SKIN DISEASE DETECTION USING DEEP LEARNING ALGORITHM

Project work submitted to Avinashilingam Institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

S. Aishwarya(20PIT001)

Under the Guidance of

Dr.(Mrs).F.Paulin M.C.A, M.phil.,Ph.D

Assistant Professor

Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

MAY-2022

DECLARATION

DECLARATION

I hereby declare that the project entitled "**SKIN DISEASE DETECTION USING DEEP LEARNING ALGORITHM"** is a record of the original work done by S.Aishwarya(20PIT001) under the guidance of **Dr**.(**Mrs**).**F.Paulin M.C.A, M.phil.,Ph.D.,** Assistant Professor, Department of Information Technology, School of Physical Sciences and Computational Sciences**,** Avinashilingam Institute for Home Science and Higher Education for Women in the partial fulfilment for the award of the degree of Master of Science in Information Technology, and this project work has not formed the basis for any Degree/Diploma/Associates.

Place: Coimbatore

Date:

Signature of the Candidate

Countersigned By

Dr.(Mrs).F.Paulin ,M.C.A,M.phil.,Ph.D.,

Assistant Professor,

Department of Information Technology,

School of Physical Sciences and Computational Sciences**.**

CERTIFICATE

CERTIFICATE

Net Tel Solutions India Pvt Ltd

Excellence in Service

10.05.2022
Coimbatore

TO WHOMSOEVER IT MAY CONCERN

This is to confirm that **Ms. Aishwarya S (Reg No:20PIT001)** final year student of **M.Sc (Information Technology)** from **"Avinashilingam Institute for Home Science and Higher Education for Women"** Coimbatore, has successfully completed his Project work on **"SKIN DISEASE DETECTION USING DEEP LEARNING ALGORITHM"** in our esteemed organization from January 2022 to May 2022.

For Net Tel Solution India Pvt Ltd

Authorized Signatory

#203, Ram Nagar, Nehru Street, Coimbatore - 641 009
www.nettelsolutions.in
info@nettelsolutions.in

This is to certify that this project work entitled "**SKIN DISEASE DETECTION USING DEEP LEARNING ALGORITHM"** done by S.Aishwarya (20PIT001) has been submitted to Avinashilingam Institute for Home science and Higher Education for Women, Coimbatore-43 in partial fulfillment of the requirement for the award of the degree of **MASTER OF SCIENCE ININFORMATION TECHNOLOGY.** This Project has not found the basis for the award of any Degree/ Associate/ fellowship or similar title to any Candidate of any University. Certified as a bonafied record of the work submitted for the Viva voce held on_____.

Signature of the HOD

Signature of the Guide

Signature of External Examiner

ACKNOWLEGEMENT

ACKNOWLEDGEMENT

Take this opportunity to express our deep sense of gratitude to our **Chancellor Dr. S. P. Thyagarajan,** Avinashilingam Institute of Home Science and Higher Education for women, Coimbatore for his support and encouragement during the course of our project work.

heartily thank **Dr.(Mrs.)Bharathi Harishanker** Ph.D.,FRSA **Vice-chancellor** for extending all resources that facilitated the conduct of the present work.

Wish to extend our sincere thanks to **Dr.(Mrs.)S.Kowsalya**M.Sc.,M.Phil.,Ph.D.**Registrar** for helping and sustaining me in all possible means to come out with the project.

wish to place on record our deep sense of gratitude to **Dr.(Mrs.) G.Padmavathi** M.Sc., M.Phil., Ph.D.,**Dean**, School of Physical Sciences and Computational Sciences, for providing all the facilities to complete the project.

Our heartiest thanks to **Dr.(Mrs.)D.Shanmugapriya** M.Sc., M.Phil., Ph.D., SET **Head of Department** of Information Technology for the valuable guidance and encouragement during the course of our project.

Take this opportunity to express our profound gratitude and deep regards to our **Guide Dr.(Mrs.)F.Paulin M.C.A,M.Phil.,Ph.D.** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life on which are about to embark.

would like to thank all the faculty member laboratory staff of the Department of Information Technology, all our friends and well-wishers who had either directly or indirectly helped us to finish this Project successfully. I thank our parents for their encouragement and moral support. Above all the thank god almighty whose grace was sufficient at all times.

ABSTRACT

ABSTRACT

Dermatological problems are one of the most common illnesses worldwide. Dermatology is a tough and expensive field to diagnose due to its intricacy. The widespread use of smart phones in underdeveloped countries such as India has opened up new opportunities for low-cost disease diagnostics. This method is simple, quick, and does not require expensive technology other than a camera and a computer. The methodology is based on colour image inputs. Then, using Convolutional Neural Architecture, resized the image to extract highlights. Smartphone camera technology can be used to diagnose diseases by utilizing the device's image processing skills. The proposed system is focused with the development of an application that aids in the diagnosis of skin diseases. Machine learning and image processing are the two components of the system. The image processing section is concerned with applying filters to images in order to remove noise and make them homogenous. As a result, Deep Learning Techniques are being used to generate alternatives that do not require large expenditures in hardware or infrastructure. Deep Learning Techniques are now being successfully utilised to predict several types of cancers in their very early stages. Finally, the client sees the results, which include the type of illness, its spread, and its potential consequences.

CONTENTS

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 Overview of the Project

Because of the increased occurrence of skin cancers, dermatologists will find computer-based diagnostic techniques to be quite beneficial in diagnosing skin illnesses. Visual evidence, various tests, and, in some cases, dermoscopic examination, biopsy, and histological research are used to diagnose skin abnormalities. Automatic classification of dermoscopic images is difficult due to changes in the visual appearance of lesions. Deep learning techniques such as CNN out performed traditional methods (Convolutional Neural Network). A combination of diagnostic laboratory tests are used to identify the proper disease in the skin disease detection process. These skin abnormalities are highly contagious and should be treated as soon as possible to prevent spread. The majority of these skin anomalies are deadly, especially if not addressed early on. Humans have a tendency to presume that the majority of skin anomalies aren't as serious as they appear, therefore they treat them accordingly. However, if these solutions are not used right away to address that specific skin problem, it becomes even more harmful.

The current diagnosis technique involves lengthy laboratory procedures, but this approach suggests a system that uses computer vision to help individuals predict skin illness. The kind and stage of skin disease are unknown to most people. Some skin diseases show symptoms months later, causing the condition to worsen and progress. With the advancement of medical technology, lasers and photonics have made it possible to identify skin illnesses considerably more rapidly, efficiently, and correctly. As a result, develop a deep learning-based image processing system for diagnosing skin disorders. This method uses image scanning to detect the type of disease by taking a digital image of the disease affecting the skin area as input.

1.1.1 Problem Definition

Since the skin is one of the largest organs in the human body, early diagnosis of skin illnesses is critical. As people's lifestyles have evolved, disease prevention and early detection are more important than ever. New diseases have emerged as a result of today's lifestyle. Chronic diseases may develop if sufficient care and safety are not provided. There is a need for a system that can diagnose skin illnesses without the use of extensive tests.

1.1.2 Objective

The purpose of the project is to develop a Convolutional Neural Network to diagnose various skin diseases such as acne, psoriasis, and warts that overcomes the drawbacks of existing methods and provides better results and accuracy, as well as to create an efficient and dependable system for dermatological disease detection that can be used as a reliable real-time teaching tool for dermatology students.

1.1.3 Scope

The field of medicine using cutting-edge technologies has been rapidly expanding. Disease detection via modern technologies and software can be viewed as an enhancement to existing medical technology applications. Neural networks can be used to simulate the human brain. The suggested method uses a Convolutional Neural Network to diagnose skin diseases to some extent. This methodology allows for the diagnosis and classification of user input photographs within the data set provided during training. The scope of skin disease detection is found in the broader applications of CNN, which enable multiple human-machine interactions for improved user convenience.

1.2 Machine Learning

Artificial Intelligence has a branch called Machine Learning (AI). The goal of machine learning is to comprehend the structure of data and fit that data into models that people can comprehend and use. Machine learning is a subfield of computer science that is distinct from usual computational methods. Algorithms are sets of clearly designed instructions used by computers to calculate or solve problems in traditional computing. Machine learning techniques, on the other hand, allow computers to train on data inputs and then integrate statistical analysis to produce results that are within a certain range. Machine learning makes it easier for computers to develop models from sample data and automate decision-making processes based on data inputs as an outcome.

1.2.1 Deep Learning

Machine learning can be regarded as a branch of deep learning. It is a field concerned with computer algorithms that learn and improve on their own. Deep learning is a machine learning and artificial intelligence (AI) technique that resembles how humans acquire knowledge. Neural networks were previously limited in complexity due to computational power constraints. Deep learning is highly useful for data scientists who are responsible with gathering, analyzing, and interpreting massive amounts of data; Deep learning can be regarded of as a means to automate predictive analytics at its most basic level. Deep learning algorithms are built in a hierarchical system of increasing complexity and abstraction, unlike typical machine learning algorithms, which are linear. Image categorization, language translation, and speech recognition have all benefited from deep learning. Deep Neural Networks (DNNs) are networks that make sense of images, music, and text by performing complicated operations such as representation and abstraction at each layer.

