

A project abstract

on

**A Comparative Study On Skin
Cancer Detection Using Transfer
Learning Models and CNN**

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

by

Shaik Umme Honey

184G1A05A7

Mekala Raviteja

184G1A0564

Kavali Swathi

184G1A05A3

Under the Guidance of

Mr.Sreedhar Boya M.Tech,M.B.A

Assistant Professor



Department of Computer Science & Engineering

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUA & Approved by AICTE) (Accredited by NAAC with 'A' Grade & Accredited by NBA (EEE, ECE & CSE))

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu – 515701.

2021 - 2022

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUA & Approved by AICTE)

(Accredited by NAAC with 'A' Grade & Accredited by NBA (EEE, ECE & CSE)

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu-515701.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



Certificate

This is to certify that the Project report entitled A **COMPARATIVE STUDY ON SKIN CANCER DETECTION USING TRANSFER LEARNING MODELS AND CNN** is the bonafide work carried out by **Shaik Umme Honey** bearing Roll Number **184G1A05A7**, **Mekala Raviteja** bearing Roll Number **184G1A0564**, **Kavali Swathi** bearing Roll Number **184G1A05A3** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2021- 2022.

Signature of the Guide

Mr.B.Sreedhar M.Tech,MBA

Assistant Professor

Head of the Department

Mr.P.Veera Prakash, M.Tech, (Ph.D)

HOD & Assistant Professor

Date:

Place: Rotarypuram

EXTERNAL EXAMINER

DECLARATION

We, Ms Shaik Umme Honey having reg no:184g1a05A7, Mr Mekala Raviteja having reg no:184g1a0564, Ms Kavali Swathi having reg no:184g1a05A3 students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that dissertation entitled “COMPARATIVE STUDY ON SKIN DISEASE” embodies the report of our project work carried out by us during IV year Bachelor of Technology in Mr.B.Sreedhar,M.Tech,MBA,,Assistant Professor, Department of CSE, SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY,ANANTAPUR and this work has been submitted for the partial fulfilment of the requirements for the award of the Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Shaik Umme Honey

Reg no :184G1A05A7

Mekala Raviteja

Reg no:184G1A0564

Kavali Swathi

Reg no:184G1A05A3

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that I would like to express my indebted gratitude to my Guide **Mr.B.Sreedhar,M.Tech,MBA Assistant Professor, Computer Science & Engineering**, who has guided me a lot and encouraged me in every step of this work. I thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this Project Work.

I am very much thankful to **Mr.P.Veera Prakash, M.Tech, (Ph.D), Assistant Professor & Head of the Department, Computer Science & Engineering**, for his kind support and for providing necessary facilities to carry out the work.

I wish to convey my special thanks to **Dr. G. Bala Krishna, Ph.D., Principal of Srinivasa Ramanujan Institute of Technology** for giving the required information in doing my project work. Not to forget, I thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my technical seminar in time.

I also express our sincere thanks to the Management for providing excellent facilities.

Finally, I wish to convey my gratitude to my family who fostered all the requirements and facilities that I need.

Project Associates

CONTENTS	Page No
List of Figures	V
Abbreviations	VI
List of Screens	VII
Abstract	IX
Chapter 1 Introduction	1
Chapter 2 Literature Review	4
Chapter 3 System Analysis and Feasibility Study	9
3.1 Existing Method	9
3.2 Proposed System	9
3.3 Block Diagram	9
3.4 Architecture Of Proposed Method	10
Chapter 4 Methadology and Algorithms	11
4.1 Convolutional Neural Network	11
4.2 ResNet50	14
4.3 Xeception Architecture	14
4.3.1 Architecture Of Xception Model	15
4.3.2 Depth Wise and PointWise Convolution	15
Chapter 5 Software Development Life Cycle	16
5.1 Feasibility Study	17
5.1.1 Economic feasibility	17
5.1.2 Social feasibility	18
5.1.3 Technical feasibility	18

Chapter 6	System Requirements Specification	19
	6.1 Functional requirements	19
	6.2 Non-functional requirements	19
	6.3 System specifications	20
Chapter 7	System Design	21
	7.1 Input design	21
	7.2 Output design	22
	7.3 Modules	22
Chapter 8	UML Diagram	23
	8.1 Use case diagram	24
	8.2 Class diagram	25
	8.3 Sequence diagram	25
	8.4 Collaboration diagram	26
	8.5 Deployment diagram	26
	8.6 Component diagram	26
	8.7 Activity diagram	27
Chapter 9	ER and Data Flow Diagrams	28
	9.1 ER Diagram	28
	9.2 DFD Diagram	29
Chapter 10	Execution Results	30
Chapter 11	Bibliography	35
	11.1 Installing python	35
	11.2 Installing Pycharm	35
	11.3 Introduction to python	37
	11.4 System Testing	51

Conclusion	55
References	56

List of Figures

Fig. No.	Figure Name	Page No
Fig. 3.1	Block diagram of proposed method	9
Fig. 3.4	Architecture of proposed method	10
Fig. 4.1.1	Convolution Operation	11
Fig. 4.1.2	Convolution Neural Networks Scan Images	12
Fig. 4.1.4	Model Architecture of Proposed CNN Model	13
Fig. 4.2.1	ResNet50 Architecture	14
Fig. 4.3.1	Architecture Of Xception Model	15
Fig. 4.3.2	Depth Wise and PointWise Convolution	15
Fig. 5.1	Waterfall Model	16
Fig. 8.1	Use Case Diagram	24
Fig. 8.2	Class Diagram	25
Fig. 8.3	Sequence Diagram	25
Fig. 8.4	Collaboration Diagram	26
Fig. 8.5	Deployment Diagram	26
Fig. 8.6	Component Diagram	26
Fig. 8.7	Activity Diagram	27
Fig. 9.1	ER Diagram	28
Fig. 9.2	Data Flow Diagram	29

LIST OF ABBREVIATIONS

CNN	Convolutonal neural network
CT	Computed tomography
DSA	Digital subtraction angiography
MR	Magnetic resonance imaging
PDE	Partial differential equation
SVM	Support vector machine
PSO	Psoriasis
ECZ	Eczema
AD	Atopic dermatitis
EEG	Electro encephalogram
API	Artificial programming interface
PC	Personal computer
MICR	Magnetic ink character recognition
OMR	Optical mark recognition
UML	Unified modelling language
ER	Entity-relationship
DBMS	Database management system
DFD	Data flow diagram
PERL	Practical extraction and report language
PHP	Hypertext preprocessor

