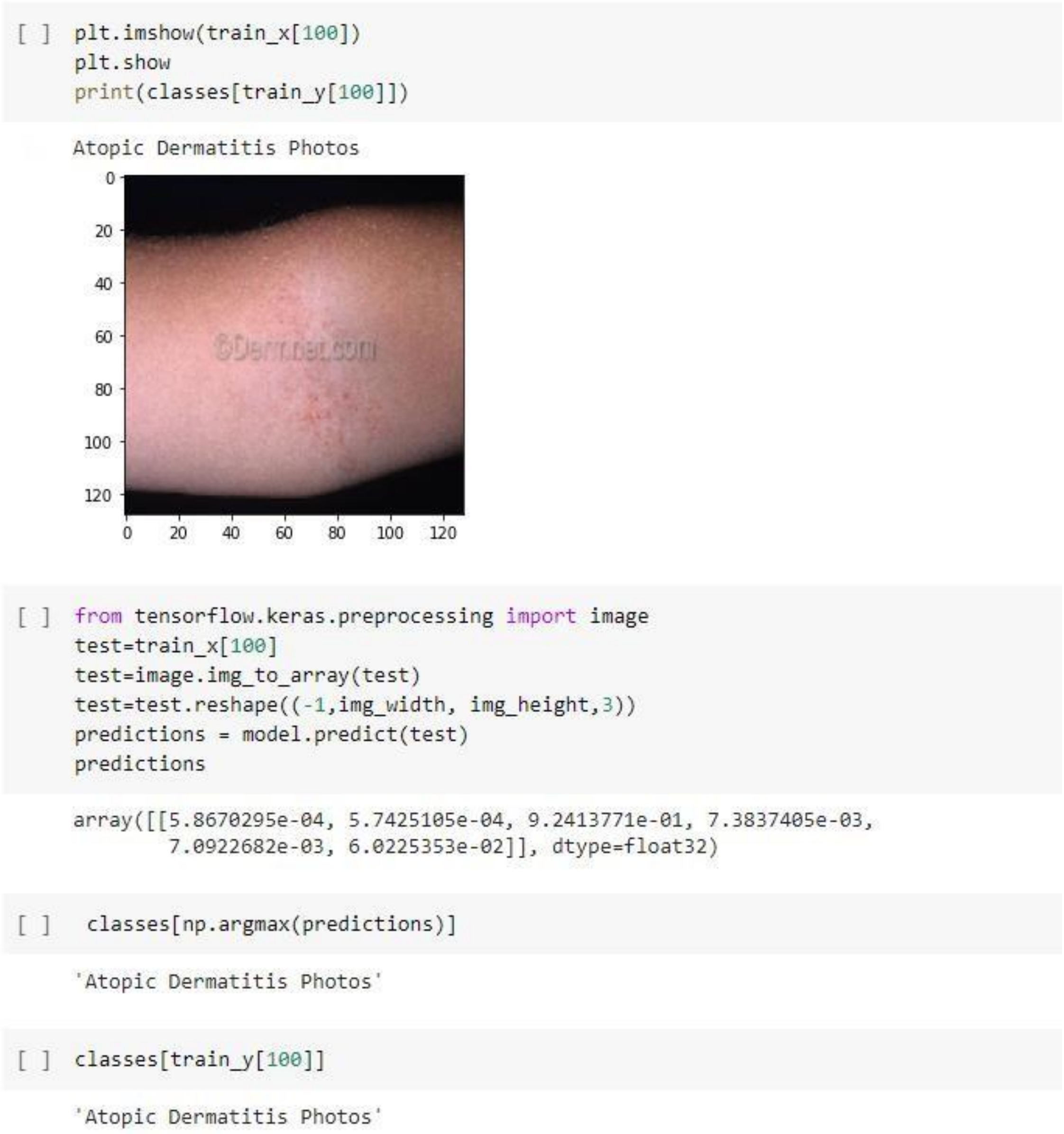
Output:



```
classes=['Acne and Rosacea Photos', 'Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions',
         'Atopic Dermatitis Photos', 'Bullous Disease Photos', 'Cellulitis Impetigo and other Bacterial Infections']
```

Fig.5.3 Classes

The classes present in the training and test dataset are stored in a list named classes. There are five classes which are Acne and Rosacea Photos, Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions, Atopic Dermatitis Photos, Bullous Disease Photos, Cellulitis Impetigo, and Other Bacterial Infections.



