****

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**A Skill Development Project Report**

on

**ADVANCE AI-Deep Learning**

Submitted in fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

Submitted by

Manoj R: R20EF241

Pranav R: R20EF248

S B Keerthan: R20EF254

Yeshwanth Reddy M: R20EF279

2023-2024

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064

www.reva.edu.in

**DECLARATION**

We, Mr. S B Keerthan, Mr. Pranav R, Mr. Manoj R and Mr. Yeshwanth Reddy M the students of Bachelor of Technology, belong in to School of Computer Science And Engineering, REVA University, declare that this Skill development Project Report / Dissertation entitled “ADVANCE AI-Deep Learning” is the result the of Skill development program done at School of Computer Science And Engineering, REVA University.

We are submitting this Skill development Project Report / Dissertation in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering by the REVA University, Bangalore during the academic year 2022-2023.

*Signature of the candidates with dates*

*Certified that this project work submitted by S B Keerthan, Pranav R, Manoj R and* Yeshwanth Reddy M *has been carried out and the declaration made by the candidate is true to the best of my knowledge.*

|  |  |  |
| --- | --- | --- |
|  |  | *Signature of Director of School* |
|  |  | *Date:…………….* |
|  |  | *Official Seal of the School* |

****

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING.**

**CERTIFICATE**

Certified that the Skill Development project work entitled **ADVANCE AI-Deep Learning**

carried out under our guidance byMeghaMa’amare bonafide students of REVA University during the academic year 2022-2023, are submitting the Skill development project report in partial fulfillment for the award of **Bachelor of Technology** in Computer Science And Engineering during the academic year **2023-2024.**

|  |  |  |
| --- | --- | --- |
|  |  | **Signature with date** |
|  |  |  |
|  |  |  |
|  |  | **Dr. Ashwin Kumar U M** |
|  |  | **Director** |

**Index**

|  |  |  |
| --- | --- | --- |
| **SL.No** | **Contents** | **Page No.** |
| 1 | Abstract | 05 |
| 2 | Introduction | 06 |
| 3 | Problem Statement, Objectives and Goals | 07 |
| 4 | Project Scope | 07-08 |
| 5 | Methodology | 08 |
| 6 | Modules identified, code and Results | 09-25 |
| 7 | Project Implementation | 25-26 |
| 8 | Conclusions | 27 |
| 9 | References | 27 |

**SKIN DISEASE RECOGNITION**

**ABSTRACT**

*A complex topic such as dermatology makes it one of the most unexpected and challenging professions to diagnose due to the complexities of the subject matter involved. According to dermatology, it is a regular practice to do extensive tests on patients to ascertain the kind of skin illness they have been afflicted duration of time varies from one practitioner to the next, depending on their experience. It is also influenced by the individual's personal experience with the subject matter. Using a technique not restricted by these limits is essential to diagnose skin diseases without these limitations. This work provides an automated image-based method for diagnosing and categorizing skin problems that use machine learning classification. Computational approaches will be used to analyze, process, and relegate picture data to consider the many different characteristics of the photos that are being processed. Skin photographs are first filtered to remove undesirable noise from the image and then processed to enhance the picture's overall quality. It is possible to extract features from an image using advanced techniques such as Convolutional Neural Network (CNN), classify the picture using the softmax classifier's algorithm, and provide a diagnostic report as an output. With more accuracy and faster delivery of results than the previous technique, this application will be a more efficient and reliable system for dermatological illness diagnosis than the conventional method. Furthermore, this may be a reliable real-time teaching tool for medical students enrolled in the dermatology stream at a university studying dermatology.* *Our suggested solution is straightforward, quick, and does not need the purchase of costly equipment beyond a photographic camera and a system computer. Consequently, we offer a skin disease diagnosis approach related to image processing methodology and techniques. This procedure starts with a digital photo of the sick skin region, then analyzed to determine the kind of illness discovered.*

**INTRODUCTION**

Dermatologic diseases are the most predominant kind of disease globally. Despite its prevalence, it is challenging to diagnose and needs a high level of expertise. According to a poll, around 24% of the population contacts their general practitioner (GP) with a skin concern in a single year. When it comes to undergraduate dermatology education, there is an unequal (and often restrictive) curriculum, implying that trainees should review their existing talents and knowledge in this discipline. At the moment, Primary Care is responsible for treating about 90% of all skin disorders and problems.