1.2.2 Convolutional Neural Network

CNN (Convolutional Neural Network) is a subfield of deep learning that is mostly used to analyse visual data. A class of deep feed forward artificial neural networks known as CNNs (ANN). This Neural Network predicts future label assignments using the dataset that has already been provided to it for training. Image identification, picture categorization, image captioning, and object detection are some of the areas where CNNs are widely used.The fundamental advantages of CNNs are that they produce a dense network that efficiently executes prediction, identification, and other tasks. CNNs are the most popular topic in the huge field of deep learning, and this is mostly due to ConvNets. Large datasets are used with CNNs; in fact, it is thought that the larger the data, the higher the accuracy;

1.2.3 Mobilenet

For Classification Tasks and Mobility Vision, MobileNet is a CNN architecture model. Other models are available, but MobileNet stands out since it requires relatively minimum computational power to execute or apply transfer learning. This makes it ideal for mobile devices, embedded systems, and PCs with limited computing efficiency or no GPU, without affecting the accuracy of the results greatly..Depthwise separable classifiers, also known as

Depthwise Separable Convolution, are the foundation of MobileNet. The network structure is another feature that help to better performance. Finally, the width and resolution can be adjusted to balance latency and accuracy.

Mobilenet Architecture:

- MobileNets, which are based on a streamlined design that leverages depthwise separable convolutions to generate low weight deep neural networks, are proposed for mobile and embedded vision applications.
- They offer two simple global hyper-parameters that efficiently trade off latency and accuracy.

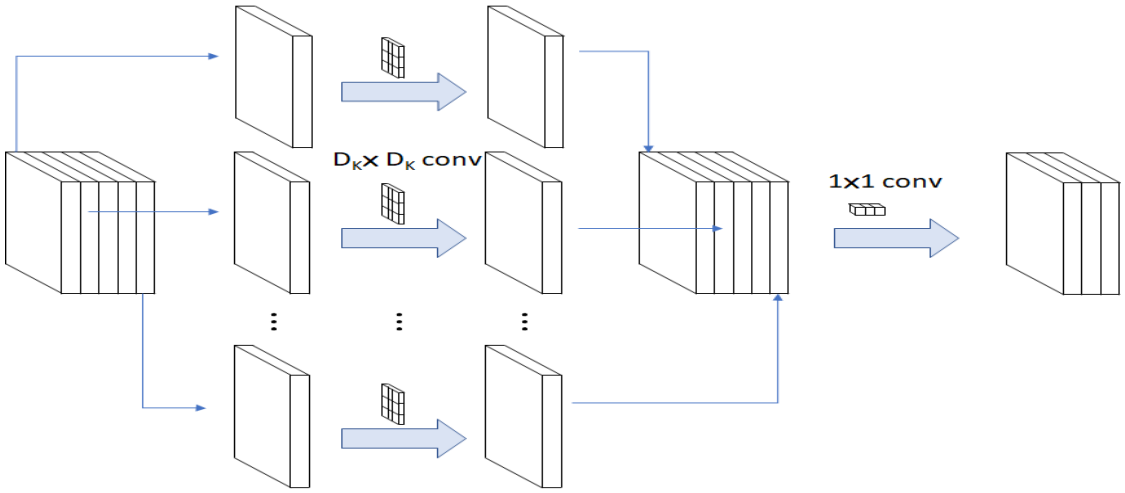


Fig.1Mobilenet Architecture

Depth-wise separable convolution:

The depth-wise separable convolution is made up of two layers: depth-wise and point-wise. In general, the first layer filters the input channels, while the second layer combines them to develop a new features.

Pointwise Convolution:

- Depthwise separable convolutions are a type of quantized convolution in which a standard convolution is categorical approach into a depthwise convolution and a 1*11*1 convolution known as a pointwise convolution.

- As a result, a pointwise convolution layer is created, which computes a linear combination of the depthwise convolution output using a 1*1 convolution.

Depthwise Convolution:

- The depthwise convolution in MobileNet applies a single filter to each input channel. After that, the pointwise convolution uses a 1*11*1 convolution to combine the depthwise convolution's outputs.
- In the case of depthwise convolution, an input feature map of dimension DF*DF and M number of kernels of channel size 1 are used, as shown in the graphic below.

1.3 Tools used

1.3.1 Python

Python is a powerful, statically typed programming language that is frequently used for general-purpose programming. It is used by software engineers, mathematicians, data analysts, scientists, network engineers, students, and accountants and is one of the world's fastest-growing programming languages.

Interpreted

The source file is processed at runtime by a processor, which reads the lines of code one by one and executes what is said. Python, like Perl and PHP, does not require to compile your programme before running it. As a result, you won't need to use a compiler. Python's byte code compilation is completely automated

High-level

Python uses high-level constructs that are then translated into a low-level language, the original code that runs on the central processor unit of a computer (CPU). A programmer will utilize a high-level language, and the produced code will be interpreted into a low-level language.

General-purpose

Python is a general-purpose programming language that may be used for almost anything. It can be used in practically any field for a wide range of tasks. Python is suitable for a wide range of jobs, from short-term software testing to long-term product development with roadmap planning.

Object-oriented

Object-oriented programming is a programming paradigm that promotes scripting and powerful code organization. The objects are then put together to form complicated computer programmes. Python also enables procedural programming in addition to object-oriented programming.

1.3.2 Jupyter notebook

Jupyter Notebook is an open source web tool for creating and sharing documents with live code, equations, visualization, and text. Project Jupyter Notebooks is a fork of the IPython project, which used to have its own IPython Notebook project. Jupyter gets its name from the three main programming languages it supports: Julia, Python, and R. Jupyter comes with the IPython kernel, which allows you to build Python programmes, although there are presently more than 100 alternative kernels available.

Notebook Documents

Notebook documents are Jupyter Notebook App documents that contain both computer code (e.g., python) and rich text components. Notebook papers are both human-readable and executable documents that contain the analysis description and results (figures, tables, and so on).

Jupyter Notebook App

The Jupyter Notebook App is a web-based server-client application for editing and running notebook papers. The Jupyter Notebook App can be run locally on a computer without internet access (as explained in this paper) or remotely on a server and accessible via the internet.

In addition to displaying, editing, and executing notebook papers, the Jupyter Notebook App includes a "Dashboard" (Notebook Dashboard), a "control panel" that displays local files and allows you to access or close notebook pages.

Kernel

A notebook kernel is a type of "computational engine" that runs the code in a Notebook document. Python code is executed via the ipython kernel, which is mentioned in this guide. Many other languages have kernels .The corresponding kernel is automatically launched when you open a Notebook document. The kernel executes the computation and produces the results when the notebook is run.

Notebook Dashboard

When run Jupyter Notebook App, the Notebook Dashboard is the first component you see. The Notebook Dashboard is primarily used to open notebook documents and control the kernels that are currently operating (visualize and shutdown).Other functions of the Notebook Dashboard are similar to those of a file manager, such as traversing directories and renaming/deleting files.

REVIEW OF LITERATURE

2. REVIEW OF LITERATURE

Leelavathy S, Jaichandran R, Shobana R, Vasudevan, Sreejith S Prasad and Nihad(2020), Discuss about the user's input photographs are processed to forecast the presence or absence of skin disease from a new input image. The suggested system accepts user input photographs by creating a user interface (UI) through which the user can submit skin images for analysis. The user interface is built using HTML. Color, texture, and shape features are recovered from the skin input photos using the proposed system's 2D Wavelet Transform method for feature extraction. For the experimental findings of this proposed system, they use a Pycharm-based Python script. It also has a basic user interface for the user's comfort. Combining computer vision and machine learning approaches, the suggested system can identify skin illness with promising results. Machine learning and image processing methods were effectively developed.

Divya Shree D V, Goggi Pragna, Prathibha L, Sushmetha G R, Mary M'Dsouza(2020) they introduced predicts multiple sorts of skin diseases by allowing users to upload skin photo photos to a system that can process and deliver a variety of skin disease absence or presence. Based on the retrieved picture features, this suggested system implemented a matlab tool to identify multiple types of skin disorders such as normal, melanoma, psoriasis, and dermo case. If an irregularity is identified, this system will send an alert to the surrounding medical team. This methodology has problems with segmentation, making the classification model less accurate.

Anushree D. Shetty, Deepthi P. Dsouza, S. G. Keerthi(2021) identified various sorts of skin diseases by allowing users to upload skin photo photos to a system that can process and deliver a variety of skin disease absence or presence. Based on the retrieved image attributes, this proposed system uses anaconda and an android tool to identify different types of skin disorders such as normal, melanoma, and psoriasis cases. The photos are then classified into three groups: train, test, and valid. The image is loaded by the user, and the application preprocesses it. The

skin disease is then detected by comparing it to the dataset, and the user is given the result. Using machine learning techniques, the proposed approach will identify skin disease with high accuracy.

Sruthi Chintalapudi, Vikas Prateek Mishra, Shubham Sharma, Sunil Kumar(2021) explored at image classification using deep learning in this paper. This research describes how to use several PC vision-based tactics (in-depth understanding) to naturally anticipate various types of skin illnesses. The user will be given access to a portal where they may upload photographs and have them examined for the most common skin disorders (including STDs) by a deep learning algorithm that has been trained with a large number of images. The method involves using pre-programmed image recognizers that have been altered to identify skin photos. It has been observed that by combining various features and deep learning, With an accuracy of 87 percent, the model would detect skin cancer development before all symptoms were identifiable to the patient.

Kumar, Vinayshekhar & Kumar, Sujay & Saboo, Varun. (2016). Detected Dermatological disease detection using image processing and machine learning. The technique is based on neural network-based texture analysis. Picture processing, image feature extraction, and data classification utilising an artificial neural network are all part of the system. The ANN can learn the patterns of symptoms associated with specific diseases and provide faster diagnosis. The grey level co-occurrence probabilities are mapped into multiple angle orientations using GLCM. Artificial neural networks (ANN) can be used as a knowledge base to improve diagnosis. The final stage is to train a feed-forward back-propagation neural network. The feed forward back-propagation neural network is trained with ten features, three of which are picture features and seven of which are user inputs. This method yields a 90% accuracy.