GPL	General public license
GUI	Graphical user interface
MFL	Memory function complete
DIL	Python imaging library
XML	Extensible markup language
HTML	Hypertext markup language
SGML	Standard generalized markup language
SQL	Structured query language
SAX	Simple api and xml
DOM	Document object model
HTTP	Hypertext transfer protocol
CSRF	Cross-site request forgery
URL	Uniform resource locator
JSON	Java script object notation
ORM	Object- relational mapping

LIST OF SCREENS

S.No	Screens	Pg.No
10.1	Home	30
10.2	About Project	30
10.3	User Home	30
10.4	Upload Image	31
10.5	Actintic keratosis Image	31
10.6	Basal cell carcinoma	31
10.7	Benign keratosis -like lesions	32
10.8	Dermatofibroma	32
10.9	Melanoma	32
10.10	Vascular Lesions	33

ABSTRACT

Dermatological diseases are the most prevalent diseases worldwide. Even though being common, diagnosis is extremely difficult and requires extensive experience in the domain. In this project, we provide an approach to detect various kinds of these diseases. Computer vision and Machine learning are dual stages which we used for identify diseases accurately. Our objective of the project is to detect the type of skin disease easily with accuracy and recommend the best. First stage of the image the skin disease is subject to various kinds of pre-processing techniques followed by feature extraction. Then the second stage involves it uses the Machine learning algorithms to identify diseases based on the analysing and observance of the skin. The proposed system is highly beneficial in rural areas where access to dermatologists is limited. For this proposed system, we use Pycharm based python script for experimental results.

Our objective of the project is to detect the type of skin disease easily with accuracy and recommend the best and global medical suggestions. skin disease much more quickly and accurately. But the cost of such diagnosis is still limited and very expensive. So, image processing techniques help to build automated screening system for dermatology at an initial stage. The extraction of features plays a key role in helping to classify skin diseases. Computer vision has a role in the detection of skin diseases in a variety of techniques. Due to deserts and hot weather, skin diseases are common in Saudi Arabia. This work contributes in the research of skin disease detection.

Keywords: Ham10000, image processing, CNN, ResNet50, Xception, Skin Disease

CHAPTER 1

INTRODUCTION

Skin diseases are more common than other diseases. Skin diseases may be caused by fungal infection, bacteria, allergy, or viruses, etc. A skin disease may change texture or color of the skin. In general, skin diseases are chronic, infectious and sometimes may develop into skin cancer. Therefore, skin diseases must be diagnosed early to reduce their development and spread. The diagnosis and treatment of a skin disease takes longer time and causes financial and physical cost to the patient. In general, most of the common people do not know the type and stage of a skin disease. Some of the skin diseases show symptoms several months later, causing the disease to develop and grow further. This is due to the lack of medical knowledge in the public.

Sometimes, a dermatologist (skin specialist doctor) may also find it difficult to diagnose the skin disease and may require expensive laboratory tests to correctly identify the type and stage of the skin disease. The advancement of lasers and photonics based medical technology has made it possible to diagnose the skin diseases much more quickly and accurately. But the cost of such diagnosis is still limited and very expensive. Therefore, we propose an image processing-based approach to diagnose the skin diseases. This method takes the digital image of disease effect skin area then use image analysis to identify the type of disease. Our proposed approach is simple, fast and does not require expensive equipment's other than a camera and a computer.

Composed of epidermis, dermis, and subcutaneous tissues, skin is the largest organ of human body, containing blood vessels, lymphatic vessels, nerves, and muscles, which can perspire, perceive the external temperature, and protect the body. Covering the entire body, the skin can protect multiple tissues and organs in the body from external invasions including artificial skin damage, chemical damage, adventitious viruses, and individuals' immune system. Besides, skin can also avoid the loss of lipids together with water within epidermis and dermis so that skin barrier function can be stabilized. In spite of defense and barrier function, skin is not indestructible in that skin tends to be constantly influenced by a variety of external and genetic factors.

Currently, there are three main types of skin diseases appearing in human body, including viral skin diseases, fungal skin diseases, and allergic skin disease. Despite the fact that these types of skin diseases can be cured at present, these diseases indeed have brought trouble to patients' life. Nowadays, the majority of conclusions on the patients' existing symptoms are drawn mainly based on doctors' years of experience or their own subjective judgments, which may lead to misjudgements and consequently delay the treatment of these.

Therefore, it is of great theoretical significance and practical value to study how to extract symptoms of diverse skin diseases on the basis of modern science and technology. Under this circumstance, effective and accurate identification of the types of skin diseases can be achieved to prescribe treatment according to patients' symptoms. Over the past few years, the image processing technique has achieved rapid development in medicine.

Some equipment based on digital image technology has also been widely applied to people's everyday life, for instance, computed tomography (CT), digital subtraction angiography (DSA), and magnetic resonance imaging (MRI). Deeper research on this direction has been carried out by scholars all over the world.

For example, the skin disease varicella was detected by Oyola and Arroyo through image processing technique's color transformation, equalization as well as edge detection, and the image of varicella was eventually collected and classified through Hough transform. The final empirical results demonstrated that a better diagnosis was received in terms of detection on varicella, and preliminary test was also conducted on varicella and herpes zoster on that basis. Chung and Sapiro put forward a method to detect the image of skin lesions based on partial differential equation (PDE), with which a contour model of skin lesions was extracted on the basis of its morphological filtering through PDE.

Skin diseases occur commonly among humans. They are usually caused by factors like different organism's cells, a different diet, and internal and external factors, such as the hierarchical genetic group of cells, hormones, and immune system of conditions. These factors may act together or in a sequence of skin disease. There are chronic and incurable diseases, like eczema and psoriasis, and malignant diseases like malignant melanoma.

Recent researchers have found the availability of cures for these diseases if they are detected in the early stages. Atopic dermatitis, commonly called eczema, is a long-term skin disease whose common symptoms are dry and itchy skin, rashes on the face, inside the elbows, behind the knees, and on the hands and feet. Melanoma is severe and life-threatening skin cancer. The "ABCD's" of moles detected on the skin are Asymmetry, Border, Colour, and Diameter. Asymmetry implies that the shape of one half does not match the other half. Border means the edges of the mole are ragged, blurred, or irregular.