Consequently, it is inferred that most skin disease complications may be cured if treatment is initiated early. Skin disease has significantly impacted the patient quality of life [1]. The prevalence of skin diseases is increasing, and early identification is crucial for improved outcomes. G.P.S have a critical role in the early diagnosis of skin problems [2]. Numerous efforts have been made to introduce outdated medicine in different regional areas of the globe, especially in less technically sophisticated countries. These attempts, however, have been impeded by barriers such as the high price of equipment and medical instruments and the nonexistence of medical competence. Skin disease is often caused by a mix of environmental and genetic causes. In the majority of the globe, the devices necessary for the early detection of these diseases are still not generally available to most people. The proposed study establishes a mechanism for recognizing various forms of these illnesses. The user provides the image of the skin sickness, which is analyzed by the system, which does feature extraction using the CNN algorithm and diagnoses the condition utilizing the softmax image classifier. If no illness is detected, the system responds negatively to the user. Consequently, a novel dermoscopy detection and classification strategy based on CNN is proposed.

A regular occurrence is that most public is unaware of the kind and stage of a skin illness. Some skin illnesses appear months after the disease has begun, enabling the condition to flourish and spread. It is due to an absence of medical understanding among the general populace. A dermatologist (a doctor specializing in skin problems) may have trouble spotting the issue and may be forced to employ costly laboratory testing to determine the kind and stage of the illness. Medical technology has progressed to the point that lasers and photonics-based equipment can detect skin illnesses quickly and precisely. However, the expense of such a diagnostic is presently restricted and prohibitive for the vast majority of people.

**Problem Statement:**

In this project, we try to implement a Deep Learning model that will help you identify skin diseases.

**Objectives:**

The main objective of this system is to achieve maximum accuracy of skin disease prediction. Detect the type of skin disease easily with accuracy and recommend the best and global medical suggestions. Other objective is to develop an accurate and reliable system that can automatically detect and classify skin diseases from images. The system would use deep learning algorithms such as convolutional neural networks (CNNs) to learn features from the images and classify them into different categories such as melanoma, psoriasis, eczema, and others.

**Goals:**

The main goals of the project would be:

Develop a large and diverse dataset of skin disease images with annotations for training and testing the deep learning models. Preprocess the data to ensure that it is suitable for use with deep learning models. Design and train deep learning models using a variety of architectures such as ResNet, Inception, and VGG. Evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1 score. Compare the performance of the different models and identify the best performing model. Deploy the best performing model as a web application or mobile application to allow for easy and convenient access to the system.

**Project Scope:**

The scope of a skin disease detection using deep learning project would include the following:

**Data collection:** Collecting a large and diverse dataset of skin disease images, which could include different skin types, ages, and genders. The dataset should be comprehensive enough to cover all major skin diseases and should be annotated with the correct disease label.

**Data preprocessing:** Preprocessing the dataset by resizing images, normalizing the pixel values, and augmenting the data to increase the number of images available for training the deep learning models.

**Model selection:** Selecting the appropriate deep learning model architecture based on the dataset and the performance metrics required.

**Model training:** Training the deep learning model using the prepared dataset and evaluating the performance of the model.

**Model optimization:** Optimizing the model using techniques such as hyperparameter tuning, regularization, and dropout to improve the model's accuracy and reduce overfitting.

**Deployment:** Deploying the final model as a web application or mobile application to allow for easy and convenient access to the system.

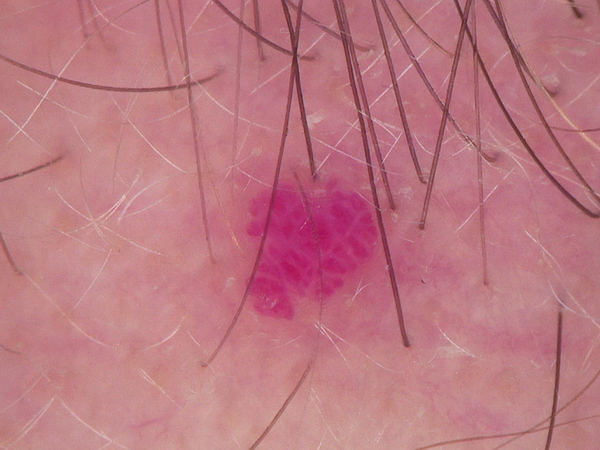
**Testing and validation:** Testing the system with new and unseen data to validate its accuracy and reliability.

**Methodology**

The Proposed methodology is an effective tool which can analyze the people input skin disease to predict skin disease. In this proposed system, hybrid architecture with image processing and machine learning techniques are used to predict type of disease with promising accuracy in a short period of time. The image processing phase invokes preprocessing, segmentation, feature extraction steps. The machine learning phase invokes 3 steps: processing, training and detection steps. The proposed system uses 2D Wavelet Transform algorithm for feature extraction in which color, texture and shape features are extracted from the skin input images. The correlation values are also been extracted from the input image. These values are passed onto classifier model. For classification the proposed system uses convolutional neural network (CNN). The classifier model detects common skin diseases like Psoriasis, Lichen Planus, Pityriasis Rosea. Integration of neural network provides good accuracy results. The proposed system act as a Common knowledge base for skin disease detection and medicinal prescription. This proposed system analyses different type of skin disease can be analyzed saving user time and cost.