Evani Sai Krishna Karthik, Potu Teja, Vaggu Sridhar, and B. Priyanka(2021) discussed about Deep Learning the foundation for categorising skin lesions is extracting required distinctive information from an image. As a result, offer a standardised automatic image-based system that implements a computational Convolutional Neural Network to detect, classify, and present the type of disease. Used an 80:20 training to testing ratio dataset with photos of various skin conditions to construct the most accurate Deep Learning model providing Convolutional Neural Network and AlexNet CNN architecture, which can further improve accuracy. The project's major goal is to develop a reliable and highly accurate DL model that can be used as a valuable diagnostic tool in dermatology.

Syeda Fatima Aijaz, Saad Jawaid Khan, Fahad Azim, Choudhary Sobhan Shakeel, and Umer Hassan(2022) Proposed a deep learning-based application for categorization of five forms normal skin prediction. Preprocessing of the input photos included data augmentation, enhancement, and segmentation, followed by colour, texture, and shape feature extraction. Each image was rotated 15 degrees, the height and width were shifted, and the image was horizontally flipped. The histogram equalisation (HE) method is utilised to convert RGB photos into equivalent hue-saturation-value (HSV) image formats. Resize function is utilised to resize input photos to 64 x 64 resolution. The classification models were built with 80 percent of the photos using two deep learning algorithms: Convolutional Neural Network (CNN) and long short-term memory (LSTM). CNN and LSTM have reported accuracy of 84.2 percent and 72.3 percent, respectively. The accuracy generated by the two deep learning algorithms differed significantly in a paired sample T-test.

Srushti, Varshitha Makam, Sushmitha Shet, Swathi V A(2020) presented a technique Smartphone cameras can be used to diagnose problems by implementing the device's image processing techniques. The suggested approach involves the implementation of an application that improves in the detection of skin diseases. It detects diseases using image processing and machine learning technology. The system is divided into two sections: image processing and machine learning. The image processing section involves applying several filters to the images in order to remove noise and uniformize them. Pillow and OpenCV toolkits handle the image

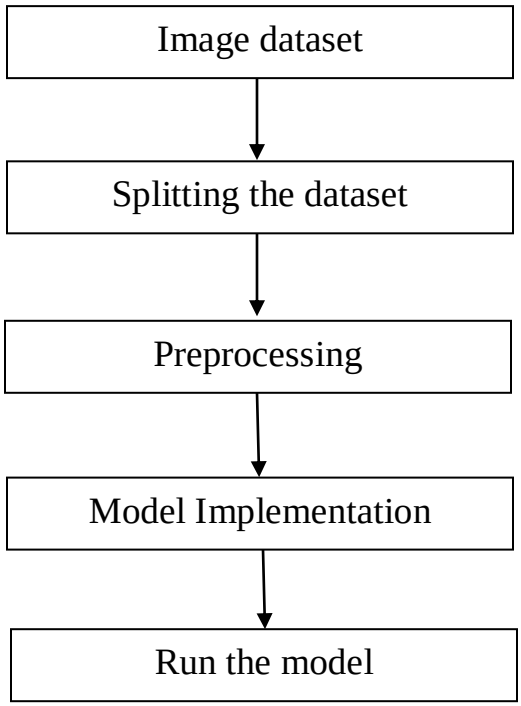
processing. Before processing, undesirable items must be removed from the image; otherwise, the output efficiency will suffer. Tkinter, a Python library, is used to create the user interface. A button on the user interface opens the file explorer, allowing you to choose the image to be reviewed. Despite the limited data set, the machine learning algorithm was able to correctly identify the condition.

Sourav Kumar Patnaik, Mansher Singh Sidhu, Yaagyanika Gehlot, Bhairvi Sharma and P Muthu(2018) introduced computer vision-based approaches to predict various types of skin disorders automatically. For feature extraction and training and testing, they used various types of Deep learning algorithms (Inception_ v3, MobileNet, Resnet, xception) and learning algorithms (ideally Random Forest or Logistic Regression).The system successfully predicts skin illness image recognition architectures: InceptionV3, InceptionResnetV2, and MobileNet with modifications for skin disease application. The feature extraction phase, the training phase, and the testing/validation phase are the three phases of the system. To educate itself with the numerous skin photos, the system use deep learning technology. The major goal of this system is to attain the highest level of skin disease prediction accuracy possible.

METHODOLOGY

3. METHODOLOGY

The Skin Disease Detection using Deep Learning algorithm comprises of the following steps:



3.1 Image Dataset

By gathering photos from various skin disease-related images, a dataset was collected. The datasets adopted in this research are open source and freely accessible over the internet. The dataset has been downloaded from www.dermnet.com. Retrieved photos of skin illnesses in ten different classes based on kinds of disease using the python file. Imported the data set and libraries. Installed TensorFlow then, followed by deployed AI and machine learning models. Installation of libraries are given below

OS module

Python's OS module involves activities for creating and removing directories (folders), retrieving their contents, altering and identifying the root directory, and more. To interface with the operating system and applications, one must first import the os module.

Warning

Warning message will be displayed using the warning() function from the 'warning' module. The warning module is essentially a subclass of the developed Python class Exception.

Shutil

The Python Shutil module provides numerous high-level manipulations on files and collections of files. It falls within the umbrella of Python's basic utility modules. This module aids in the automated processes of file and directory copying and removal.

Random

The Python Random module is a built-in module for randomly selecting input values in Python. These are pseudo-random numbers, which do not represent true randomness. This module can be used to do things like generate random numbers, output a random value for a list or string, and so on

Seaborn

Seaborn is a Python package that is mostly used to create statistical visuals. Seaborn is a Python information visualization package based on matplotlib that is deeply linked with pandas data structures. The core component of Seaborn is visual representation, which aids in data exploration and comprehension.

Matplotlib

Matplotlib is a data visualisation and graphical charting package for Python and its numerical extension NumPy that works across platforms. As a result, it acts as an open source replacement for MATLAB. Matplotlib's APIs (Application Programming Interfaces) can also be used to incorporate charts in GUI programmes.

3.2 Splitting the Dataset

The proposed methodology is an excellent strategy for analysing people's input in order to forecast skin illness. The project's goal is to split skin illness photographs into ten categories based on the type of condition. The database is divided into three sections: training and validation/testing. A training set is used to discover how to fit the parameters and is used to change the system's variable weights and errors in each training run. The validation/testing set fine-tunes the settings and is only used to evaluate the system's effectiveness and efficiency. The division mode is set to 80% for data training and 20% for data validation and testing in this method.

3.3 Preprocessing

To achieve better performance in a skin disease detection system, some main challenges must be overcome. Creating a directory and standardized image dimensions are two examples. An input image is either increased or decreased in size to overcome the problem of varying image sizes in the database. By modifying picture sizes, all images will have the same number of characteristics. Furthermore, shrinking the image cuts down on processing time, improving system speed. The original image has been downsized to a new size of 256*256 pixels.

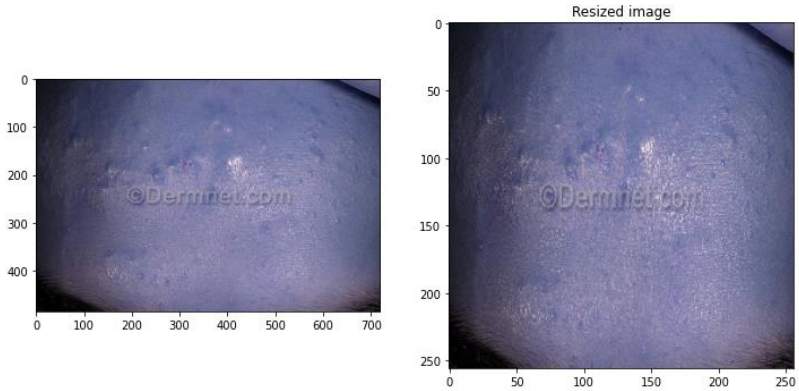


Fig.2 resized image

3.4 Build Model

Convolutional Neural Networks are used in the proposed methodology. The user enters the skin disease, which the system processes, extracts features using the CNN method, then diagnoses the ailment using the soft-max classifier. If no disease is discovered, the system returns a negative result, indicating that the disease was not found in the dataset. The architecture is divided into two sections: extraction and categorization. The image will be enhanced by the feature extraction device, which will remove noise and unnecessary skin parts. The photos are first pre-processed and then converted to a standard size. After that image is transmitted into the network's first layer as an input. It is then subjected to a Convolutional Neural Network until high-level properties like as colour, shape, and texture are acquired. The images obtained during image acquisition comprise up the testing dataset. The classifier determines the skin disease category. The Softmax classifier is the network's final layer, which calculates the actual probability of each label.

Use MobileNet in this project, which comes with 92 layers by default, and specify Dense and Dropout. Also mentioned are the class sensitivities. Pre-trained weights are used to start the model. This project is a deep learning Convolutional model with four classes that accepts images in 256 x 256 dimensions and three channels (RGB). This model's output prediction is done using global average pooling (GAP) layers to reduce overfitting by reducing the total number of parameters in the model and a dense layer activated with softmax to detect partial matching to another class by evaluating the values given by each node in the layer. Currently, the disease class is determined by indexing the largest of the values.

Algorithm for Convolutional Neural Network

`Input` : $a^{[l-1]}$ with size $(n_H^{[l-1]}, n_W^{[l-1]}, n_C^{[l-1]})$, $a^{[0]}$ being the image in the input
`Padding` : $p^{[l]}$, `stride` : $s^{[l]}$
`Number of filters` : $n_C^{[l]}$ where each $K^{(n)}$ has the dimension: $(f^{[l]}, f^{[l]}, n_C^{[l-1]})$
`Bias` of the n^{th} convolution: $b_n^{[l]}$
`Activation function` : $\psi^{[l]}$
`Output` : $a^{[l]}$ with size $(n_H^{[l]}, n_W^{[l]}, n_C^{[l]})$

Batch normalization

Batch normalization standardizes the inputs to a layer for each mini-batch while training very deep neural networks. This stabilises the learning process and significantly reduces the number of training epochs needed to create deep networks.