CHAPTER 2

LITERATURE REVIEW

[1] **Yasir, R., Rahman, M. A., & Ahmed, N.** Skin diseases are among the most common health problems worldwide. In this article we proposed a method that uses computer vision based techniques to detect various kinds of dermatological skin diseases. We have used different types of image processing algorithms for feature extraction and feed forward artificial neural network for training and testing purpose.

The system works on two phases- first pre-process the colour skin images to extract significant features and later identifies the diseases. The system successfully detects 9 different types of dermatological skin diseases with an accuracy rate of 90%. Dermatology is the branch of medicine dealing with the hair, nails, skin and its diseases, It is a specialty with both medical and surgical aspects A dermatologist takes care of diseases, in the widest sense, and some cosmetic problems of the skin, scalp, hair, and nails. Human skin is one of the most unpredictable and difficult terrains to automatically synthesize and analyze due to its complexity of jaggedness, tone, presence of hair and other mitigating features. In a developing country like Bangladesh it is expensive for a large number of people to go to dermatologist for their skin disease problem.

Every year a large number of populations in the developing countries like Bangladesh suffer due to different types of skin diseases. So, it is very necessary for both the patients and dermatologists to have an automated skin disease detection system especially in developing countries. Even though there have been several researches conducted to detect dermatological skin diseases using Computer Vision based techniques but almost every one worked for only 2-3 diseases. In our work we have worked to detect 9 different types of skin diseases. They are Eczema, Acne, Leprosy, Psoriasis, Scabies, Foot Ulcer, Vitiligo, Tinea Corporis and Pityriasis Rosea. We have used 8 different types of algorithms for image preprocessing (YCbCr, grey image, sharpening filter, median filter, smooth filter, binary mask, histogram and sobel operator).

Summary:

An automated system for detecting various kinds of skin diseases. At first it was not easy for us as we were trying to analyse human skin characteristics since human skin is one of the most difficult surface to analyse. It is unique and novel since our dataset was not acquired

from secondary sources; rather it was a work of months of toil in an actual hospital of Bangladesh making the dataset a standard dataset which is completely new in perspective of Bangladesh. The system examines an image of infected human skin and detects the disease with an accuracy rate of 90%. We have tested a total number of 775 skin images for 9 diseases.

[2] ALKolifi ALEnezi, N. S.: Skin diseases are more common than other diseases. Skin diseases may be caused by fungal infection, bacteria, allergy, or viruses, etc. The advancement of lasers and Photonics based medical technology has made it possible to diagnose the skin diseases much more quickly and accurately. But the cost of such diagnosis is still limited and very expensive. So, image processing techniques help to build automated screening system for dermatology at an initial stage. The extraction of features plays a key role in helping to classify skin diseases.

Computer vision has a role in the detection of skin diseases in a variety of techniques. Due to deserts and hot weather, skin diseases are common in Saudi Arabia. This work contributes in the research of skin disease detection. We proposed an image processing-based method to detect skin diseases. This method takes the digital image of disease effect skin area, then use image analysis to identify the type of disease. Our proposed approach is simple, fast and does not require expensive equipment other than a camera and a computer. The approach works on the inputs of a color image.

Then resize of the image to extract features using pretrained convolutional neural network. After that classified feature using Multiclass SVM. Finally, the results are shown to the user, including the type of disease, spread, and severity. The system successfully detects 3 different types of skin diseases with an accuracy rate of 100%. In general, most of the common people do not know the type and stage of a skin disease. Some of the skin diseases show symptoms several months later, causing the disease to develop and grow further. This is due to the lack of medical knowledge in the public.

Sometimes, a dermatologist (skin specialist doctor) may also find it difficult to diagnose the skin disease and may require expensive laboratory tests to correctly identify the type and stage of the skin disease. The advancement of lasers and photonics based medical technology has made it possible to diagnose the skin diseases much more quickly and

accurately. But the cost of such diagnosis is still limited and very expensive. Therefore, we propose an image processing-based approach to diagnose the skin diseases. This method takes the digital image of disease affected skin area then use image analysis to identify the type of disease. Our proposed approach is simple, fast and does not require expensive equipment's other than a camera and a computer.

Summary:

Detection of skin diseases is a very important step to reduce death rates, disease transmission and the development of the skin disease. Clinical procedures to detect skin diseases are very expensive and time-consuming. Image processing techniques help to build automated screening system for dermatology at an initial stage. The extraction of features plays a key role in helping to classify skin diseases. In this research the method of detection was designed by using pre-trained convolutional neural network (AlexNet) and SVM. In conclusion, we must not forget that this research has an effective role in the detection of skin diseases in Saudi Arabia because it has a very hot weather for the presence of deserts; this indicates that skin diseases are widespread. This research supports medical efficiency in Saudi Arabia.

[3] Wu, H., Yin, H., Chen, H., Sun, M., Liu, X., Yu, Y. Lu, Q.: Inflammatory skin diseases refer to the skin disorder involving inflammatory cell infiltration and inflammatory cytokine dramatically elevated. Inflammatory skin diseases affect more than 1/5 human population in the worldwide. Inflammatory skin diseases include psoriasis (Pso), eczema (Ecz), and atopic dermatitis (AD), etc. Dermatologists usually diagnose these diseases by “first impression” and then follow pathological analysis and laboratory tests to confirm the “first impression”. However, less experienced dermatologists and young dermatologists are particularly susceptible to errors since Pso, Ecz and AD are very easily to be misdiagnosed.

To solve this problem and assist dermatologists, in this study, we developed an end-to-end deep learning model, which is based on clinical skin images, for automated diagnosis of Pso, Ecz and AD. Convolutional neural networks (CNNs) have shown great power for the analysis of clinical images, and an increasing number of studies have reported promising results for CNNs in a variety of diseases. For example, CNNs have been applied to assist in the early diagnosis and detection of Alzheimer disease from brain electroencephalogram (EEG) spectral images and MRIs, to predict the risk of osteoarthritis from knee cartilage MRIs, for segmentation of multiple sclerosis lesions from multichannel 3D MRIs, to

diagnosis breast nodules and lesions from ultrasound images, and for the detection of diabetic retinopathy from retinal fundus photographs.

The development of CNNs in dermatology stem from the development of pioneering technologies for assisting in melanoma diagnosis. There are now multiple examples of AI tools facilitating cancer diagnosis based on data input from dermoscopes and from histological images of skin biopsy tissues. However, to the best of our knowledge, we are unaware of any applications of AI tools to assist in diagnosing skin diseases other than cancers.

