S.Kalaiarasi *et al*, International Journal of Computer Science and Mobile Applications,
Vol.6 Issue. 4, April- 2018, pg. 109-118 **ISSN: 2321-8363**
Impact Factor: 5.515

Dermatological Disease Detection using Image Processing and Neural Networks

Mrs. S.Kalaiarasi[1], Harsh Kumar[2], Sourav Patra[3]

[1]*Assistant Professor, Department of CSE, SRM IST, Ramapuram, kalaimaris27.cse@gmail.com*
[2]*Student, Department of CSE, SRM IST, Ramapuram, harsh.singh343@gmail.com*
[3]*Student, Department of CSE, SRM IST, Ramapuram, patrasouravskp@gmail.com*

Abstract

Dermatology is one of the most unpredictable and difficult terrains to diagnose due to its complexity. In most developing countries, it is expensive for a large number of people. According to World Health Organization (WHO), skin diseases are the most common non-communicable diseases in India. The ubiquitous use of smartphones in developing countries like India has opened up new avenues for inexpensive diagnosis of diseases. The camera in smartphones can used to exploit the image processing capabilities of the device for diagnosis. The proposed system deals with the creation of an application that helps in diagnosis of Skin disease. It uses image processing and machine learning technology to detect diseases. The system consists of 2 parts- image processing and the machine learning. The image processing part deals with applying various filters to the images to remove noise and make them uniform. It is necessary to remove the unwanted elements from the image before processing else it will affect the output efficiency. The Machine learning part deals with the processing of data and generation of result.

Keywords*:* Dermatology, Image Processing, Machine Learning.

1. Introduction

This project uses neural networks to achieve the results. Along with this the user interface of the application is developed using tkinter module in python. All of the mentioned processes are done in python. The image processing is done using the 'pillow' and 'openCV' modules where as the machine learning is done using 'Keras' module in python. Artificial neural networks (ANNs) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming. An ANN is based on a collection of connected units or nodes called artificial neurons (a simplified version of biological neurons in an animal brain). Each connection (a simplified version of a synapse) between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

In common ANN implementation, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is calculated by a non-linear function of the sum of its inputs. Artificial neurons and connections typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that only if the aggregate signal crosses that threshold is the signal sent. Keras is an open source neural network library written in Python. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with

image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. The image processing is handled by Pillow and OpenCV toolkits.

Section 2 of this paper explains the existing system present. Section 3 is a brief literature survey of the systems present; Section 4 then explains the system in detail, with the modules and the overarching description and working. Section 5 describes how the implementation is made using the software. Finally, section 6 (conclusion & future work) concludes the paper.

2. Existing System

The existing system relies mostly on the help of doctors (General Physicians and Dermatologists) to examine the condition of the patient with skin disease. Skin conditions that looks like a simple rash but may have indications of underlying serious condition which also may prove fatal if not treated properly. Despite being common it is very difficult to identify most skin disease at an early stage. Most people wait for the natural recovery of the affected region which results in worse situations. Also, children with skin conditions are ignored by their parents for the same reason. The worse condition damages the self-confidence, mental-confidence as well as wellbeing of people. The cost is another reason why people think twice before approaching the doctors is the cost-factor. Hospitals charge extra visitation fee on top of the medical expenditure.

3. Related Works & Literature Survey

Looking into the current situations of computerized skin disease diagnosis systems, there are few solutions available which are still under research developments. Certain limitations and drawbacks are identified in those hence this solution tries to overcome the existing problems with different approach.

The prior publication by Damilola A. Okuboyejo, Oludayo O. Olugbara, and Solomon A. Odunaike, titled ‘Automating Skin Disease Diagnosis Using Image Classification’[1] examines a similar premise. Here a study has been made to design and model a system that uses medical imaging to reduce heavy dependencies on medical expert for diagnosis procedure of Pigmented skin Lesion (PSL) in patients. The methodology of this work is based on soft science design to realize a prototype system for the diagnosis of skin disease represented by a skin image. The intention being the use of feature based on texture analysis and classifies the lesion using techniques such as thresholding and neural networks to develop and prototype a new algorithm for skin disease diagnosis.