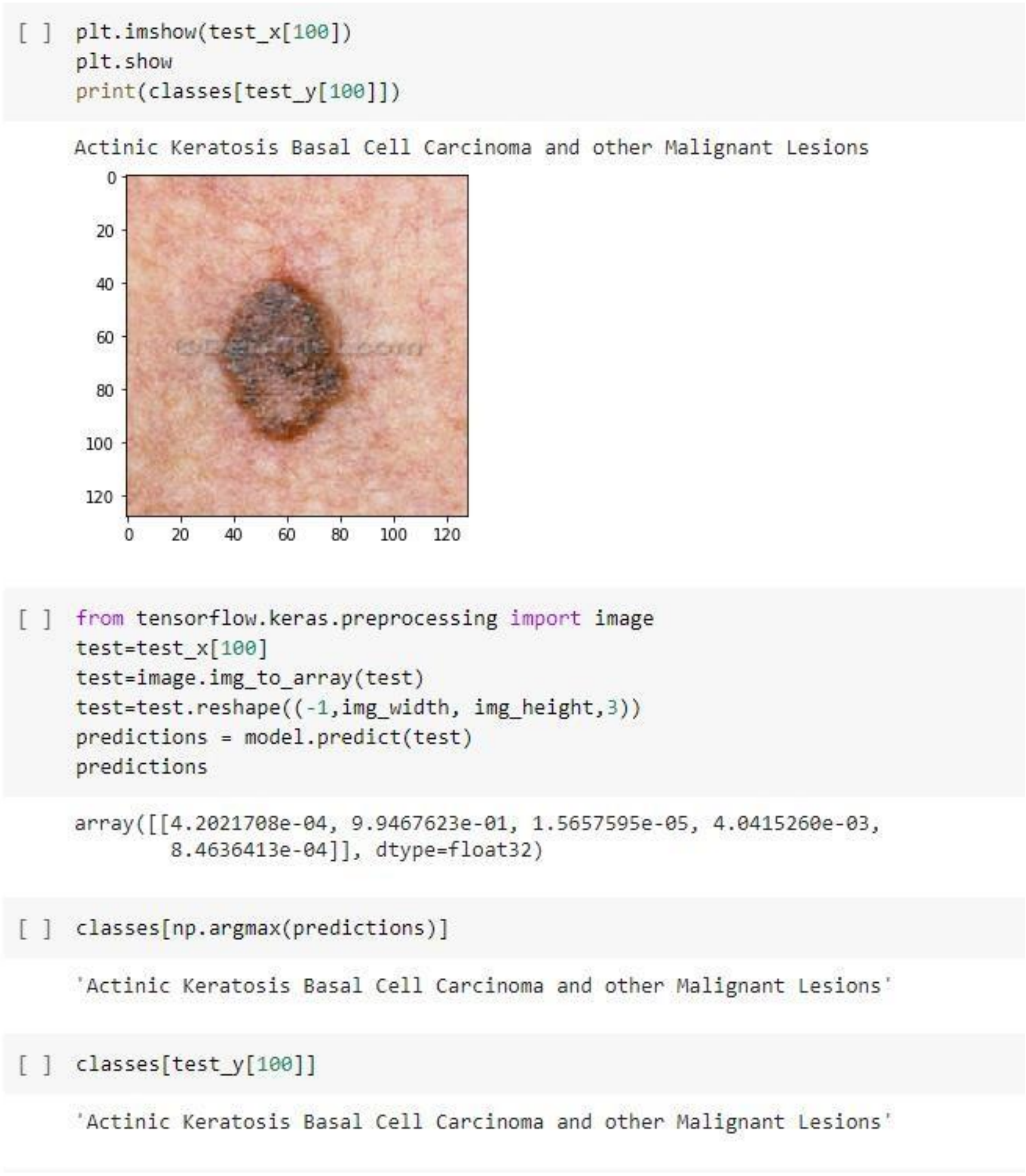
```
[ ]  plt.imshow(train_x[100])
     plt.show
     print(classes[train_y[100]])

     Atopic Dermatitis Photos
```

```
[ ]  from tensorflow.keras.preprocessing import image
     test=train_x[100]
     test=image.img_to_array(test)
     test=test.reshape((-1,img_width, img_height,3))
     predictions = model.predict(test)
     predictions

     array([[5.8670295e-04, 5.7425105e-04, 9.2413771e-01, 7.3837405e-03,
             7.0922682e-03, 6.0225353e-02]], dtype=float32)
```

```
[ ]   classes[np.argmax(predictions)]

     'Atopic Dermatitis Photos'
```

```
[ ]  classes[train_y[100]]

     'Atopic Dermatitis Photos'
```