**Code:**

**#Viewing the image**  
import cv2  
from google.colab.patches import cv2\_imshow  
a=cv2.imread("/content/drive/MyDrive/skin\_disease/Train/vascular lesion/ISIC\_0024475.jpg")  
cv2\_imshow(a)  
print(a.shape)



(450, 600, 3)

# **CNN**

**#Neural Network Architecture**  
import pandas as pd  
import tensorflow as tf  
from tensorflow import keras  
from keras import models  
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense  
model=models.Sequential() #Sequential Model  
model.add(Conv2D(32,(3,3),activation="relu",input\_shape=(200, 200, 3))) #Convolution Layer  
model.add(MaxPooling2D((2,2))) #Pooling Layer  
model.add(Conv2D(64,(3,3),activation="relu")) #Convolution Layer  
model.add(MaxPooling2D((2,2))) #Pooling Layer  
model.add(Conv2D(128,(3,3),activation="relu")) #Convolution Layer  
model.add(MaxPooling2D((2,2))) #Pooling Layer  
model.add(Conv2D(128,(3,3),activation="relu")) #Convolution Layer  
model.add(MaxPooling2D((2,2)))#Pooling Layer  
model.add(Flatten()) #Flattening  
model.add(Dense(256,activation="relu")) #Dense Layer  
model.add(Dense(256,activation="relu")) #Dense Layer  
model.add(Dense(128,activation="relu")) #Dense Layer  
model.add(Dense(9,activation="softmax")) #Output Layer  
model.summary() #Model Summary

Model: "sequential\_5"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 conv2d\_20 (Conv2D) (None, 198, 198, 32) 896   
   
 max\_pooling2d\_20 (MaxPoolin (None, 99, 99, 32) 0   
 g2D)   
   
 conv2d\_21 (Conv2D) (None, 97, 97, 64) 18496   
   
 max\_pooling2d\_21 (MaxPoolin (None, 48, 48, 64) 0   
 g2D)   
   
 conv2d\_22 (Conv2D) (None, 46, 46, 128) 73856   
   
 max\_pooling2d\_22 (MaxPoolin (None, 23, 23, 128) 0   
 g2D)   
   
 conv2d\_23 (Conv2D) (None, 21, 21, 128) 147584   
   
 max\_pooling2d\_23 (MaxPoolin (None, 10, 10, 128) 0   
 g2D)   
   
 flatten\_5 (Flatten) (None, 12800) 0   
   
 dense\_20 (Dense) (None, 256) 3277056   
   
 dense\_21 (Dense) (None, 256) 65792   
   
 dense\_22 (Dense) (None, 128) 32896   
   
 dense\_23 (Dense) (None, 9) 1161   
   
=================================================================  
Total params: 3,617,737  
Trainable params: 3,617,737  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**#Compiling the Model**model.compile(optimizer="Adam",loss=keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"])

**#ImageDataGenerator to access Image from Specified Directory**  
from keras.preprocessing.image import ImageDataGenerator  
x\_train\_datagen=ImageDataGenerator(rescale=1./255)  
x\_val\_datagen=ImageDataGenerator(rescale=1./255)

**#Specifying the Train, Test Directory Path , and image size**train\_datagen=x\_train\_datagen.flow\_from\_directory("/content/drive/MyDrive/skin\_disease/Train",  
 target\_size=(200,200),batch\_size=128,class\_mode="sparse")  
val\_datagen=x\_val\_datagen.flow\_from\_directory("/content/drive/MyDrive/skin\_disease/Test",  
 target\_size=(200,200),batch\_size=128,class\_mode="sparse")

Found 1366 images belonging to 9 classes.  
Found 1366 images belonging to 9 classes.