Summary: AI based smartphone medical application is occurred in 2017 by Bain and his colleagues. They used a smartphone's camera to visually identify the patients with schizophrenia, the drug and confirm ingestion via a neural network computer vision algorithm. Another example is in the field of diabetes, in which Shao and his colleagues applied smartphone to control engineered cells to maintain semiautomatic glucose homeostasis in diabetic mice, shedding a light on the application of smartphone platform in management of chronic diseases.

[4] T. Swapna¹ , D.A. Vineela , M. Navyasree , N. Sushmtha ,P. Bhavana: Skin disease among humans has been a common disease, millions of people are suffering from various kinds of skin diseases. Usually, these diseases have hidden dangers which lead to not only lack of self-confidence and psychological depression but also lead to a risk of skin cancer. Medical experts and high-level instruments are needed to diagnosis these skin diseases due to non-availability of visual resolution in skin disease images.

The proposed framework includes deep learning techniques such as CNN architecture and three predefined models called Alex Net, ResNet, InceptionV3. A Dataset of images with seven diseases has been taken for the Classification of Skin diseases. They include diseases like Melanoma, Nevus, Seborrheic Keratosis etc. The dataset was extended by adding images having cuts and burns, which were classified as skin disease by most of the existing systems.

The usage of Deep Learning algorithms has reduced the need for human labor, such as manual feature extraction and data reconstruction for classification purposes. Skin is one in every of the most important and quickest developing tissues of the human body. The burden of skin disease is regarded as a multidimensional concept that comprehends psychological,

social and economic significance of the skin disease at the sufferers and their households and on society. It is a contamination that takes place in humans of all ages. Skin is regularly broken due to the fact it's far a touchy a part of the body. There are more than 3000 skin diseases. A cosmetically look spoiler disease will have a big effect and might reason extensive ache and everlasting injury. Most of the chronic skin conditions, along with atopic eczema, psoriasis, vitiligo and leg ulcers, aren't right now deadly, they may be diagnosed as an extensive problem on fitness popularity which include physical, emotional and economic outcome. On the other hand, skin cancers are potentially lethal and their trouble is associated with the temporality that they carry.

Summary:

The proposed system is to classify the given skin image as 'Not a Skin disease' or one of the seven diseases namely Warts Molluscum, Systemic Disease, Seborrheic Keratosis, Nevus, Bullous, Actinic Keratosis, Acne and Rosacea using deep learning techniques. The dataset is divided into training and testing and deep learning models are build using CNN and three pretrained networks called Alex Net, ResNet, InceptionV3. The train and test accuracies of The train and test accuracies of CNN, Resnet, Alexnet and Inception-v3 are also shown in the bar graph .From the graph it is observed that CNN has outperformed over training Data but not on test data so it has lead to overfitting of the model. Resnet Train and test accuracies are better compared to the other models. So Resnet architecture can be used for detecting and prediction of skin diseases

CHAPTER 3

SYSTEM ANALYSIS & FEASIBILITY STUDY

3.1 Existing Method:

This model emphasizes an existing method that which is designed using the some of the algorithms of deep learning. Here the process is performed using the ANN methods, which is one of the Machine Learning methods, but this could not get the high accuracy.

Disadvantages:

- Less feature compatibility
- Low accuracy

3.2 Proposed System:

In purposed method we are performing the classification of either the Skin Disease identification using Convolution Neural Network (CNN) of deep learning along with the transfer learning methods. As image analysis based approaches for skin disease classification. Hence, proper classification is important for the proper nutrition that which will be possible by using our proposed method. Block diagram of proposed method is shown below.

3.3 Block Diagram

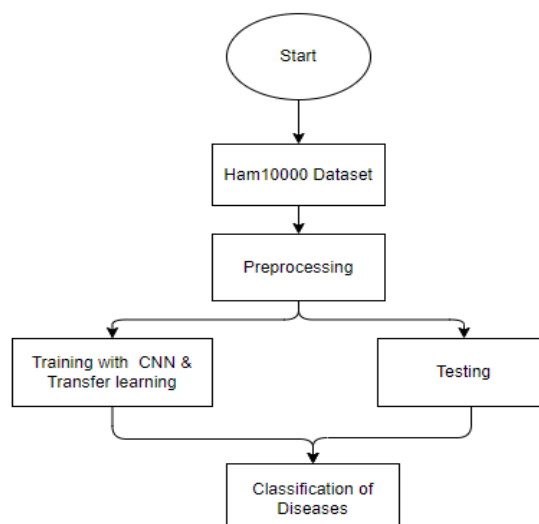
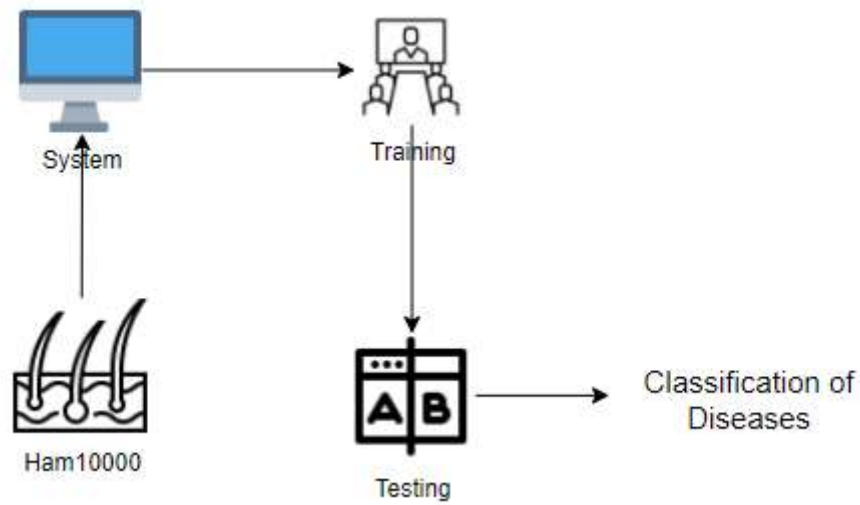


Fig 3.1. Block diagram of proposed method

3.4 ARCHITECTURE OF PROPOSED METHOD

**Advantages:**

- Accurate classification
- Less complexity
- High performance
- Easy Identification