Rectified Linear Unit (ReLU)

The Rectified Linear Unit (ReLU) transform function only activates a node if the input reaches a specified limit; if the input is below zero, the output is zero; however, if the input above a certain threshold, the output is linear. The primary goal is to eliminate all negative values from the convolution and maxpooling processes. All positive values remain unchanged, but all negative values are reset to zero.

Dense layer

In a neural network, a dense layer is basically a regular layer of neurons. Each neuron is tightly linked because it receives input from all the activations in the previous layer. A weight matrix W, a bias vector b, and the activations from the preceding layer make up this layer. The number of input neurons and the activation function are input into the dense layer.

Dropout

Dropout is a strategy for dealing with Imbalanced datasets. The fraction of neurons to drop is passed to the Dropout method in the keras. Layers module as a float between 0 and 1. Dropout prevents overfitting by setting a fraction rate of input values to 0 at random intervals during training.

Softmax

The softmax classification layer's primary intention is to convert all of the net activations in the final output layer into a sequence of values that may be read as probabilities. The final layer takes in parameters such as the number of output labels and the softmax activation. After the model has been generated, it is used to train it. To calculate the loss sustained during training and accuracy, use the compile function. The number of epochs and batch size are adjusted to fit the model. It's then applied to predicting the class label. The user chooses an image, which is

then tested. The image is chosen using the simple GUI module. Each image is adjusted to fit within a regular frame.

Pseudocode for the Processing of Convolution layers

```
for j = 1:J {% Loop on output feature maps
  for i=1:1 {% Loop on Input feature maps
    for m= 1:M {% Loop on rows of output feature maps
      for n= 1:N (% Loop on columns of output feature maps
        for p = 1:P {% Loop on rows of filer kernel
          for q=1:Q {% Loop on columns of filter kernel
            OFI (m,n) = OFİ(m, n) + Fij (p, q) x IF (m+p-1, n + q − 1);
}}}}}}% Loops end
```

3.5 Run model

The MobileNet architecture is used in the system. The model is trained and processed by the neural network by itself. In order for the input parameters to match the target, CNN looks for patterns in the training data. A machine learning model that may be used to make predictions is the result of the training process. Image is processed through depthwise separable convolution, as linked to image processing approaches deployed in similar works. The system recognizes ten diseases, including acne, eczema, psoriasis, and melanoma, among others, with over 5000 data samples. A CNN classification algorithm is a layered architecture in which numerous layers conduct different operations on visual data to train and test it. The Fully Connected Layer multiplies an input with a weight matrix, adds a bias vector, and applies the SoftMax Layer to create a model for the classification layer. SoftMax Layer is a multiclass classification function that uses a logistic activation function.

EXPERIMENTAL RESULTS AND DISCUSSION

4. EXPERIMENTAL RESULTS AND DISCUSSION

The system is implemented in Python and implemented in a Jupyter notebook. In this study, above 5000 skin photographs taken from the Internet were used by numerous dermatological condition patients. The results of the implementation are displayed in the input photos are preprocessed, then features are retrieved using a pretrained CNN. The proposed methodology, accepts user input photos by creating a user interface (UI) through which the user can submit skin images for analysis. HTML is used to create the user - friendly interface. The input image is processed to feature extraction and a machine learning-based classifier is trained in the backend, and feature extraction and classifier stages are done based on the input image provided by the user. Finally, a prediction is made, along with a medical recommendation.

The proposed approach has 91 percent accuracy in detecting ten skin disorders. 15 percent of the photographs were utilised for validation, 5% for testing, and 80% images for training. The system is effective. The technology has a 91 percent detection rate. Able to construct a light weight model using the MobileNet architecture, which may be used on devices with even lower processing power. In the scope of skin disease identification, this approach is useful.

Table.1 Evaluation of MobileNet Model by its Value Loss and Accuracy

S.No	Model	Accuracy	Value loss
1	Mobilenet	91	1.3

```
Epoch 00005: val_top_3_accuracy improved from 0.78331 to 0.83146, saving model to model.h5
Epoch 6/10
334/334 [==============================] - 2623s 8s/step - loss: 1.9139 - categorical_accuracy: 0.5970 - top_2_accura
cy: 0.7788 - top_3_accuracy: 0.8624 - val_loss: 1.4002 - val_categorical_accuracy: 0.5474 - val_top_2_accuracy: 0.728
7 - val_top_3_accuracy: 0.8363

Epoch 00006: val_top_3_accuracy improved from 0.83146 to 0.83628, saving model to model.h5
Epoch 7/10
334/334 [==============================] - 1631s 5s/step - loss: 1.8330 - categorical_accuracy: 0.6161 - top_2_accura
cy: 0.7964 - top_3_accuracy: 0.8735 - val_loss: 1.1980 - val_categorical_accuracy: 0.5939 - val_top_2_accuracy: 0.765
7 - val_top_3_accuracy: 0.8555

Epoch 00007: val_top_3_accuracy improved from 0.83628 to 0.85554, saving model to model.h5
Epoch 8/10
334/334 [==============================] - 1542s 5s/step - loss: 1.7645 - categorical_accuracy: 0.6302 - top_2_accura
cy: 0.8049 - top_3_accuracy: 0.8823 - val_loss: 1.4302 - val_categorical_accuracy: 0.5698 - val_top_2_accuracy: 0.736
8 - val_top_3_accuracy: 0.8379

Epoch 00008: val_top_3_accuracy did not improve from 0.85554
Epoch 9/10
334/334 [==============================] - 1538s 5s/step - loss: 1.6555 - categorical_accuracy: 0.6551 - top_2_accura
cy: 0.8264 - top_3_accuracy: 0.8984 - val_loss: 1.5142 - val_categorical_accuracy: 0.4960 - val_top_2_accuracy: 0.712
7 - val_top_3_accuracy: 0.8074

Epoch 00009: val_top_3_accuracy did not improve from 0.85554

Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.0024999999441206455.
Epoch 10/10
334/334 [==============================] - 1506s 5s/step - loss: 1.4440 - categorical_accuracy: 0.6945 - top_2_accura
cy: 0.8553 - top_3_accuracy: 0.9198 - val_loss: 0.9825 - val_categorical_accuracy: 0.6742 - val_top_2_accuracy: 0.839
5 - val_top_3_accuracy: 0.9101

Epoch 00010: val_top_3_accuracy improved from 0.85554 to 0.91011, saving model to model.h5
```

Fig.3 In this system ten iteration will give 91% accuracy

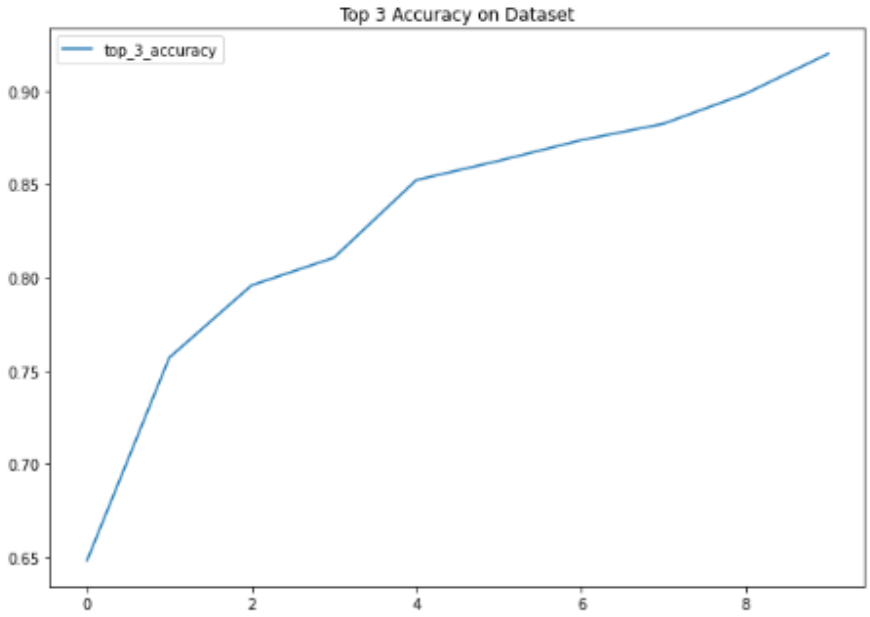


Fig.4 Graphical representation for accuracy

CONCLUSION

5. CONCLUSION

The model is implemented on a jupyter notebook running on a Windows operating system. This work focuses on a model developed using the CNN and Mobilenet structures for skin image diagnosis, with a dataset that contains photos of skin diseased people. Based on results, it can be concluded that Convolutional Neural Networks are capable of extracting features from raw picture data for the identification of skin diseases. The suggested system can detect a wide range of diseases with numerous dispersed locations close together, but it cannot detect diseases with distributed regions separated by distance. However, accurate detection necessitates the development of reliable detection techniques that may be employed by expert doctors in disease detection. The findings demonstrate that CNN can recognise skin diseases. The experiment produced considerable results in terms of accurate skin cancer diagnosis, and also learned that internal system tweaks can improve the system's performance. This model detected ten forms of skin illnesses. This research outlined the achievements of deep learning in the field of skin disease diagnostics. So the tools are free to use and available to the user, the system can be deployed for free. Based on the assumption that the machine learning data set was limited, the system was able to accurately identify the condition.