Fig.5.4 Output 1



```
[ ]  plt.imshow(test_x[100])
     plt.show
     print(classes[test_y[100]])
```

```
Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions
```

```
[ ]  from tensorflow.keras.preprocessing import image
     test=test_x[100]
     test=image.img_to_array(test)
     test=test.reshape((-1,img_width, img_height,3))
     predictions = model.predict(test)
     predictions
```

```
array([[4.2021708e-04, 9.9467623e-01, 1.5657595e-05, 4.0415260e-03,
        8.4636413e-04]], dtype=float32)
```

```
[ ]  classes[np.argmax(predictions)]
```

```
'Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions'
```

```
[ ]  classes[test_y[100]]
```

```
'Actinic Keratosis Basal Cell Carcinoma and other Malignant Lesions'
```

Fig.5.5 Output 2

A random image is taken from the dataset. The disease of that image is predicted using the model we created. model.predict function gives an array whose size is the number of classes in the dataset. The index of the array which has the maximum value is sent to the classes list which gives us the disease name.

7. CONCLUSION & FUTURE SCOPE

This project helps people to diagnose the initial stage of skin disease and seek their symptoms before getting a serious disease. There are huge similarities between these diseases if not detected properly it may lead to various health hazards. A model has been built where the model has been trained to diagnose a skin by taking its complexities and views of skin tone, color, and presence of hair. A Convolutional Neural Network has been used for image pre-processing. Using deep learning techniques, the type of skin disease has been made possible by providing a screening test to diagnose the initial stage of the disease through the created model. To improve the ability to analyze and process a large number of images, the traditional convolution neural network is improved.

The accuracy detected by this model is less which can be further improved by increasing images in the dataset. To get an efficient output the number of classes i.e. number of diseases can be increased in the dataset. Various other methods can be used to increase accuracy.

Dermatologists use laser treatment for the treatment of many dermatologic conditions and detect their disease through a biopsy which is nothing but a sample of tissue taken from the body to examine it closely. A hardware device can be built where a model can be deployed in the device which not only detects the disease but also does a biopsy to closely examine the skin disease. Common skin disorders can easily be treated by their smartphones without consulting any doctor, by creating an app that has a voice bot that can be trained by a highly skilled dermatologist. By providing this application any serious skin disease can be treated more optimally and efficiently.

BIBLIOGRAPHY

1. M. N. Islam, J. Gallardo-Alvarado, M. Abu, N. A. Salman, S. P. Rengan and S. Said, "Skin disease recognition using texture analysis," *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*, 2017, pp. 144-148, doi: 10.1109/ICSGRC.2017.8070584.

2. Verma AK, Pal S, Kumar S. Classification of Skin Disease using Ensemble Data Mining Techniques. *Asian Pac J Cancer Prev*. 2019;20(6):1887-1894. Published 2019 Jun 1. doi:10.31557/APJCP.2019.20.6.1887

3. Z. Wu et al., "Studies on Different CNN Algorithms for Face Skin Disease Classification Based on Clinical Images," in IEEE Access, vol. 7, pp. 66505-66511, 2019, doi: 10.1109/ACCESS.2019.2918221.

4. S. Chakraborty et al., "Detection of skin disease using metaheuristic supported artificial neural networks," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), 2017, pp. 224-229, doi: 10.1109/IEMECON.2017.8079594.

5. Y A Gerhana *et al., "*Implementation of Nearest Neighbor using HSV to Identify Skin Disease*",* 2018 *IOP Conf. Ser.: Mater. Sci. Eng.* 288 012153

6. B. Ahmad, M. Usama, C. Huang, K. Hwang, M. S. Hossain, and G. Muhammad, "Discriminative Feature Learning for Skin Disease Classification Using Deep Convolutional Neural Network," in IEEE Access, vol. 8, pp. 39025-39033, 2020, doi: 10.1109/ACCESS.2020.2975198.

7. J. Rathod, V. Waghmode, A. Sodha, and P. Bhavathankar, "Diagnosis of skin diseases using Convolutional Neural Networks," 2018 Second International Conference on Electronics, Communication, and Aerospace Technology (ICECA), 2018, pp. 1048-1051, doi: 10.1109/ICECA.2018.8474593.