CHAPTER 4

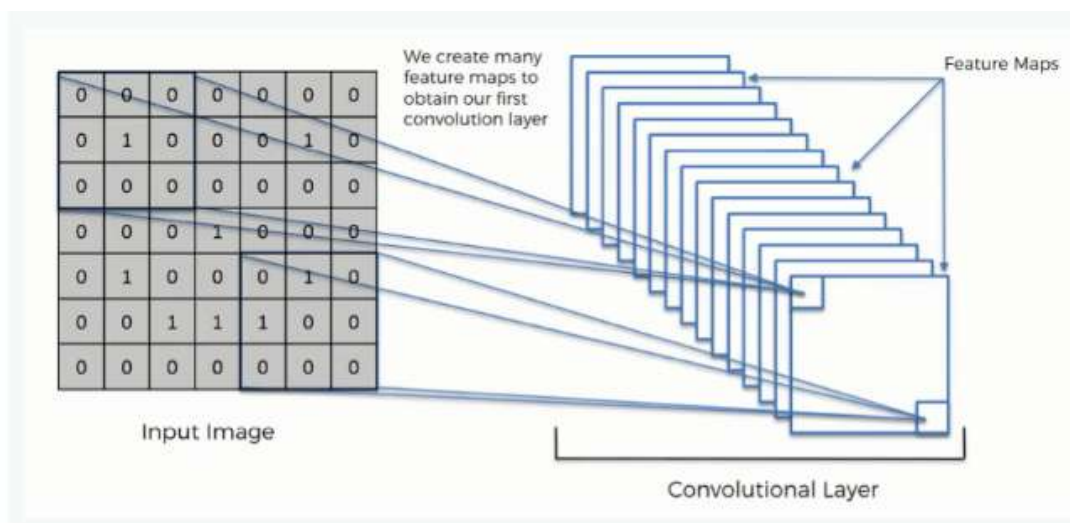
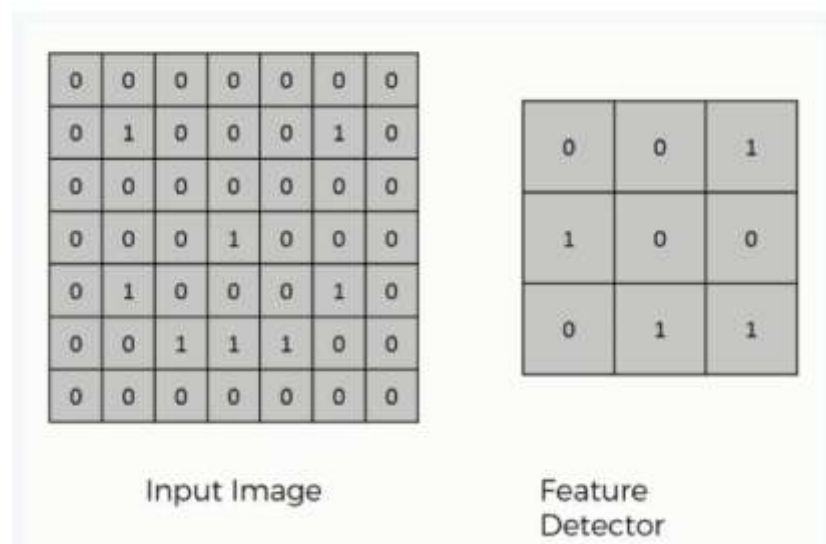
METHODOLOGY AND ALGORITHMS

4.1 Convolutional Neural Network

Step1: convolutional operation

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

The Convolution Operation



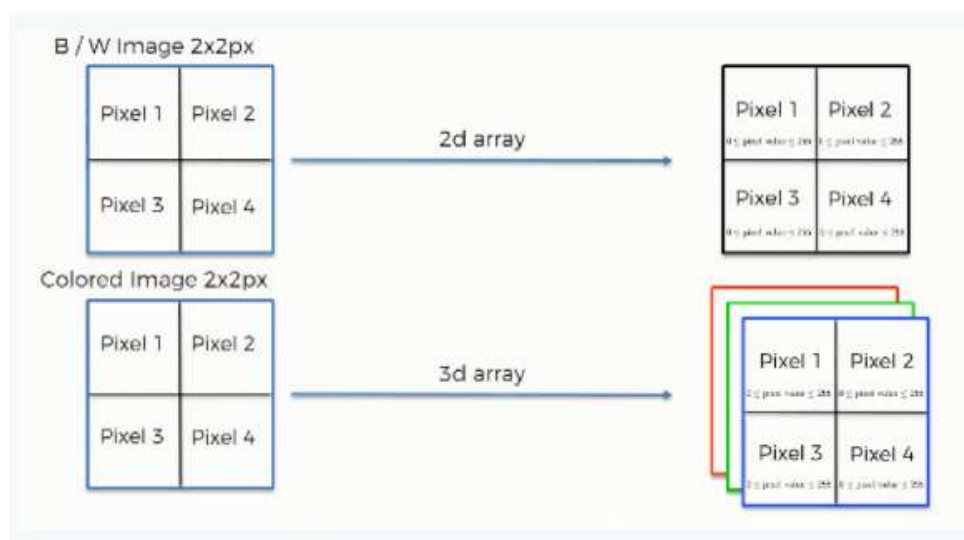
4.1.1 Convolution Operation

Step (1b): ReLU Layer

The second part of this step will involve the Rectified Linear Unit or ReLU. We will cover ReLU layers and explore how linearity functions in the context of Convolutional Neural Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.

Convolutional Neural Networks Scan Images



4.1.2 Convolution Neural Networks Scan Images

Step 2: Conv2D

Keras Conv2D is 2D Convolution Layer; this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

Kernel: In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image

Step 3: Flattening

This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

Step 4: Full Connection

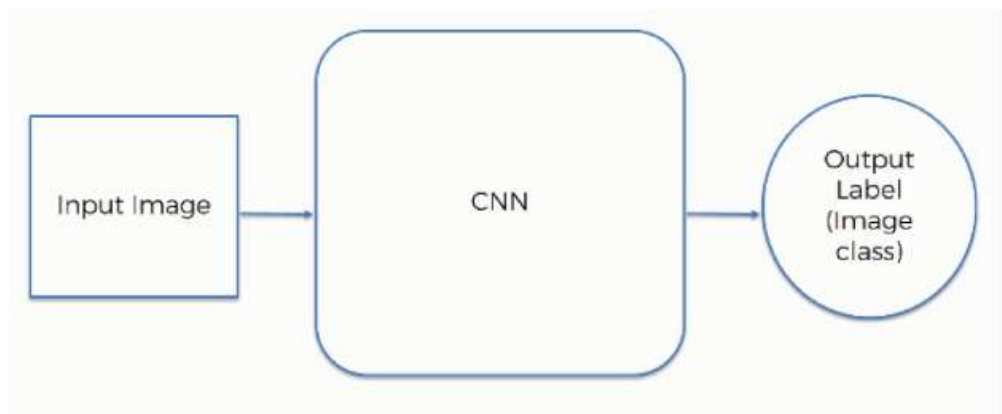
In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