FUTURE ENHANCEMENT

6. FUTURE ENHANCEMENT

The potential benefits of applying deep learning to diagnose skin problems automatically are ridiculously large. As a result, must keep in mind that only an automated skin-cancer detection model should be thoroughly tested before being employed in real-world clinical detection. It also provides a straightforward user interface for ease of usage. Because this technology is movable, it can also be used in remote regions. In the future, the application could provide information and the location of surrounding doctors. The website currently detects ten diseases. More diseases can be training into the model in the future. By far, the system is more accurate, and it may be further enhanced by increasing dataset sizes. By making small modifications, this model can immediately be used in real-life diagnosis. Because of its stability and accuracy, Convolutional Neural Networks may represent the future of prediction software. The algorithms can be made more efficient and perform better by using Mobilenet. Other architecture, in addition to CNN, could be used to improve accuracy.

Further improvements would be to increase the amount of photographs in the data set to improve the system's learning capabilities, as the efficiency of the deep learning algorithm improves with larger data sets. The data set can also be created using the users input image. The photos can be saved in a database and used for supervised or unsupervised learning depending on whether they are categorized correctly. Also, the system has only been trained on a few diseases so far, but if and when the data set for each disease becomes accessible, it will be expanded to many more in the future.

BIBLOGRAPHY

7. BIBLOGRAPHY

Anushree D. Shetty , Deepthi P. Dsouza, S. G. Keerthi (July,2021)

Mobile Application based Skin Disease Detection using Mobilenet Model (International Journal of Research in Engineering, Science and Management) Volume 4, Issue 7

Divya Shree D V, Goggi Pragna, Prathibha L, Sushmetha G R, Mary M'Dsouza(July-2020)

Skin Disease Detection Using Image Processing and Neural Networks (International Journal of Progressive Research in Science and Engineering) Volume-1, Issue-4

Sruthi Chintalapudi, Vikas Prateek Mishra, Shubham Sharma, Sunil Kumar(2021)

Skin Disease Detection Using Deep Learning (International Research Journal of Engineering and Technology – IRJET), Volume: 08 Issue: 04

Leelavathy S, Jaichandran R, Shobana R, Vasudevan, Sreejith S Prasad and Nihad (2020)

Skin Disease Detection Using Computer Vision And Machine Learning Technique (European Journal of Molecular & Clinical Medicine) Volume 7, Issue 4.

Evani Sai Krishna Karthik, Potu Teja; Vaggu Sridhar and B. Priyanka(2021)

Detection of Skin Diseases and Classification using Deep Learning Algorithm (International Journal for Modern Trends in Science and Technology) 7, 0706203,

Srushti, Varshitha Makam, Sushmitha Shet, Swathi V A(2020)

Skin Disease Detection Using Deep Learning, International Journal for Research Trends and Innovation Volume 5, Issue 8 [ISSN: 2456-3315]

Navabharathi S1, Padmadevi S2, Vishnupriya R3, Ganesh K R(2020)

Digital Dermatology : Skin Disease Detection Using Image Processing, www.ijariie.com Vol-6 Issue-2 2020 [IJARIIE-ISSN(O)-2395-439

Payal Bose, Prof. Samir K. Bandyopadhyay, Prof. Amiya Bhaumik, Dr. Sandeep Poddar,

Skin Disease Detection: Machine Learning vs Deep Learning https://doi.org/10.20944/preprints202109.0209.v1.

Abunadi, I. Senan, E.M. (2021)

Deep Learning and Machine Learning Techniques of Diagnosis Dermoscopy Images for Early Detection of Skin Diseases. Electronics ,https://www.mdpi.com/journal/electronics 10, 3158. https://doi.org/10.3390/electronics10243158

K. V. Swamy and B. Divya,(2021)

Skin Disease Classification using Machine Learning Algorithms,2nd International Conference on Communication, Computing and Industry 4.0 (C2I4),pp. 1-5, doi: 10.1109/C2I454156.2021.9689338.

Nikhil Pancholi, Silky Goel, Rahul Nijhawan, Siddharth Gupta,

Classification and Detection of Acne on the Skin using Deep Learning Algorithms, 19th OITS International Conference on Information Technology (OCIT), pp.110-114.

S.Sindoora, S.Yogarasi, S. G. I. (2020)

Skin Disease Detection and Classification using Deep Learning Algorithms International Journal of Advanced Science and Technology, 29(3s), 255 - 260. http://sersc.org/journals/index.php/IJAST/article/view/5587.

Sourav Kumar Patnaik, Mansher SinghSidhu,Yaagyanika Gehlot, BhairviSharma P Muthu(2018)

Automated Skin Disease Identification using Deep Learning Algorithm, Biomedical & Pharmacology Journal, September 2018.

J. Rathod, V. Waghmode, A. Sodha and P. Bhavathankar, (2018)

Diagnosis of skin diseases using Convolutional Neural Networks, Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1048-1051, doi: 10.1109/ICECA.2018.8474593.