In the paper ‘ Skin Disease Diagnosis System using Image Processing and Data Mining’ by R. S. Gound, Priyanka S. Gadre, Jyoti B. Galikwad, Priyanka K. Wagh[2], a system has been proposed an image obtained from the user is processed and segmented to create a model that can predict the disease for a new image of a skin disease. Feature extraction is done on each image to extract features that can be used to create classification model. With this classification model, system finally can predict the disease for a new image of a skin disease which will be obtained by the user through Android application. And based on this predicted disease, system will ask question from the user and based on answer, system will decide disease type. Finally, the system suggests medical treatment or the advice based on predicted skin disease result. The diseases taken into consideration are Eczema, Fungal infection and Urticaria.

Another paper ‘A Image analysis System to Detect Skin Diseases’ by Pravin S. Ambad and A. S. Shirsat[3] presents the image analysis system to detect different skin diseases, where user will be able to take images of different moles or skin patches and the system will analyse and process the image and classifies the image to a variety of skin disease. This system captures image from standard database and put in to the system to inform the user for preventing the threats linked to skin diseases. The system will analyse and process the image and classifies the image to normal, melanoma, psoriasis or dermo case based extraction the image features. An alert will be provided to the user to seek medical help if the mole belongs to the atypical or melanoma category. This methodology also suffers from the issues with segmentation.

4. Method & System Design

The Disease Detection System involves several elements, such as the processing of the image and training of the system using machine learning. It consists of three modules, each of which is involved in a separate "phase"; the overall architecture diagram is shown below.

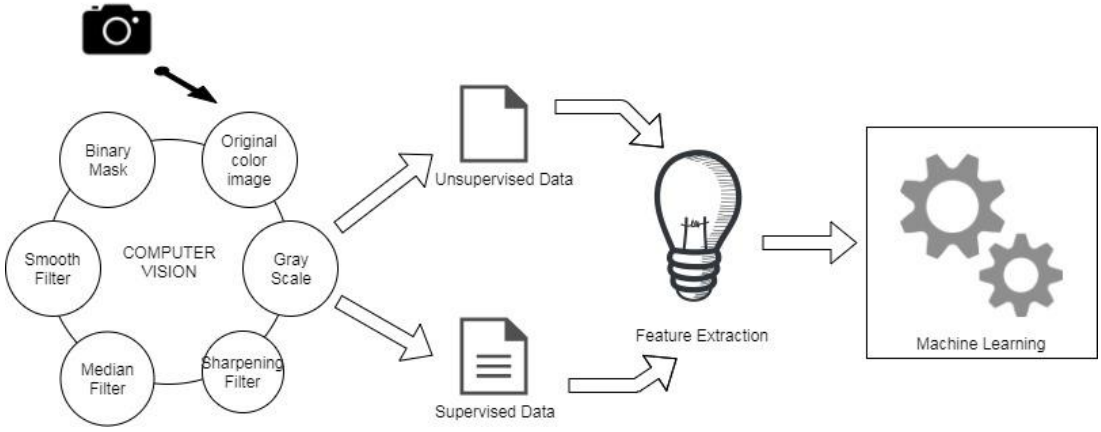

Fig 4.1 The architecture diagram of the system

The **phases** are:

***Data collection**:* The efficiency of a Machine learning model depends on the size of the data set. So, to achieve acceptable efficiency, a large data set is required. The data used in this project are images. The images are downloaded using google images. As we need a large number of images, downloading one image at a time is out of the question here. So, the process is automated using a simple script in python. The scripts searches for the keywords on google images and extracts the raw HTML file. Then all the <image> tags are scrapped and the images are downloaded in a folder. This module provides a high-level interface for fetching data across the World Wide Web.