Summary

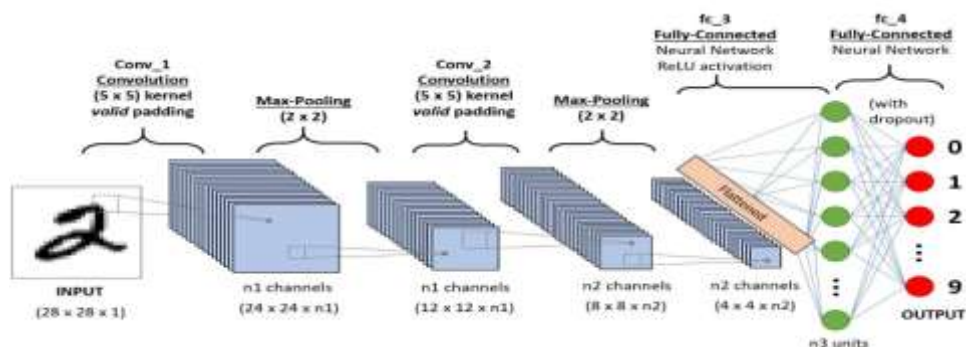
In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Softmax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.

Convolutional neural network (CNN):

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution.



4.1.3 CNN Input and Output



4.1.4 Model Architecture of Proposed CNN Model

4.2 ResNet50:

ResNet50 is a convolutional neural network which has a depth of 50 layers. It was build and trained by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 and you can access the model performance results on their paper, titled Deep Residual Learning for Image Recognition. This model is also trained on more than 1 million images from the ImageNet database. Just like VGG-19, it can classify up to 1000 objects and the network was trained on 224x224 pixels colored images. Here is brief info about its size and performance:

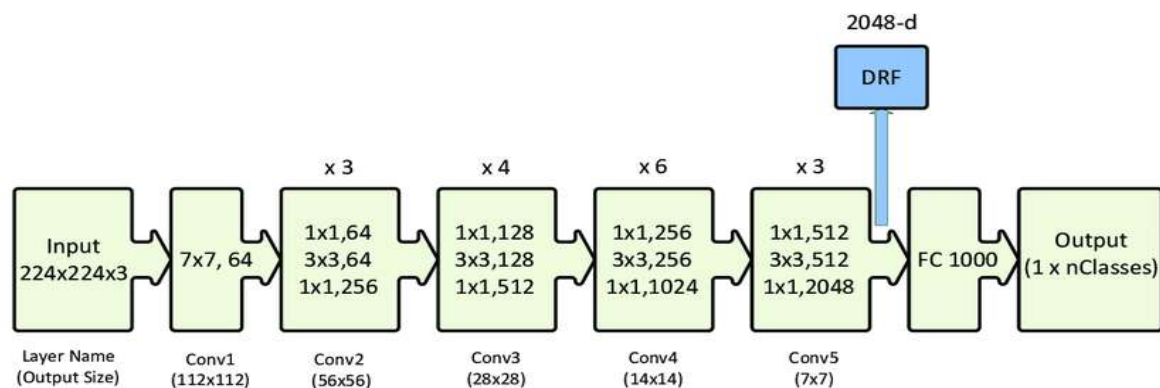


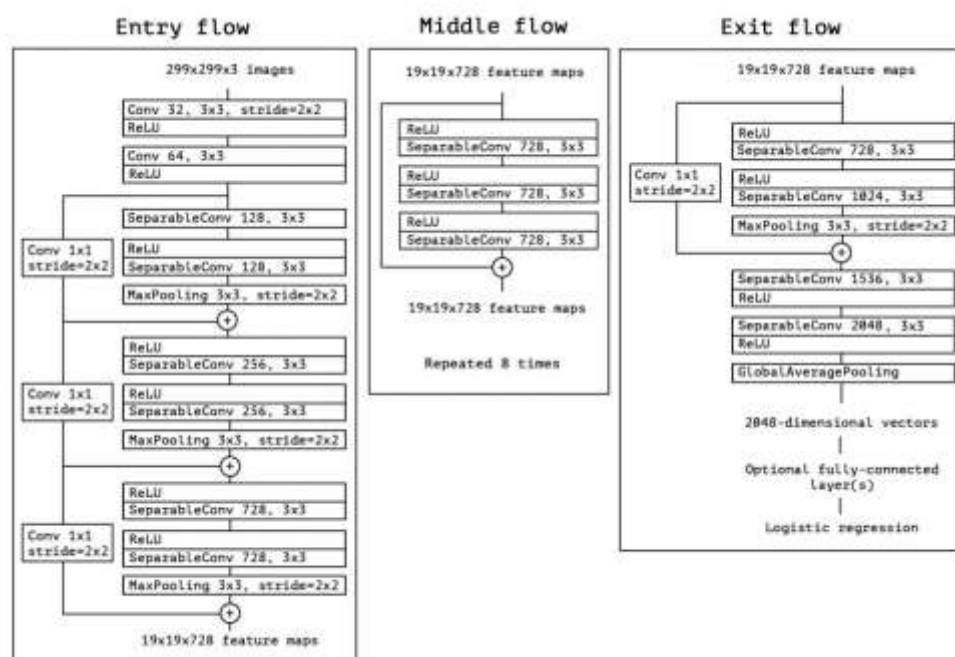
Fig 4.2.1 ResNet50 Architecture

To sum up, residual network or ResNet was a major innovation that has changed the training of deep convolutional neural networks for tasks related to computer vision. While the original Resnet had 34 layers and used 2-layer blocks, other advanced variants such as the Resnet50 made the use of 3-layer bottleneck blocks to ensure improved accuracy and lesser training time. Keras is a deep learning API that is popular due to the simplicity of building models using it. Keras comes with several pre-trained models, including Resnet50, that anyone can use for their experiments. Therefore, building a residual network in Keras for computer vision tasks like image classification is relatively simple. You only need to follow a few simple steps.