**# dir(train\_datagen)**  
for i in train\_datagen.classes[::]:  
 print(i,end=" ")

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

**#Training the model and saving the history in variable named h**h=model.fit(train\_datagen,epochs=30)

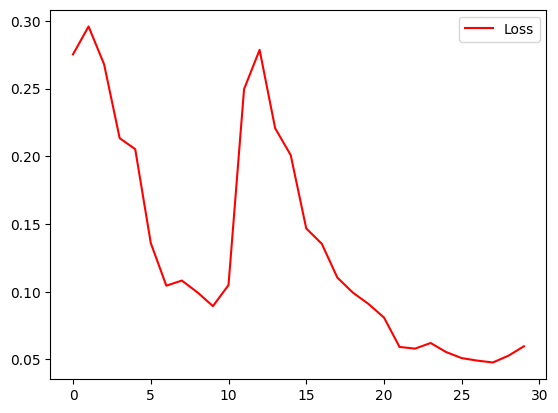
Epoch 1/30  
11/11 [==============================] - 30s 3s/step - loss: 0.2753 - accuracy: 0.8968  
Epoch 2/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2958 - accuracy: 0.8931  
Epoch 3/30  
11/11 [==============================] - 28s 2s/step - loss: 0.2679 - accuracy: 0.9034  
Epoch 4/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2134 - accuracy: 0.9143  
Epoch 5/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2053 - accuracy: 0.9180  
Epoch 6/30  
11/11 [==============================] - 27s 3s/step - loss: 0.1360 - accuracy: 0.9480  
Epoch 7/30  
11/11 [==============================] - 27s 2s/step - loss: 0.1045 - accuracy: 0.9517  
Epoch 8/30  
11/11 [==============================] - 27s 3s/step - loss: 0.1082 - accuracy: 0.9575  
Epoch 9/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0996 - accuracy: 0.9568  
Epoch 10/30  
11/11 [==============================] - 27s 3s/step - loss: 0.0894 - accuracy: 0.9649  
Epoch 11/30  
11/11 [==============================] - 27s 2s/step - loss: 0.1049 - accuracy: 0.9495  
Epoch 12/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2495 - accuracy: 0.9092  
Epoch 13/30  
11/11 [==============================] - 27s 3s/step - loss: 0.2786 - accuracy: 0.8982  
Epoch 14/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2207 - accuracy: 0.9173  
Epoch 15/30  
11/11 [==============================] - 27s 2s/step - loss: 0.2009 - accuracy: 0.9224  
Epoch 16/30  
11/11 [==============================] - 26s 2s/step - loss: 0.1467 - accuracy: 0.9436  
Epoch 17/30  
11/11 [==============================] - 26s 2s/step - loss: 0.1354 - accuracy: 0.9444  
Epoch 18/30  
11/11 [==============================] - 27s 2s/step - loss: 0.1104 - accuracy: 0.9590  
Epoch 19/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0993 - accuracy: 0.9561  
Epoch 20/30  
11/11 [==============================] - 26s 2s/step - loss: 0.0910 - accuracy: 0.9561  
Epoch 21/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0809 - accuracy: 0.9634  
Epoch 22/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0592 - accuracy: 0.9671  
Epoch 23/30  
11/11 [==============================] - 27s 3s/step - loss: 0.0580 - accuracy: 0.9656  
Epoch 24/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0621 - accuracy: 0.9641  
Epoch 25/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0554 - accuracy: 0.9678  
Epoch 26/30  
11/11 [==============================] - 28s 3s/step - loss: 0.0510 - accuracy: 0.9641  
Epoch 27/30  
11/11 [==============================] - 29s 3s/step - loss: 0.0491 - accuracy: 0.9685  
Epoch 28/30  
11/11 [==============================] - 28s 3s/step - loss: 0.0478 - accuracy: 0.9663  
Epoch 29/30  
11/11 [==============================] - 27s 3s/step - loss: 0.0528 - accuracy: 0.9678  
Epoch 30/30  
11/11 [==============================] - 27s 2s/step - loss: 0.0597 - accuracy: 0.9649

**#Evaluating the model Performance**  
model.evaluate(val\_datagen)

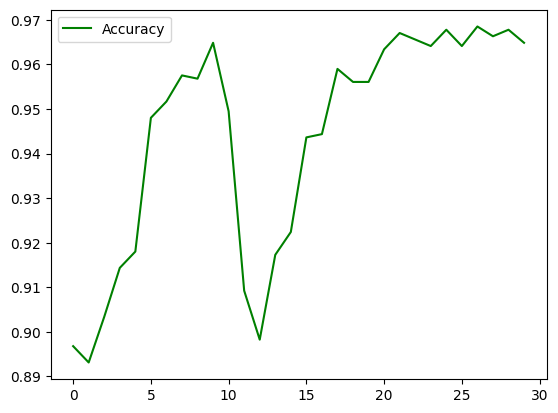
11/11 [==============================] - 27s 2s/step - loss: 0.0445 - accuracy: 0.9714

[0.044476065784692764, 0.9714494943618774]