In particular, the urlopen() function is similar to the built-in function open(), but accepts Universal Resource Locators (URLs) instead of filenames. Some restrictions apply — it can only open URLs for reading, and no seek operations are available. The urllib.request module is used to open or download a file over HTTP. Specifically, the urlretrieve method of this module is what we'll use for actually retrieving the file. To use this method, you need to pass two arguments to the urlretrieve method: The first argument is the URL of the resource that you want to retrieve, and the second argument is the local file path where you want to store the downloaded file. We first import the urllib.request module. Next we create a variable url that contains the path of the file to be downloaded. Finally, we call the urlretrieve method and pass it the url variable as the first argument, destination as second parameter for the file's destination. Keep in mind that you can pass any filename as the second parameter and that is the location and name that your file will have, assuming you have the correct permissions.

Another way to download files in Python is via the urllib2 module. The urlopen method of the urllib2 module returns an object that contains file data. The open method accepts two parameters, the path to the local file and the mode in which data will be written. Here "wb" states that the open method should have permission to write binary data to the given file. You can also download files using requests module. The get method of the requests module is used to download the file contents in binary format. You can then use the open method to open a file on your system, just like we did with the previous method, urllib2.urlopen. With the requests module, you can also easily retrieve relevant meta-data about your request, including the status code, headers

and much more. In the above script, you can see how we access some of this meta-data. One of the simplest way to download files in Python is via wget module, which doesn't require you to open the destination file. The download method of the wget module downloads files in just one line. The method accepts two parameters: the URL path of the file to download and local path where the file is to be stored.

Image processing*:* To make the image suitable for processing it goes through a number of filters and algorithms which makes the images noise free and makes the features well defined. We pre-processed the images and trained the resulted images in our feed forward back-propagation neural network. Eight different types of algorithms were used, they are grey image, sharpening filter, median filter, smooth filter, binary mask, histogram, YCbCr and sobel operator. The algorithms were implemented sequentially. In photography, computing, and colorimetry, a grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Images of this sort, also known as black-and-white or monochrome, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only two colors, black and white (also called bilevel or binary images). Grayscale images have many shades of gray in between. Graphics programs can be used to both sharpen and blur images in a number of ways, such as unsharp masking or deconvolution. Portraits often appear more pleasing when selectively softened (particularly the skin and the background) to better make the subject stand out. This can be achieved with a camera by using a large aperture, or in the image editor by making a selection and then blurring it. Edge enhancement is an extremely common technique used to make images appear sharper, although purists frown on the result as appearing unnatural. Another form of image sharpening involves a form of contrast. This is done by finding the average color of the pixels around each pixel in a specified radius, and then contrasting that pixel from that average color. This effect makes the image seem clearer, seemingly adding details.

An example of this effect can be seen to the right. It is widely used in the printing and photographic industries for increasing the local contrasts and sharpening the images. The median filter is a nonlinear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise (but see discussion below), also having applications in signal processing. A binary image is a digital image that has only two possible values for each pixel. Typically, the two colors used for a binary image are black and white. The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

In the document-scanning industry, this is often referred to as "bi-tonal". Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit—i.e., a 0 or 1. The names black-and-white, B&W, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images. In Photoshop parlance, a binary image is the same as an image in "Bitmap" mode. YCbCr, Y′CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y'CBCR, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems.

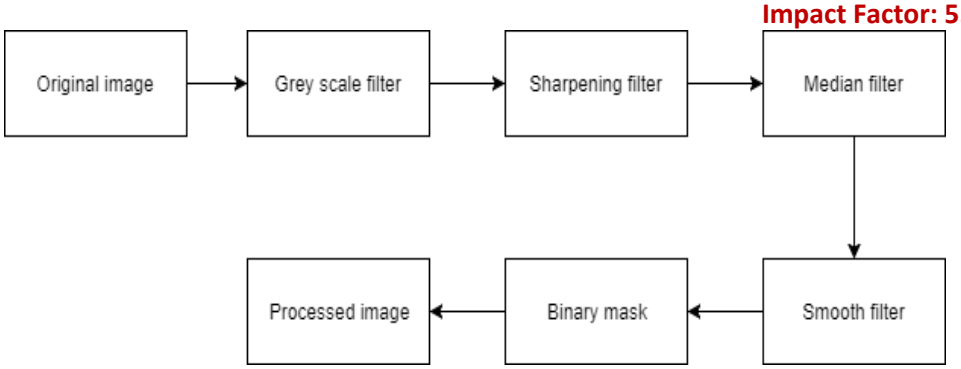

Fig 4.2 The work-flow diagram of Image Processing module of the system

Machine learning*:* To use the system, the machine needs to be trained and it takes a lot of time to do it. This step is done individually so that the user doesn't have to deal with the complexity of the system and they don't have to train the model every time before use. Machine learning is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs.

Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank, and computer vision. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioural profiles for various entities and then used to find meaningful anomalies.

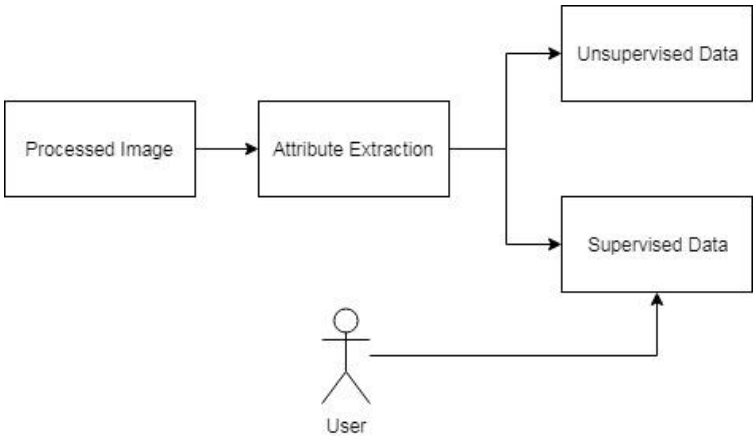

Fig 4.3 The work-flow diagram of the Machine Learning module of the system

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables. The model is trained using a large data set of images of different skin diseases using Artificial Neural Network and saved as a .h5 file.

User Interface*:* The user interface is developed using the tkinter module in python. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with the standard Microsoft Windows and Mac OS X install of Python. The name Tkinter comes from Tk interface.

Tkinter was written by Fredrik Lundh. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application. Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter).

5. Implementation Details

The platform used in this project is a computer running on Windows operating system with Python 3.6 installed. All the necessary modules (Keras, Tkinter, OpenCV, Pillow) are installed using Pip installer. The machine learning model is created beforehand using Keras module. The model is saved in a .h5 file. This model is again loaded in the user interface module which will produce the result in reference to this model.

5.1 Simulation parameters

The simulation is done on a windows laptop from Dell. It has an Intel® Core™ i5-2450M CPU which has a clock speed of 2.50Hz. The installed memory (RAM) is 8.00 GB with a 64-bit operating System. As discussed in previous section, Python 3.6 platform has been used for running the software. The Python IDLE Integrated Development Platform (IDE) has been used. For the training and the testing set, jpeg images have been used which are downloaded from the internet.

5.2 The Data collection phase

For the collection of the data-set a python Script has been used that is hosted on Github. The Script searches over the internet and downloads any images with the same name as provided as a parameter. This program is compatible with both the versions of python 2.x and 3.x. It can be downloaded from github and can be executed with no change to the file.

To install the program, a module named '*google_images_download'* can be installed that directly saves the program as a library file. It can also be run manually by cloning/downloading the project. Once downloaded, the program can be run from the command line interface with the required images to download as parameters.

5.3 The Image Processing phase

The image processing phase involves the extraction of features from the input images (from the data-set) using the python OpenCV module. Different functions have been used to change the input step by step to the optimum image with only the necessary features. Firstly, the image is converted to a grey scale image as luminance is the required part that represents all the features present in the image. Also since the colour of the infected region is the same for more or less every skin disease being tested, we do not need to consider the chrominance of the image.

Next we apply a sharpening filter function to the image to increase the contrast of the edge mask of the image. Subsequently we apply some noise reduction filtering functions like the median filter and smooth filter. The sobel operator is used for edge detection.