ANNEXURE

ANNEXURE

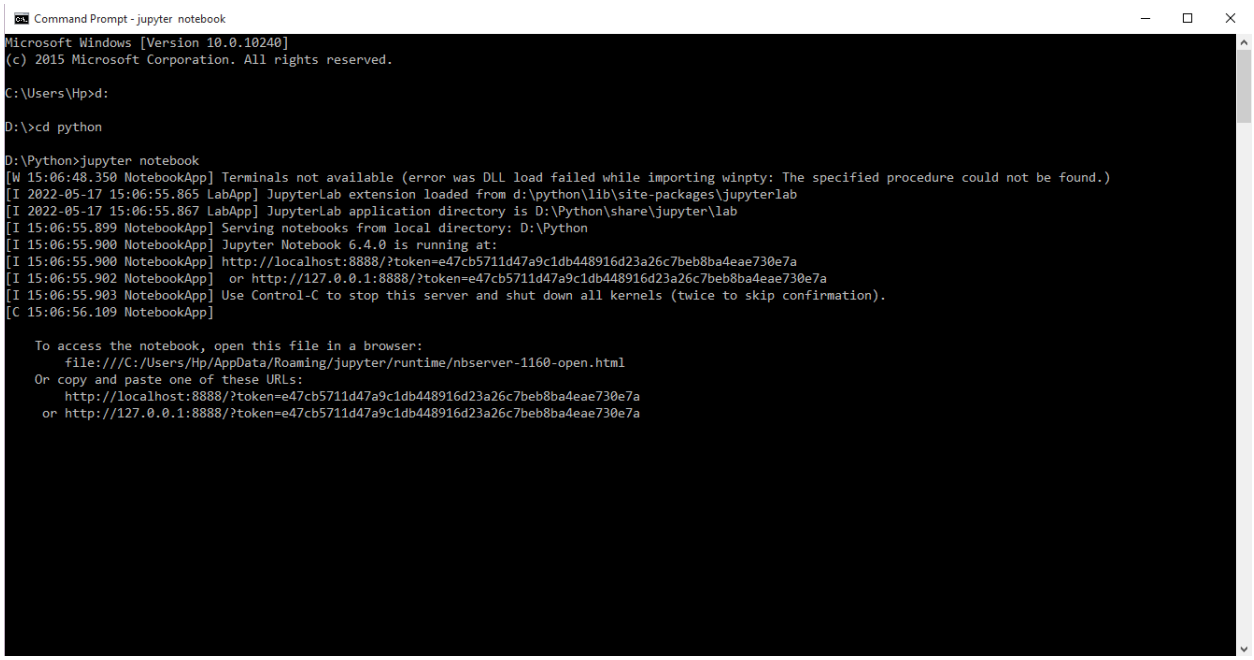


Fig.5 Command prompt

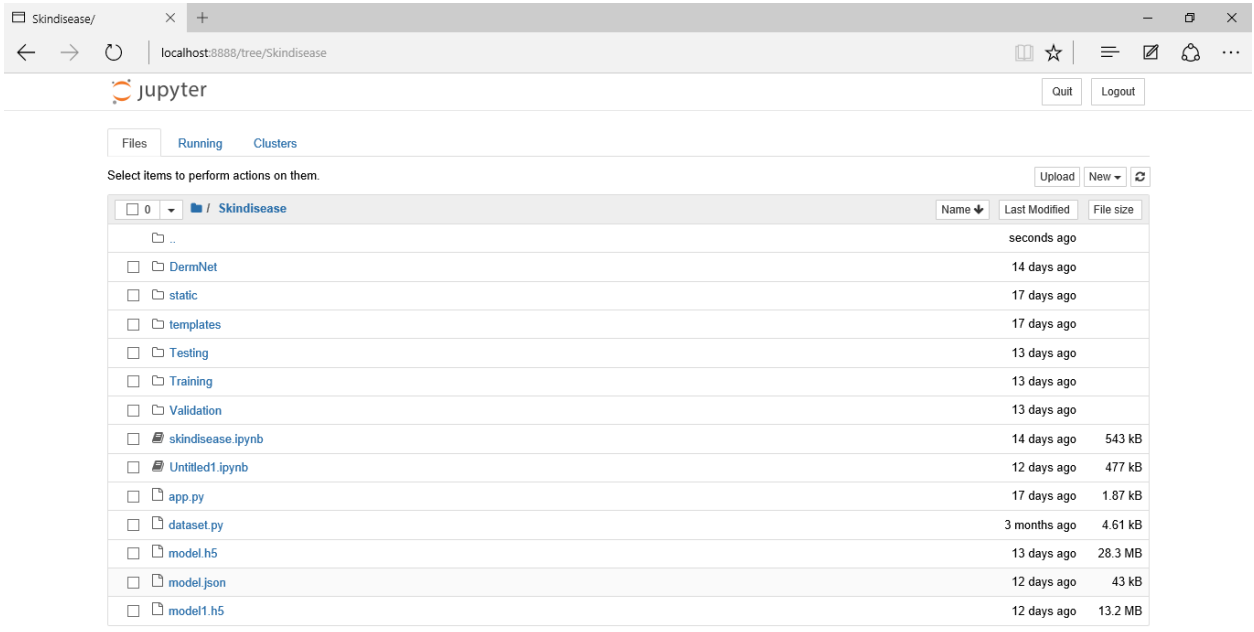


Fig.6 Jupyter Notebook Homepage

```
In [6]: import os
ROOT_FOLDER = 'DermNet/'
totalFiles = 0
totalDir = 0
for base, dirs, files in os.walk(ROOT_FOLDER):
    print('Searching in : ',base)
    for directories in dirs:
        totalDir += 1
    for Files in files:
        totalFiles += 1

print('Total number of files',totalFiles)
print('Total Number of directories',totalDir)
print('Total:',(totalDir + totalFiles))
```

```
Searching in :  DermNet/
Searching in :  DermNet/Acne-and-Rosacea-Photos
Searching in :  DermNet/Eczema-Photos
Searching in :  DermNet/Melanoma-Skin-Cancer-Nevi-and-Moles
Searching in :  DermNet/Nail-Fungus-and-other-Nail-Disease
Searching in :  DermNet/Psoriasis-pictures-Lichen-Planus-and-related-diseases
Searching in :  DermNet/Scabies-Lyme-Disease-and-other-Infestations-and-Bites
Searching in :  DermNet/Seborrheic-Keratoses-and-other-Benign-Tumors
Searching in :  DermNet/Systemic-Disease
Searching in :  DermNet/Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections
Searching in :  DermNet/Warts-Molluscum-and-other-Viral-Infections
Total number of files 6282
Total Number of directories 10
Total: 6292
```

Fig.7 Listing Directories and count the total files

```
In [7]: classes=os.listdir(rootdir)
source_path=[f'DermNet/{a}' for a in classes]

for cl_dir,cl_path in zip(classes,source_path):
    print(cl_dir,': ',len(os.listdir(cl_path)))
```

```
Acne-and-Rosacea-Photos :  615
Eczema-Photos :  593
Melanoma-Skin-Cancer-Nevi-and-Moles :  629
Nail-Fungus-and-other-Nail-Disease :  795
Psoriasis-pictures-Lichen-Planus-and-related-diseases :  918
Scabies-Lyme-Disease-and-other-Infestations-and-Bites :  278
Seborrheic-Keratoses-and-other-Benign-Tumors :  946
Systemic-Disease :  182
Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections :  591
Warts-Molluscum-and-other-Viral-Infections :  735
```

Fig.8 Counting the files in each directories

```
In [9]: TRAINING_PATH='Training'
VALIDATION_PATH='Validation'
TESTING_PATH='Testing'
training_dir_path=[f'Training/{a}' for a in classes]
print('\n',training_dir_path)
validation_dir_path=[f'Validation/{a}' for a in classes]
print('\n',validation_dir_path)
testing_dir_path=[f'Testing/{a}' for a in classes]
print('\n',testing_dir_path)
```

```
 ['Training/Acne-and-Rosacea-Photos', 'Training/Eczema-Photos', 'Training/Melanoma-Skin-Cancer-Nevi-and-Moles', 'Trai
ning/Nail-Fungus-and-other-Nail-Disease', 'Training/Psoriasis-pictures-Lichen-Planus-and-related-diseases', 'Trainin
g/Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Training/Seborrheic-Keratoses-and-other-Benign-Tumors', 'T
raining/Systemic-Disease', 'Training/Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Training/Warts-Mollusc
um-and-other-Viral-Infections']

 ['Validation/Acne-and-Rosacea-Photos', 'Validation/Eczema-Photos', 'Validation/Melanoma-Skin-Cancer-Nevi-and-Moles',
'Validation/Nail-Fungus-and-other-Nail-Disease', 'Validation/Psoriasis-pictures-Lichen-Planus-and-related-diseases',
'Validation/Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Validation/Seborrheic-Keratoses-and-other-Benign
-Tumors', 'Validation/Systemic-Disease', 'Validation/Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Valida
tion/Warts-Molluscum-and-other-Viral-Infections']

 ['Testing/Acne-and-Rosacea-Photos', 'Testing/Eczema-Photos', 'Testing/Melanoma-Skin-Cancer-Nevi-and-Moles', 'Testin
g/Nail-Fungus-and-other-Nail-Disease', 'Testing/Psoriasis-pictures-Lichen-Planus-and-related-diseases', 'Testing/Scab
ies-Lyme-Disease-and-other-Infestations-and-Bites', 'Testing/Seborrheic-Keratoses-and-other-Benign-Tumors', 'Testing/
Systemic-Disease', 'Testing/Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Testing/Warts-Molluscum-and-oth
er-Viral-Infections']
```

Fig.9 Creating the directories training ,testing and validation

```
In [15]: split_size = .85
for source,train_dir_path,val_dir_path,test_dir_path in zip(source_path,\
                                training_dir_path,validation_dir_path, testing_dir_path):
    print('source: ',source,'\n', train_dir_path,'\n',val_dir_path,'\n')
    split_data(source,train_dir_path,val_dir_path,test_dir_path, split_size)
    print('Splitting \n')
```

```
source:  DermNet/Acne-and-Rosacea-Photos
 Training/Acne-and-Rosacea-Photos
 Validation/Acne-and-Rosacea-Photos

Split Data
SOURCE:  DermNet/Acne-and-Rosacea-Photos
 TRAINING Training/Acne-and-Rosacea-Photos
 VALIDATION Validation/Acne-and-Rosacea-Photos
  615
training_length: 522
validation_length: 61
testing_length: 32
522
61
32
Splitting

source:  DermNet/Eczema-Photos
 Training/Eczema-Photos
```

Fig.10 spiltting the dataset for training 80% and 20% for testing and validation

```
In [16]: import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.image import imread
import pathlib

image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-oth

nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('D:/Python/Skindisease/Training/'+i+'/'))
    nimgs[i]=nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Training Dataset')
plt.show()
```

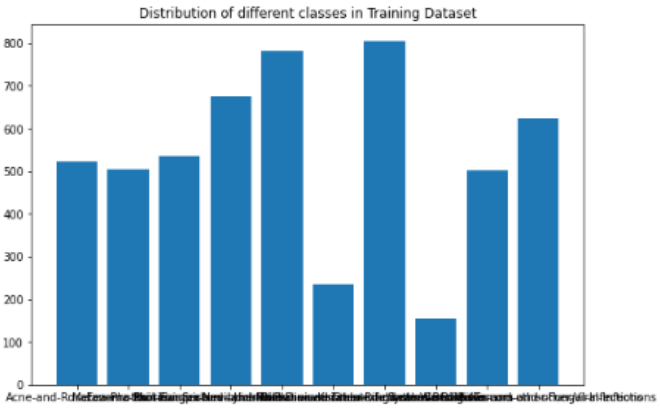


Fig.11 Plot for Training directory in different classes

```
In [17]: for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nail
    print('Training {} images are: '.format(i)+str(len(os.listdir('D:/Python/Skindisease/Training/'+i+'/'))))

Training Acne-and-Rosacea-Photos images are: 522
Training Eczema-Photos images are: 504
Training Melanoma-Skin-Cancer-Nevi-and-Moles images are: 534
Training Nail-Fungus-and-other-Nail-Disease images are: 675
Training Psoriasis-pictures-Lichen-Planus-and-related-diseases images are: 780
Training Scabies-Lyme-Disease-and-other-Infestations-and-Bites images are: 236
Training Seborrheic-Keratoses-and-other-Benign-Tumors images are: 804
Training Systemic-Disease images are: 154
Training Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections images are: 502
Training Warts-Molluscum-and-other-Viral-Infections images are: 624
```

Fig.12 counting the images Training directory in different classes

```
In [18]: image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-othe
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('D:/Python/Skindisease/Validation/'+i+'/'))
    nimgs[i]=nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Validation Dataset')
plt.show()
```

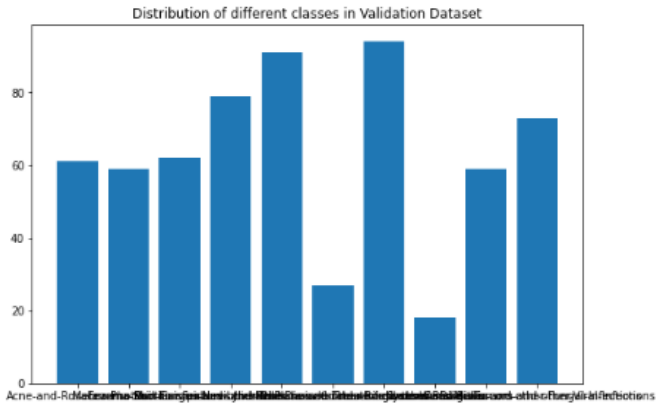


Fig.11 Plot for Validation directory in different classes

```
In [19]: for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nail
    print('Valid {} images are: '.format(i)+str(len(os.listdir('D:/Python/Skindisease/Validation/'+i+'/'))))
```

```
Valid Acne-and-Rosacea-Photos images are: 61
Valid Eczema-Photos images are: 59
Valid Melanoma-Skin-Cancer-Nevi-and-Moles images are: 62
Valid Nail-Fungus-and-other-Nail-Disease images are: 79
Valid Psoriasis-pictures-Lichen-Planus-and-related-diseases images are: 91
Valid Scabies-Lyme-Disease-and-other-Infestations-and-Bites images are: 27
Valid Seborrheic-Keratoses-and-other-Benign-Tumors images are: 94
Valid Systemic-Disease images are: 18
Valid Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections images are: 59
Valid Warts-Molluscum-and-other-Viral-Infections images are: 73
```