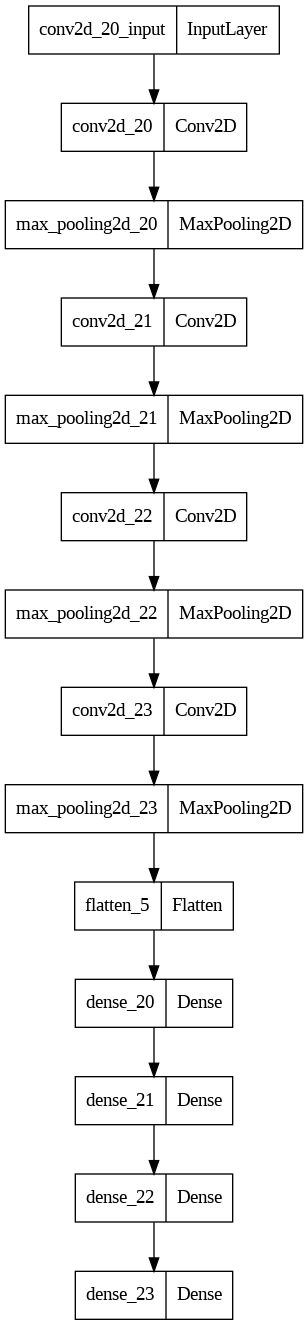
**#Plotting the Loss**  
import matplotlib.pyplot as plt  
plt.plot(h.history["loss"],color="red")  
# plt.plot(h.history["accuracy"],color="green")  
plt.legend(["Loss"],loc="upper right")  
plt.show()



**#plotting the Accuracy**   
import matplotlib.pyplot as plt  
plt.plot(h.history["accuracy"],color="green")  
plt.legend(["Accuracy"])  
plt.show()



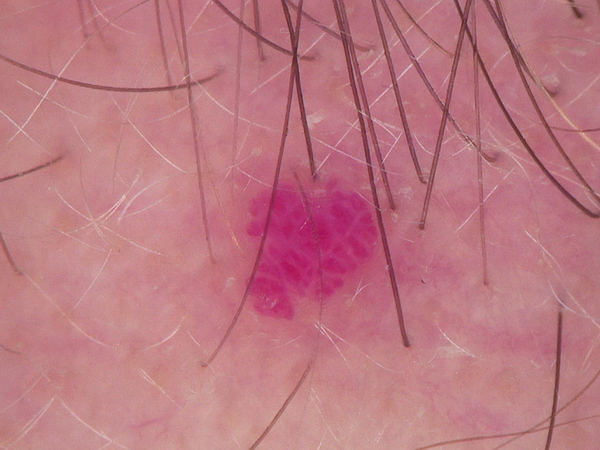
**#Diagrammatic representation of Neural Network**  
from keras.utils import plot\_model  
plot\_model(model)



**#Saving the Model**model.save("model1.h5")

**#Prediction**  
import numpy as np  
from keras.utils import load\_img, img\_to\_array  
a="/content/drive/MyDrive/skin\_disease/Train/vascular lesion/ISIC\_0024475.jpg" #image path  
test\_image = load\_img(a, target\_size=(200,200)) #loading the image  
test\_image =img\_to\_array(test\_image) #converting image to array  
test\_image = np.expand\_dims(test\_image, axis = 0) #expanding the dimension  
result = model.predict(test\_image) #predicting img  
print(result)  
b=cv2.imread(a)  
cv2\_imshow(b) #Viewing the Image

1/1 [==============================] - 0s 21ms/step  
[[0. 0. 0. 0. 0. 0. 0. 0. 1.]]

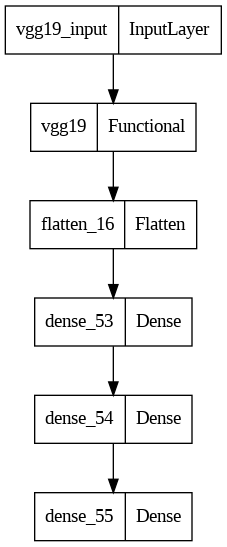


# **VGG 19**

**#Transfer Learning**classifier = models.Sequential()  
pre\_trained\_model = keras.applications.vgg19.VGG19(  
 include\_top=False,  
 weights='imagenet',  
 input\_shape=(200,200,3),  
 pooling="avg",  
 classes=1  
)  
for layer in pre\_trained\_model.layers:  
 layer.trainable = False #Since its a Pretrained Model, we are retaining the weights and top most layers ie. Conv & MaxPool  
classifier.add(pre\_trained\_model)  
classifier.add(keras.layers.Flatten()) #Flattening  
classifier.add(keras.layers.Dense(512, activation = "relu")) #Dense  
classifier.add(keras.layers.Dense(128, activation = "relu")) #Dense  
classifier.add(keras.layers.Dense(9,activation = "softmax")) #Output Layer  
  