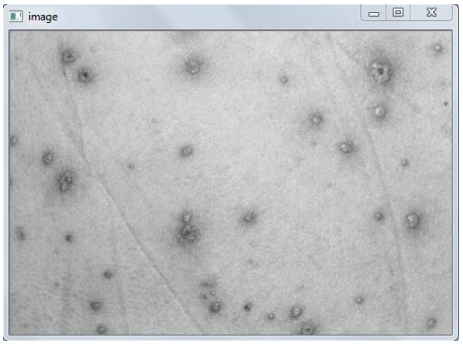

Figure 5.1 Image to grey scale

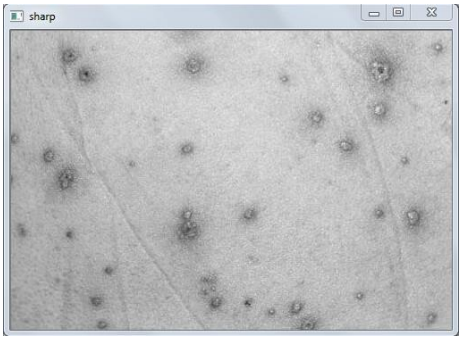

Figure 5.2 Applying sharpening filter

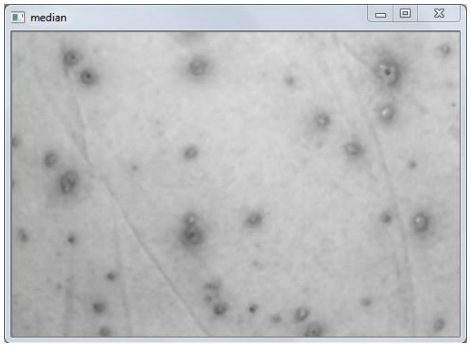

Figure 5.3 Applying median filter

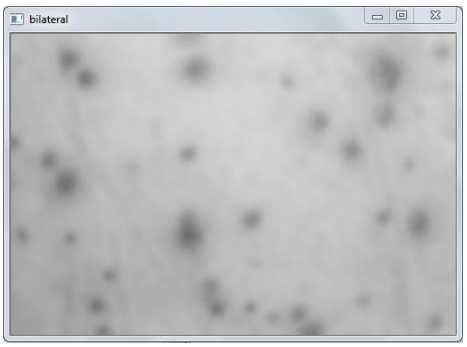

Figure 5.4 Applying bilateral filters

5.4 The Machine Learning phase

Finally, in the machine learning phase the feature-rich images are feed into the training set. For the implementation purpose we have used the Keras library in python to run the deep learning algorithm. Deep learning is a subfield of the machine learning inspired by the functioning of the brain. The whole model consists of layers, just like neurons in the brain. This model learns by considering examples. The sequential model is created using Keras. Initially two layers are added with a rectifier function. The next step is adding a pooling layer of 2x2 matrix. Now the pooled images are converted into a continuous vector. The hidden layer is now added, this layer will contain all the nodes, called artificial neurons. These layers simulate the human brain by learning layers by layers. Once all the layers are added, the model is compiled and saved as a h5 file which can be loaded on the front end so that we don't have to train the model every time.