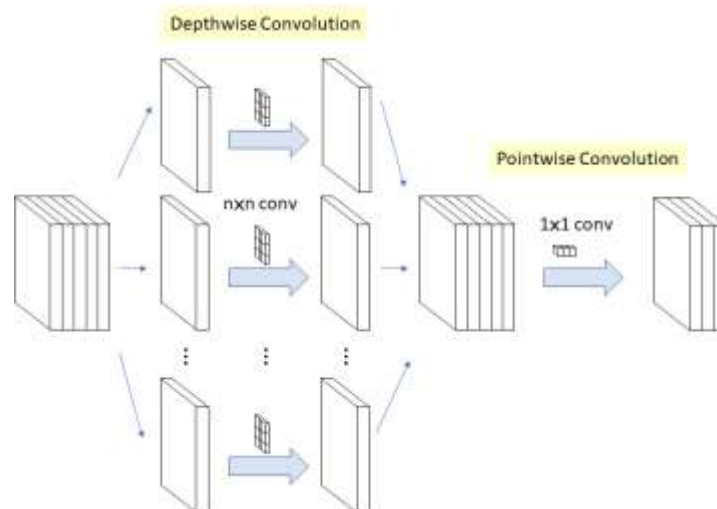
4.3 Xception Architecture

Xception is a deep convolutional neural network architecture that involves Depth wise Separable Convolutions. This observation leads them to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depth wise separable convolutions.

Xception is a deep convolutional neural network architecture that involves Depth wise Separable Convolutions. It was developed by Google researchers. Google presented an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depth wise separable convolution operation (a depth wise convolution followed by a point wise convolution). In this light, a depth wise separable convolution can be understood as an Inception module with a maximally large number of towers. This observation leads them to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depth wise separable convolutions.



4.3.1 Architecture Of Xception Model



4.3.2 DepthWise and PointWise Convolution

CHAPTER 5

SOFTWARE DEVELOPMENT LIFE CYCLE

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.

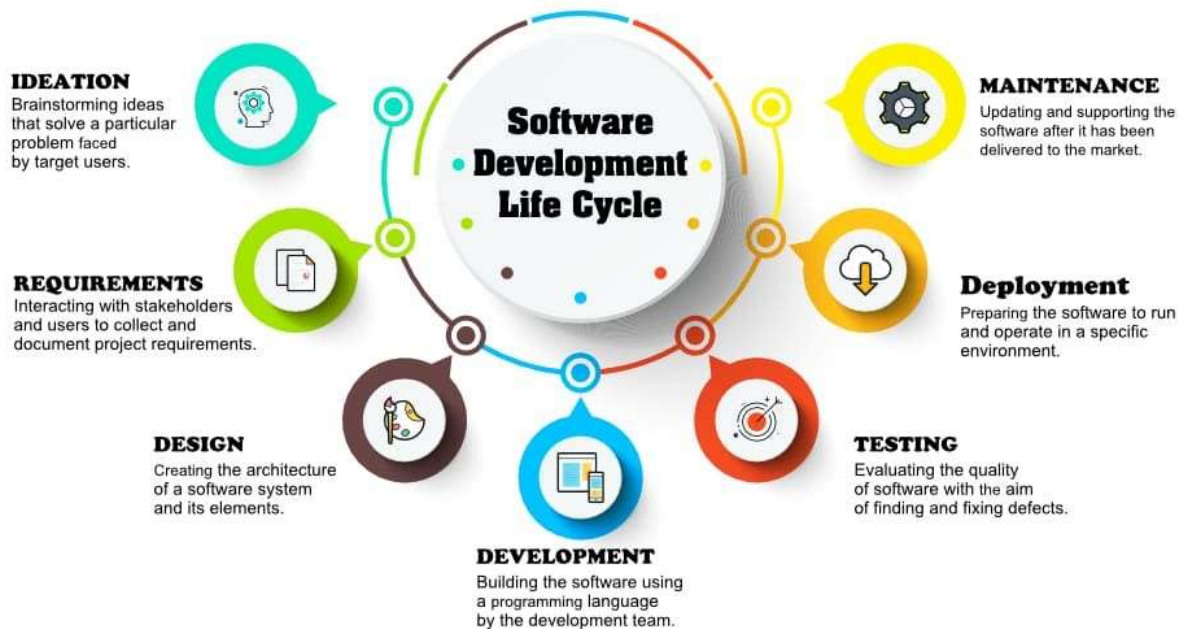


Fig 5.1: Waterfall Model

- **Requirement Gathering and analysis** – all possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

5.1 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

5.1.1 Economic feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. Technical feasibility: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.2 Social feasibility:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.1.3 Technical Feasability

Technical feasibility is a standard practice for companies to conduct feasibility studies before commencing work on a project. Businesses undertake a technical feasibility study to assess the practicality and viability of a product or service before launching it. Whether you are working as a product engineer, product designer or team manager, there may be plenty of situations in your career where you have to prepare a technical feasibility study. In this article, we discuss what is technical feasibility, explain how to conduct one and share tips on writing a feasibility study report.

CHAPTER 6

SYSTEM REQUIREMENTS SPECIFICATION

Functional and non-functional requirements:

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

6.1 Functional Requirements:

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

- 1) Authentication of user whenever he/she logs into the system
- 2) System shutdown in case of a cyber-attack
- 3) A verification email is sent to user whenever he/she register for the first time on some software system.

6.2 Non-functional requirements:

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Examples of non-functional requirements:

- 1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
- 2) The processing of each request should be done within 10 seconds
- 3) The site should load in 3 seconds whenever of simultaneous users are > 10000

6.3 SYSTEM SPECIFICATIONS:

H/W Specifications:

- Processor : I5/Intel Processor
- RAM : 8GB (min)
- Hard Disk : 128 GB

S/W Specifications:

- Operating System : Windows 10
- Server-side Script : Python 3.6
- IDE : PyCharm, Jupyter notebook
- Libraries Used : Numpy, IO, OS, Flask, Keras, pandas, tensorflow

CHAPTER 7

SYSTEM DESIGN

7.1 Input Design

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.
- All these objectives are obtained using the knowledge of basic design principles regarding
 - What are the inputs needed for the system?
 - How end users respond to different elements of forms and screens.

Objectives for Input Design

The objectives of input design are

- To design data entry and input procedures
- To reduce input volume
- To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

7.2 Output Design

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

Objectives of Output Design:

The objectives of output design are:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

7.3 MODULES:

1. System:

1.1 Create Dataset:

The dataset containing images of the Skin disease classification images with the Classification i.e., normal are to be classified is split into training and testing dataset with the test size of 30-20%.