classifier.summary() #Model Summary

Model: "sequential\_13"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 vgg19 (Functional) (None, 512) 20024384   
   
 flatten\_16 (Flatten) (None, 512) 0   
   
 dense\_53 (Dense) (None, 512) 262656   
   
 dense\_54 (Dense) (None, 128) 65664   
   
 dense\_55 (Dense) (None, 9) 1161   
   
=================================================================  
Total params: 20,353,865  
Trainable params: 329,481  
Non-trainable params: 20,024,384  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**#Visualizing the Model Architecture**  
from keras.utils import plot\_model  
plot\_model(classifier)



**#Specifying train and test Directory along with Image Resizing**  
train\_datagen1=x\_train\_datagen.flow\_from\_directory("/content/drive/MyDrive/skin\_disease/Train", target\_size=(200,200),batch\_size=128,class\_mode="sparse")  
val\_datagen1=x\_val\_datagen.flow\_from\_directory("/content/drive/MyDrive/skin\_disease/Test", target\_size=(200,200),batch\_size= 128,class\_mode="sparse")

Found 1366 images belonging to 9 classes.  
Found 1366 images belonging to 9 classes.

**#Compiling the Model**  
classifier.compile(optimizer="Adam",loss=keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"])

**#Training the Model**  
h1=classifier.fit(train\_datagen1,epochs=40)

Epoch 1/40  
11/11 [==============================] - 29s 2s/step - loss: 2.2168 - accuracy: 0.1318  
Epoch 2/40  
11/11 [==============================] - 28s 2s/step - loss: 2.1211 - accuracy: 0.1728  
Epoch 3/40  
11/11 [==============================] - 28s 3s/step - loss: 2.0719 - accuracy: 0.2064  
Epoch 4/40  
11/11 [==============================] - 28s 3s/step - loss: 2.0293 - accuracy: 0.2269  
Epoch 5/40  
11/11 [==============================] - 28s 3s/step - loss: 1.9805 - accuracy: 0.2796  
Epoch 6/40  
11/11 [==============================] - 28s 3s/step - loss: 1.9353 - accuracy: 0.2899  
Epoch 7/40  
11/11 [==============================] - 28s 3s/step - loss: 1.9115 - accuracy: 0.2848  
Epoch 8/40  
11/11 [==============================] - 28s 2s/step - loss: 1.8678 - accuracy: 0.3243  
Epoch 9/40  
11/11 [==============================] - 28s 2s/step - loss: 1.8457 - accuracy: 0.3221  
Epoch 10/40  
11/11 [==============================] - 28s 3s/step - loss: 1.8059 - accuracy: 0.3521  
Epoch 11/40  
11/11 [==============================] - 28s 3s/step - loss: 1.7963 - accuracy: 0.3419  
Epoch 12/40  
11/11 [==============================] - 28s 3s/step - loss: 1.7517 - accuracy: 0.3653  
Epoch 13/40  
11/11 [==============================] - 30s 3s/step - loss: 1.7346 - accuracy: 0.3763  
Epoch 14/40  
11/11 [==============================] - 32s 3s/step - loss: 1.7150 - accuracy: 0.3748  
Epoch 15/40  
11/11 [==============================] - 30s 3s/step - loss: 1.7086 - accuracy: 0.3968  
Epoch 16/40  
11/11 [==============================] - 29s 3s/step - loss: 1.6705 - accuracy: 0.4180  
Epoch 17/40  
11/11 [==============================] - 29s 3s/step - loss: 1.6549 - accuracy: 0.4165  
Epoch 18/40  
11/11 [==============================] - 28s 3s/step - loss: 1.6288 - accuracy: 0.4173  
Epoch 19/40  
11/11 [==============================] - 28s 3s/step - loss: 1.6198 - accuracy: 0.4202  
Epoch 20/40  
11/11 [==============================] - 28s 3s/step - loss: 1.5944 - accuracy: 0.4400  
Epoch 21/40  
11/11 [==============================] - 28s 3s/step - loss: 1.5717 - accuracy: 0.4517  
Epoch 22/40  
11/11 [==============================] - 28s 3s/step - loss: 1.5526 - accuracy: 0.4495  
Epoch 23/40  
11/11 [==============================] - 29s 3s/step - loss: 1.5481 - accuracy: 0.4429  
Epoch 24/40  
11/11 [==============================] - 28s 3s/step - loss: 1.5521 - accuracy: 0.4546  
Epoch 25/40  
11/11 [==============================] - 28s 2s/step - loss: 1.5099 - accuracy: 0.4656  
Epoch 26/40  
11/11 [==============================] - 28s 2s/step - loss: 1.5107 - accuracy: 0.4553  
Epoch 27/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4905 - accuracy: 0.4722  
Epoch 28/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4764 - accuracy: 0.4678  
Epoch 29/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4559 - accuracy: 0.4802  
Epoch 30/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4566 - accuracy: 0.4714  
Epoch 31/40  
11/11 [==============================] - 29s 3s/step - loss: 1.4469 - accuracy: 0.4736  
Epoch 32/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4238 - accuracy: 0.4795  
Epoch 33/40  
11/11 [==============================] - 28s 3s/step - loss: 1.3884 - accuracy: 0.5073  
Epoch 34/40  
11/11 [==============================] - 28s 3s/step - loss: 1.3786 - accuracy: 0.5117  
Epoch 35/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4069 - accuracy: 0.5029  
Epoch 36/40  
11/11 [==============================] - 28s 3s/step - loss: 1.4055 - accuracy: 0.4941  
Epoch 37/40  
11/11 [==============================] - 28s 3s/step - loss: 1.3888 - accuracy: 0.4993  
Epoch 38/40  
11/11 [==============================] - 28s 3s/step - loss: 1.3357 - accuracy: 0.5117  
Epoch 39/40  
11/11 [==============================] - 29s 3s/step - loss: 1.3361 - accuracy: 0.5212  
Epoch 40/40  
11/11 [==============================] - 29s 2s/step - loss: 1.3212 - accuracy: 0.5220