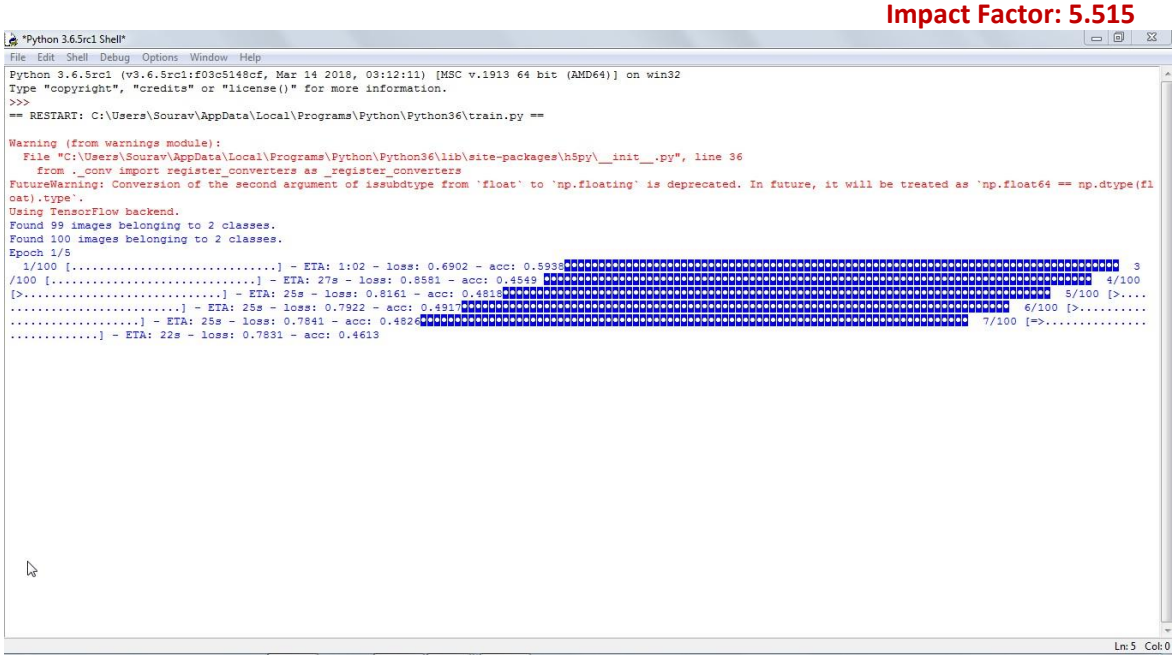

Figure 5.5 System under Training phase - I

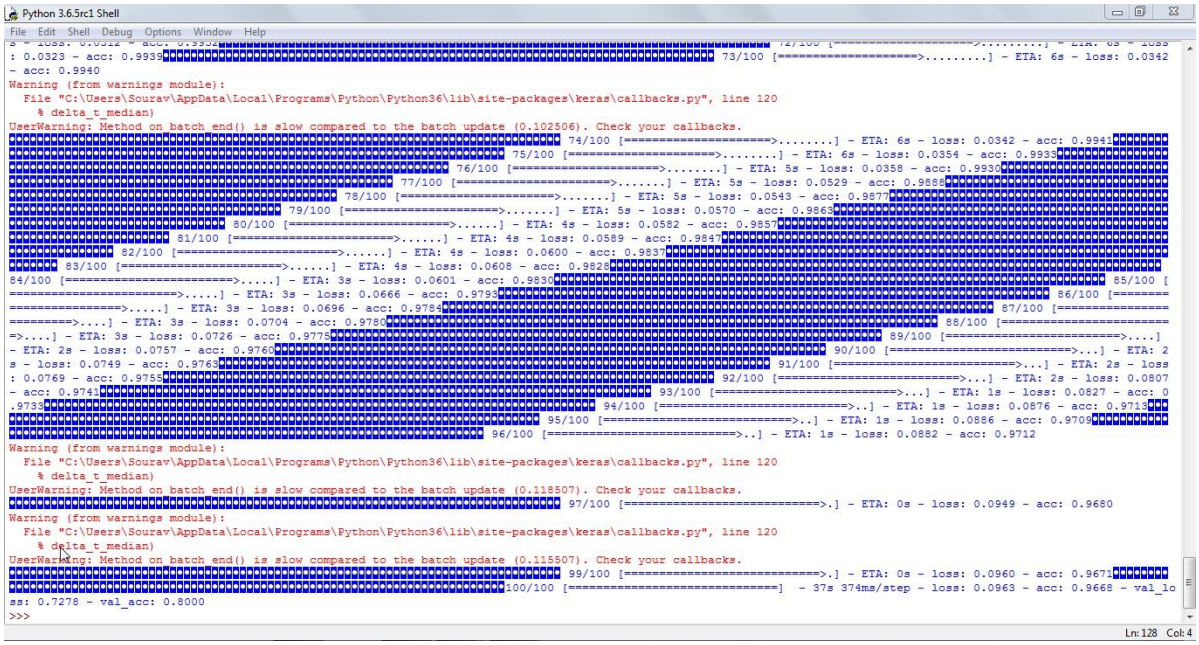

Figure 5.5 System under Training phase – II

5.5 The User Interface

As discussed above, the user interface is made using tkinter module in Python. The User interface consists of a button which will open file explorer to select the image to be analysed. After selection, the image is shown on