Fig.12 counting the images Validation directory in different classes

```
In [20]: image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-oth
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('D:/Python/Skindisease/Testing/'+i+'/'))
    nimgs[i]=nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Testing Dataset')
plt.show()
```

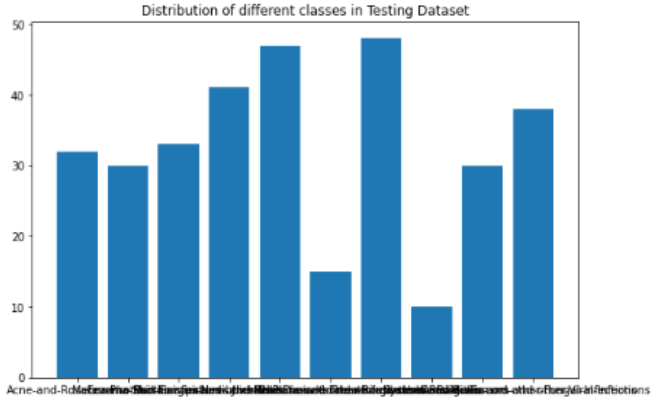


Fig.11 Plot for Testing directory in different classes

```
In [21]: for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nai
    print('Testing {} images are: '.format(i)+str(len(os.listdir('D:/Python/Skindisease/Testing/'+i+'/'))))
```

```
Testing Acne-and-Rosacea-Photos images are: 32
Testing Eczema-Photos images are: 30
Testing Melanoma-Skin-Cancer-Nevi-and-Moles images are: 33
Testing Nail-Fungus-and-other-Nail-Disease images are: 41
Testing Psoriasis-pictures-Lichen-Planus-and-related-diseases images are: 47
Testing Scabies-Lyme-Disease-and-other-Infestations-and-Bites images are: 15
Testing Seborrheic-Keratoses-and-other-Benign-Tumors images are: 48
Testing Systemic-Disease images are: 10
Testing Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections images are: 30
Testing Warts-Molluscum-and-other-Viral-Infections images are: 38
```

Fig.12 counting the images Testing directory in different classes

```
In [28]: mobile = tensorflow.keras.applications.mobilenet.MobileNet()

In [29]: mobile.summary()
```

```
Model: "mobilenet_1.00_224"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048
```

Fig.13 Mobilenet network

```
In [44]: plt.figure(figsize=(10,7))
         plt.plot(history.history['loss'], label='train loss')
         plt.plot(history.history['categorical_accuracy'], label='train_acc')
         plt.title("Training Loss and Accuracy on Dataset")
         plt.legend()
         plt.show()
```

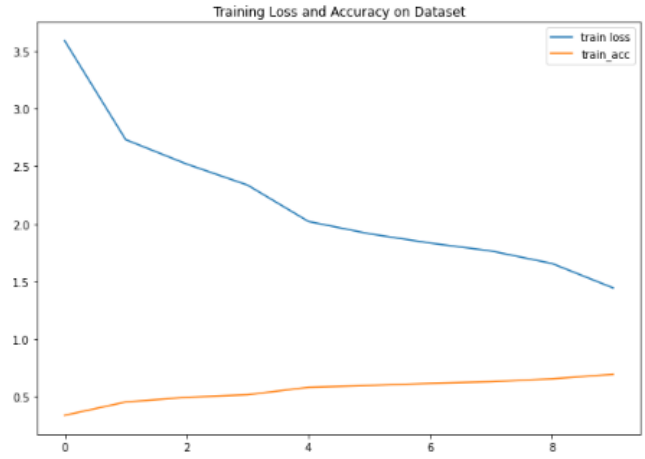


Fig.14 Training accuracy and Training loss

```
In [46]: # number of epochs vs Top 3 accuracy
plt.figure(figsize=(10,7))
plt.plot(history.history['top_3_accuracy'], label='top_3_accuracy')
plt.title("Top 3 Accuracy on Dataset")
plt.legend()
plt.show()
```