**#Evaluating the models Performance**  
classifier.evaluate(val\_datagen1)

11/11 [==============================] - 28s 3s/step - loss: 1.3093 - accuracy: 0.5227

[1.3092535734176636, 0.5226939916610718]

**RESNET**

**#RESNET50; Explination same as before ie. VGG19**classifier = models.Sequential()  
pre\_trained\_model = keras.applications.resnet50.ResNet50(  
 include\_top=False,  
 weights='imagenet',  
 input\_shape=(200,200,3),  
 pooling="avg",  
 classes=1  
)  
for layer in pre\_trained\_model.layers:  
 layer.trainable = False  
classifier.add(pre\_trained\_model)  
classifier.add(keras.layers.Flatten())  
classifier.add(keras.layers.Dense(512, activation = "relu"))  
classifier.add(keras.layers.Dense(128, activation = "relu"))  
classifier.add(keras.layers.Dense(9,activation = "softmax"))  
classifier.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5  
94765736/94765736 [==============================] - 3s 0us/step  
Model: "sequential\_14"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 resnet50 (Functional) (None, 2048) 23587712   
   
 flatten\_17 (Flatten) (None, 2048) 0   
   
 dense\_56 (Dense) (None, 512) 1049088   
   
 dense\_57 (Dense) (None, 128) 65664   
   
 dense\_58 (Dense) (None, 9) 1161   
   
=================================================================  
Total params: 24,703,625  
Trainable params: 1,115,913  
Non-trainable params: 23,587,712  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

classifier.compile(optimizer="Adam",loss=keras.losses.SparseCategoricalCrossentropy(),metrics=["accuracy"])

h1=classifier.fit(train\_datagen1,epochs=10)

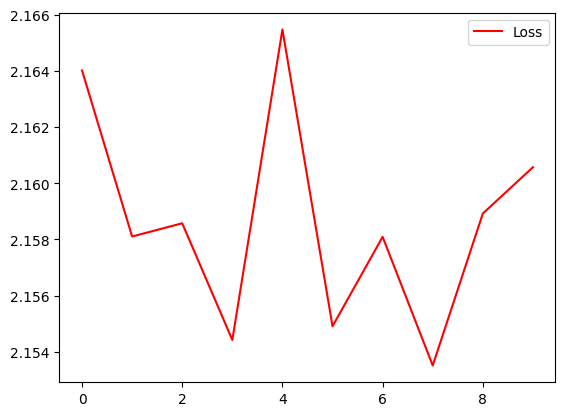
Epoch 1/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1640 - accuracy: 0.1493  
Epoch 2/10  
11/11 [==============================] - 27s 3s/step - loss: 2.1581 - accuracy: 0.1384  
Epoch 3/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1586 - accuracy: 0.1435  
Epoch 4/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1544 - accuracy: 0.1640  
Epoch 5/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1655 - accuracy: 0.1340  
Epoch 6/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1549 - accuracy: 0.1296  
Epoch 7/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1581 - accuracy: 0.1428  
Epoch 8/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1535 - accuracy: 0.1567  
Epoch 9/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1589 - accuracy: 0.1413  
Epoch 10/10  
11/11 [==============================] - 27s 2s/step - loss: 2.1606 - accuracy: 0.1581

classifier.evaluate(val\_datagen1)