the user interface to clarify whether the right image has been selected or not. To show the result, there is one another button which once pressed will analyse the image and display the output in the user interface.

Figure 5.6 Image of Ringworm imported to the user interface

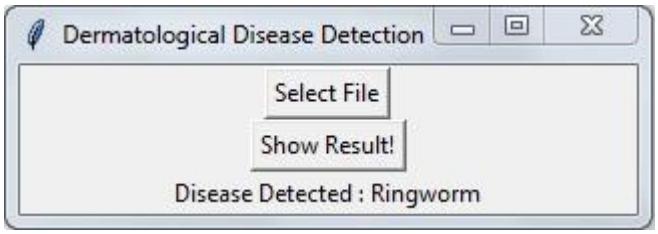

Figure 5.7 The User Interface displays the detected disease as Ringworm

6. Conclusion

The proposed system is able to successfully detect the dermatological disease present in the image. 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. Though the machine learning data-set was small, the system was able to identify the disease with minimum error. 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 deep learning algorithms were successfully implemented. The diseases on which the system is implemented on are Nevus and Ringworms. Sufficient testing has been done, taking a lot of images for testing purpose.

The future scope of the project has a lot of possibilities. Starting from the user interface; it can be readily improved from feedback from the users. Another improvement would be to increase the number of images in the data-set for the better learning of the system since the efficiency of deep learning algorithm increases with bigger data-sets. The data-set can also be made from the input image given by the users. The images can be stored in a database that when labelled correctly and be used for supervised learning or unsupervised learning when used without labels. Also currently the system has been trained on only a few diseases that can be increased to many more in the future as and when the data-set of the particular disease is available.

References

[1] Okuboyejo, Damilola A, et al. "Automating Skin Disease Diagnosis Using Image Classification." *Proceedings of WCECS 2013, October 23 - 25, 2013, San Francisco, USA, IAENG Open Access Publication*, 24 Oct. 2013, www.iaeng.org/publication/WCECS2013/.

[2] Ajith, Archana, et al. "Dermatological Disease Detection Using Image Processing and Artificial Neural Network." *Dermatological Disease Detection Using Image Processing and Artificial Neural Network - IEEE Conference Publication*, 15 June 2015, ieeexplore.ieee.org/document/7026918/.

[3] Ambad, Pravin S., and A. S. Shirat. "A Image Analysis System to Detect Skin Diseases." Index of /Iosr-Jvlsi/Papers/vol6-issue5/Version-1, 1 Oct. 2016, www.iosrjournals.org/iosr-jvlsi/papers/vol6-issue5/Version-1/.

[4] Gound, R. S., et al. "Number 16 (ISBN: 973-93-80898-01-6)." *IJCA - Number 16 (ISBN: 973-93-80898-01-6)*, 16 Jan. 2018, www.ijcaonline.org/archives/volume179/number16/.

[5] Kumar, Vinayshekhar Bannihatti, et al. "Dermatological Disease Detection Using Image Processing and Machine Learning." *Dermatological Disease Detection Using Image Processing and Machine Learning - IEEE Conference Publication*, 20 Sept. 2016, ieeexplore.ieee.org/document/7585217/.

[6] Yadav, Prashant B., and S. S. Patil. *Dermatological Disease Diagnosis Using Color-Skin Images - IEEE Conference Publication*, ieeexplore.ieee.org/document/6359626/

[7] "OpenCV Library." *OpenCV Library*, opencv.org/.

[8] "Keras: The Python Deep Learning Library." *Keras Documentation*, keras.io/.