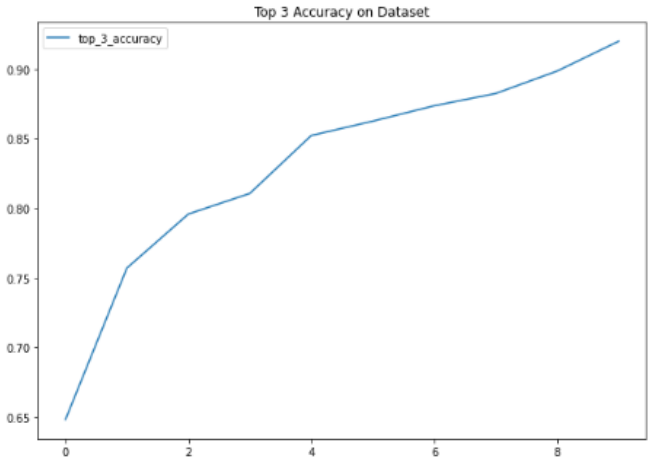


Fig.15 Accuracy Graph

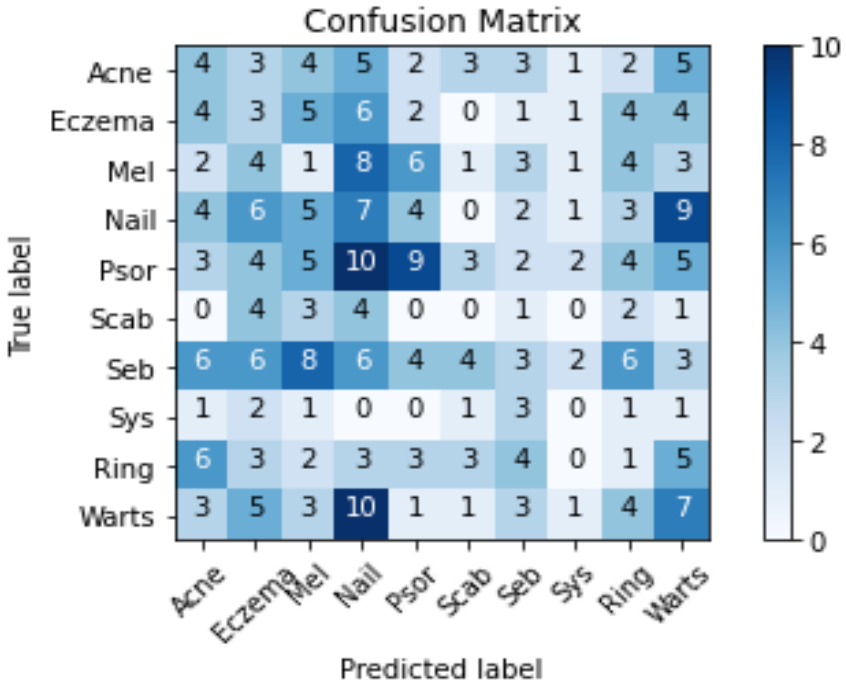


Fig.16 Confusion matrix

CODING

```
import os
import warnings
warnings.filterwarnings('ignore')
# Get all the paths
data_dir_list = os.listdir('D:/Python/Skindisease/DermNet')
print(data_dir_list)
path, dirs, files = next(os.walk("D:/Python/Skindisease/DermNet"))
file_count = len(files)
# print(file_count)
from os import listdir
# get the path or directory
rootdir ='DermNet'
len(os.listdir(rootdir))
for fol_dir in os.listdir('DermNet'):
    print(fol_dir)
import os
ROOT_FOLDER = 'DermNet/'
totalFiles = 0
totalDir = 0
for base, dirs, files in os.walk(ROOT_FOLDER):
   print('Searching in : ',base)
   for directories in dirs:
      totalDir += 1
   for Files in files:
      totalFiles += 1
print('Total number of files ',totalFiles)
```

```
print('Total Number of directories ',totalDir)
print('Total: ',(totalDir + totalFiles))
classes=os.listdir(rootdir)
source_path=[f'DermNet/{a}' for a in classes]
for cl_dir,cl_path in zip(classes,source_path):
    print(cl_dir,': ',len(os.listdir(cl_path)))
try:
    os.mkdir('Training')
    os.mkdir('Validation')
    os.mkdir('Testing')
except OSError:
    pass
TRAINING_PATH='Training'
VALIDATION_PATH='Validation'
TESTING_PATH='Testing'
training_dir_path=[f'Training/{a}' for a in classes]
print('\n',training_dir_path)
validation_dir_path=[f'Validation/{a}' for a in classes]
print('\n',validation_dir_path)
testing_dir_path=[f'Testing/{a}' for a in classes]
print('\n',testing_dir_path)
for train_dir_path in training_dir_path:
    try:
        os.mkdir(train_dir_path)
    except OSError:
        pass
for val_dir_path in validation_dir_path:
```

```
    try:
        os.mkdir(val_dir_path)
    except OSError:
        pass
for test_dir_path in testing_dir_path:
    try:
        os.mkdir(test_dir_path)
    except OSError:
        pass
import random
import shutil
from shutil import copyfile
def split_data(SOURCE, TRAINING, VALIDATION, TESTING, SPLIT_SIZE):
    files = [ ]
    print('Split Data')
    for filename in os.listdir(SOURCE):
        file = SOURCE +'/'+ filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")
training_length = int( len(files)* SPLIT_SIZE)
validation_length = int(len(files) * 0.10)
testing_length = int(len(files) - training_length - validation_length)
print('SOURCE: ',SOURCE, '\n TRAINING', TRAINING, '\nVALIDATION',VALIDATION,
'\n ',len(files))
print('training_length:',training_length)
print('validation_length:',validation_length)
```

```
print('testing_length:',testing_length)
shuffled_set = random.sample(files, len(files))
training_set = shuffled_set[0:training_length]
validation_set = shuffled_set[training_length:(training_length+validation_length)]
testing_set=shuffled_set[:testing_length]
print(len(training_set))
print(len(validation_set))
print(len(testing_set))
for filename in training_set:
    this_file = SOURCE +'/'+ filename
    destination = TRAINING +'/'+ filename
    copyfile(this_file, destination)
for filename in validation_set:
    this_file = SOURCE +'/'+ filename
    destination = VALIDATION+'/' + filename
    copyfile(this_file, destination)
for filename in validation_set:
    this_file = SOURCE +'/'+ filename
    destination = TESTING+'/' + filename
    copyfile(this_file, destination)
split_size = .85
for source,train_dir_path,val_dir_path,test_dir_path in zip(source_path,\ training_dir_path,
validation_dir_path, testing_dir_path):
print('source: ',source,'\n', train_dir_path,'\n',val_dir_path,'\n')
split_data(source,train_dir_path,val_dir_path,test_dir_path, split_size)
print('Splitting \n')
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from matplotlib.image import imread

import pathlib

image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-diseases', 'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-other-Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Warts-Molluscum-and-other-Viral-Infections']

nimgs = {}

for i in image_folder:

    nimages = len(os.listdir('D:/Python/Skindisease/Training/'+i+'/'))

    nimgs[i]=nimages

plt.figure(figsize=(9, 6))

plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')

plt.xticks(range(len(nimgs)), list(nimgs.keys()))

plt.title('Distribution of different classes in Training Dataset')

plt.show()

for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-diseases', 'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-other-Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Warts-Molluscum-and-other-Viral-Infections']:

print('Training {} images are: '.format(i)+str (len(os.listdir ('D:/Python/Skindisease/Training/'+i+'/'))))

image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles', 'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-diseases', 'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-other-Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-Infections', 'Warts-Molluscum-and-other-Viral-Infections']

nimgs = {}

for i in image_folder:

    nimages = len(os.listdir('D:/Python/Skindisease/Validation/'+i+'/'))

    nimgs[i]=nimages

plt.figure(figsize=(9, 6))
```

```
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')

plt.xticks(range(len(nimgs)), list(nimgs.keys()))

plt.title('Distribution of different classes in Validation Dataset')

plt.show()

for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles',
'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-diseases',
'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-other-
Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-
Infections', 'Warts-Molluscum-and-other-Viral-Infections']:

    print('Valid {} images are: '.format(i)+str(len(os.listdir
('D:/Python/Skindisease/Validation/'+i+'/'))))

image_folder = ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-
and-Moles', 'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-
diseases', 'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-
other-Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-
Infections', 'Warts-Molluscum-and-other-Viral-Infections']

nimgs = {}

for i in image_folder:

    nimages = len(os.listdir('D:/Python/Skindisease/Testing/'+i+'/'))

    nimgs[i]=nimages

plt.figure(figsize=(9, 6))

plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')

plt.xticks(range(len(nimgs)), list(nimgs.keys()))

plt.title('Distribution of different classes in Testing Dataset')

plt.show()

for i in ['Acne-and-Rosacea-Photos', 'Eczema-Photos', 'Melanoma-Skin-Cancer-Nevi-and-Moles',
'Nail-Fungus-and-other-Nail-Disease', 'Psoriasis-pictures-Lichen-Planus-and-related-diseases',
'Scabies-Lyme-Disease-and-other-Infestations-and-Bites', 'Seborrheic-Keratoses-and-other-
Benign-Tumors', 'Systemic-Disease', 'Tinea-Ringworm-Candidiasis-and-other-Fungal-
Infections', 'Warts-Molluscum-and-other-Viral-Infections']:

    print('Testing {} images are:
'.format(i)+str(len(os.listdir('D:/Python/Skindisease/Testing/'+i+'/'))))

import tensorflow
```

```
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
img_width=256; img_height=256
batch_size=16
TRAINING_DIR = 'D:/Python/Skindisease/Training/'
train_datagen = ImageDataGenerator(rescale = 1/255.0,
                          rotation_range=180,
                           width_shift_range=0.1,
                           height_shift_range=0.1,
                           zoom_range=0.1,
                           horizontal_flip=True,
                           vertical_flip=True,
                           brightness_range = [0.7, 1.3] )
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                    batch_size=batch_size,
                                    class_mode='categorical',
                                    color_mode = 'rgb',
                                    target_size=(img_height, img_width))
VALIDATION_DIR = 'D:/Python/Skindisease/Validation/'
validation_datagen = ImageDataGenerator(rescale = 1/255.0,
                           rotation_range=180,
                           width_shift_range=0.1,
                           height_shift_range=0.1,
```

```
                    zoom_range=0.1,
                    horizontal_flip=True,
                    vertical_flip=True,
                    brightness_range = [0.7, 1.3])
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        color_mode='rgb',
                                        target_size=(img_height, img_width))
TESTING_DIR = 'D:/Python/Skindisease/Testing/'
testing_datagen = ImageDataGenerator(rescale = 1/255.0,
                                    rotation_range=180,
                                    width_shift_range=0.1,
                                     height_shift_range=0.1,
                                    zoom_range=0.1,
                                    horizontal_flip=True,
                                    vertical_flip=True,
                                    brightness_range = [0.7, 1.3])
testing_generator = testing_datagen.flow_from_directory(TESTING_DIR,
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        color_mode='rgb',
                                        target_size=(img_height, img_width))
import numpy as np
num_train_samples = len(training_dir_path)
num_val_samples = len(validation_dir_path)
train_batch_size = 10
```

```
val_batch_size = 10
image_size = 224
train_steps = np.ceil(num_train_samples / train_batch_size)
val_steps = np.ceil(num_val_samples / val_batch_size)
mobile = tensorflow.keras.applications.mobilenet.MobileNet()
mobile.summary()
# How many layers does MobileNet have?
len(mobile.layers)
x = mobile.layers[-6].output
x = Dropout(0.25)(x)
predictions = Dense(10, activation='softmax')(x)
model = Model(inputs=mobile.input, outputs=predictions)
model=Model(inputs=mobile.input,outputs=predictions)
model.summary()
for layer in model.layers[:-23]:
    layer.trainable = False
from tensorflow.keras.metrics import categorical_accuracy, top_k_categorical_accuracy
def top_3_accuracy(y_true, y_pred):
    return top_k_categorical_accuracy(y_true, y_pred, k=3)
def top_2_accuracy(y_true, y_pred):
    return top_k_categorical_accuracy(y_true, y_pred, k=2)
model.compile(Adam(learning_rate=0.01), loss='categorical_crossentropy',
metrics=[categorical_accuracy, top_2_accuracy, top_3_accuracy])
train_generator.class_indices
validation_generator.class_indices
class_weights={
    0:3.0,
    1:1.5,
```

```
    2:2.0,
    3:2.4,
    4:1.0,
    5:1.4,
    6:1.3,
    7:2.5,
    8:2.2,
    9:1.6,
}
filepath = "model.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_top_3_accuracy', verbose=1,
save_best_only=True, mode='max')
reduce_lr = ReduceLROnPlateau(monitor='val_top_3_accuracy', factor=0.5, patience=2,
verbose=1, mode='max', min_lr=0.00001)
callbacks_list = [checkpoint, reduce_lr]
history = model.fit_generator(train_generator,
# steps_per_epoch=int(918/45),
class_weight=class_weights, validation_data=validation_generator,
# validation_steps=int(111/45),
epochs=10, verbose=1, callbacks=callbacks_list)
model.metrics_names
val_loss, val_cat_acc, val_top_2_acc, val_top_3_acc = \model.evaluate_generator
(validation_generator, steps=len(validation_dir_path))
print('val_loss:', val_loss)
print('val_cat_acc:', val_cat_acc)
print('val_top_2_acc:', val_top_2_acc)
print('val_top_3_acc:', val_top_3_acc)
val_loss, val_cat_acc, val_top_2_acc, val_top_3_acc = \
```

```
model.evaluate_generator(validation_generator, steps=len(validation_dir_path))
print('val_loss:', val_loss)
print('val_cat_acc:', val_cat_acc)
print('val_top_2_acc:', val_top_2_acc)
print('val_top_3_acc:', val_top_3_acc)
plt.figure(figsize=(10,7))
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['categorical_accuracy'], label='train_acc')
plt.title("Training Loss and Accuracy on Dataset")
plt.legend()
plt.show()
plt.figure(figsize=(10,7))
plt.plot(history.history['top_2_accuracy'], label='top_2_accuracy')
plt.title("Top 2 Accuracy on Dataset")
plt.legend()
plt.show()
# number of epochs vs Top 3 accuracy
plt.figure(figsize=(10,7))
plt.plot(history.history['top_3_accuracy'], label='top_3_accuracy')
plt.title("Top 3 Accuracy on Dataset")
plt.legend()
plt.show()
test_labels = testing_generator.classes
# We need these to plot the confusion matrix.
test_labels
# Print the label associated with each class
testing_generator.class_indices
```

```
# make a prediction
predictions = model.predict_generator(testing_generator, verbose=1)
predictions.shape
test_labels = testing_generator.classes
cm = confusion_matrix(test_labels, predictions.argmax(axis=1))
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import itertools
import shutil
import matplotlib.pyplot as plt
%matplotlib inline
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
```

```
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()


test_labels.shape
# argmax returns the index of the max value in a row
cm = confusion_matrix(test_labels, predictions.argmax(axis=1))
cm_plot_labels = ['Acne', 'Eczema', 'Mel', 'Nail', 'Psor', 'Scab', 'Seb', 'Sys', 'Ring', 'Warts']


plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
```