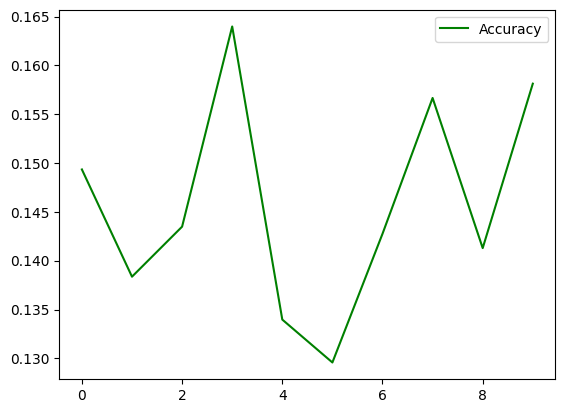
11/11 [==============================] - 35s 3s/step - loss: 2.1538 - accuracy: 0.1493

[2.153787851333618, 0.1493411362171173]

import matplotlib.pyplot as plt  
plt.plot(h1.history["loss"],color="red")  
# plt.plot(h.history["accuracy"],color="green")  
plt.legend(["Loss"],loc="upper right")  
plt.show()



import matplotlib.pyplot as plt  
plt.plot(h1.history["accuracy"],color="green")  
plt.legend(["Accuracy"])  
plt.show()



**Project Implementation:**

The steps in implementation are :

**Data collection:** The first step in implementing a skin disease detection system is to collect a large dataset of skin images. This can be done by partnering with hospitals or clinics and obtaining permission to use their patient records. Alternatively, publicly available datasets such as the International Skin Imaging Collaboration (ISIC) dataset can be used.

**Preprocessing:** Once the dataset is collected, it needs to be preprocessed. This involves tasks such as data cleaning, normalization, and augmentation to ensure that the data is of high quality and can be used for training the AI-DL model.

**Model selection:** There are several AI-DL models that can be used for skin disease detection. For example, convolutional neural networks (CNNs) have been shown to be effective for this task. The choice of model will depend on factors such as the size of the dataset and the complexity of the skin diseases being detected.

**Training:** After selecting the AI-DL model, it needs to be trained on the preprocessed dataset. The training process involves feeding the model the skin images along with their corresponding disease labels. The model then learns to identify patterns in the data that are indicative of different skin diseases.

**Evaluation:** Once the model is trained, it needs to be evaluated to ensure that it is accurate and reliable. This involves testing the model on a separate dataset of skin images and comparing its predictions to the ground truth labels. Various metrics such as accuracy, precision, and recall can be used to evaluate the model's performance.

**Deployment:** Finally, the skin disease detection system can be deployed for use in clinical settings. This involves integrating the AI-DL model into a user-friendly interface that can be accessed by healthcare professionals. The system should also be regularly updated with new data to ensure that it stays accurate and up-to-date with the latest skin diseases and treatments.

**Conclusion:**

The proposed system is able to detect the skin disease with promising results combining computer vision and machine learning techniques. It can be used to help people from all over the world and can be used in doing some productive work. The tools used are free to use and are available for the user, hence, the system can be deployed free of cost. The application developed is light-weight and can be used in machines with low system specifications. It has also a simple user interface for the convenience of the user. The image processing and machine learning algorithms were successfully implemented.

**References:**

Hay, R. J. et al. The global burden of skin disease in 2010: an analysis of the prevalence and impact of skin conditions. *J. Invest. Dermatol.* **134**, 1527–1534 (2014).

Lowell, B. A., Froelich, C. W., Federman, D. G. & Kirsner, R. S. Dermatology in primary care: prevalence and patient disposition. *J. Am. Acad. Dermatol.* **45**, 250–255 (2001).

Awadalla, F., Rosenbaum, D. A., Camacho, F., Fleischer, A. B. Jr & Feldman, S. R. Dermatologic disease in family medicine. *Fam. Med.* **40**, 507–511 (2008).

Feng, H., Berk-Krauss, J., Feng, P. W. & Stein, J. A. Comparison of dermatologist density between urban and rural counties in the United States. *JAMA Dermatol.* **154**, 1265–1271 (2018).

Ramsay, D. L. & Weary, P. E. Primary care in dermatology: whose role should it be? *J. Am. Acad. Dermatol.* **35**, 1005–1008 (1996).

The Distribution of the US Primary Care Workforce (Agency for Healthcare Research &Quality,2012); <https://www.ahrq.gov/research/findings/factsheets/primary/pcwork3/index.html>