1.2 Pre-processing:

Resizing and reshaping the images into appropriate format to train our model.

1.3 Training:

Use the pre-processed training dataset is used to train our model using CNN Deep learning along with Resnet50 transfer learning methods.

1.4 Classification:

The results of our model are display of Skin disease classification images are either with different labels

2. User:

2.1 Upload Image

The user has to upload an image which needs to be classified.

2.2 View Results

CHAPTER 8

UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with UML. The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

8.1 USE CASE DIAGRAM

- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
- Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

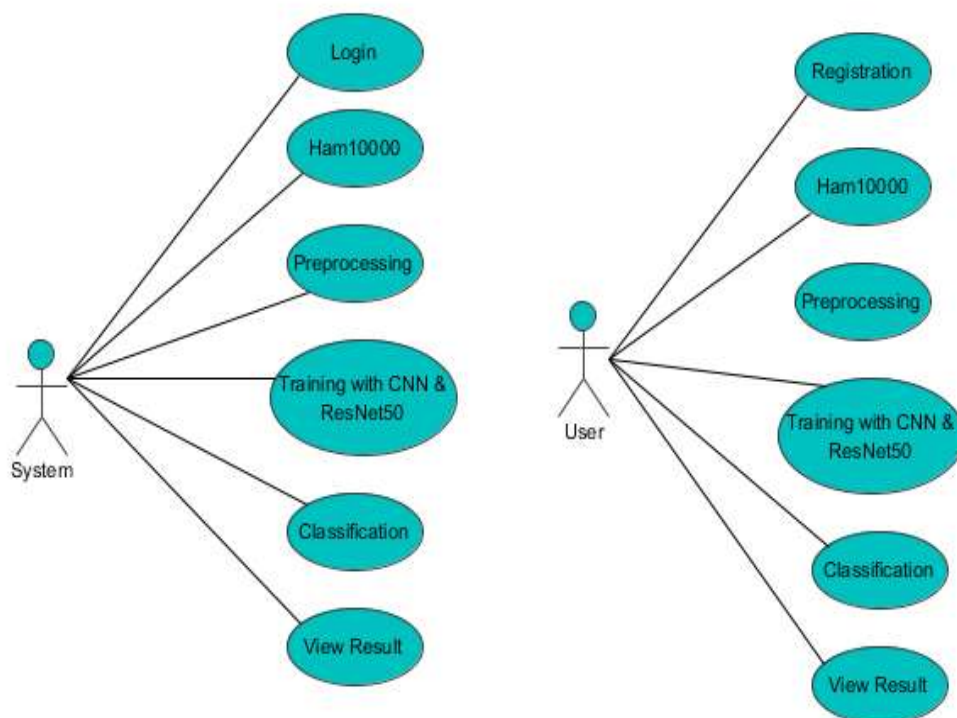


Fig 8.1 Use Case Diagram

8.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information



Fig 8.2 Class Diagram

8.3 SEQUENCE DIAGRAM

- A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
- It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams

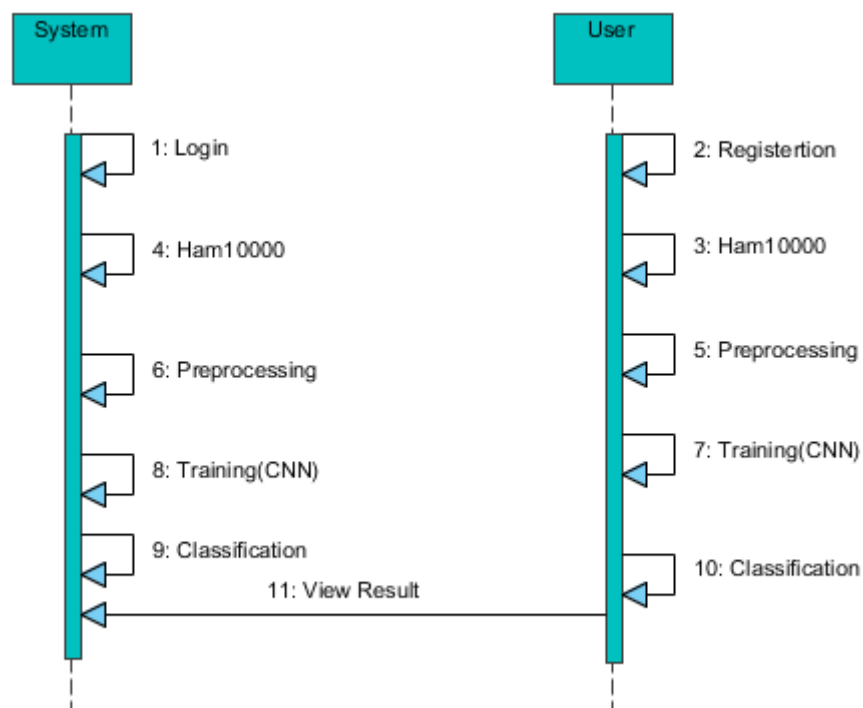


Fig 8.3 Sequence Diagram

8.4 COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does

not describe the object organization whereas the collaboration diagram shows the object organization.

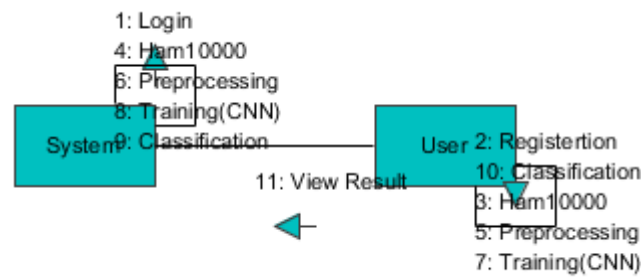


Fig 8.4 Collabaration Diagram

8.5 DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.

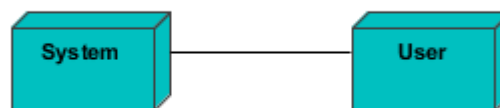


Fig 8.5 Deployment Diagram

8.6 COMPONENT DIAGRAM:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.

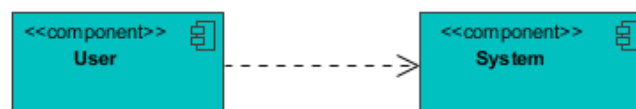


Fig 8.6 Component Diagram

8.6 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

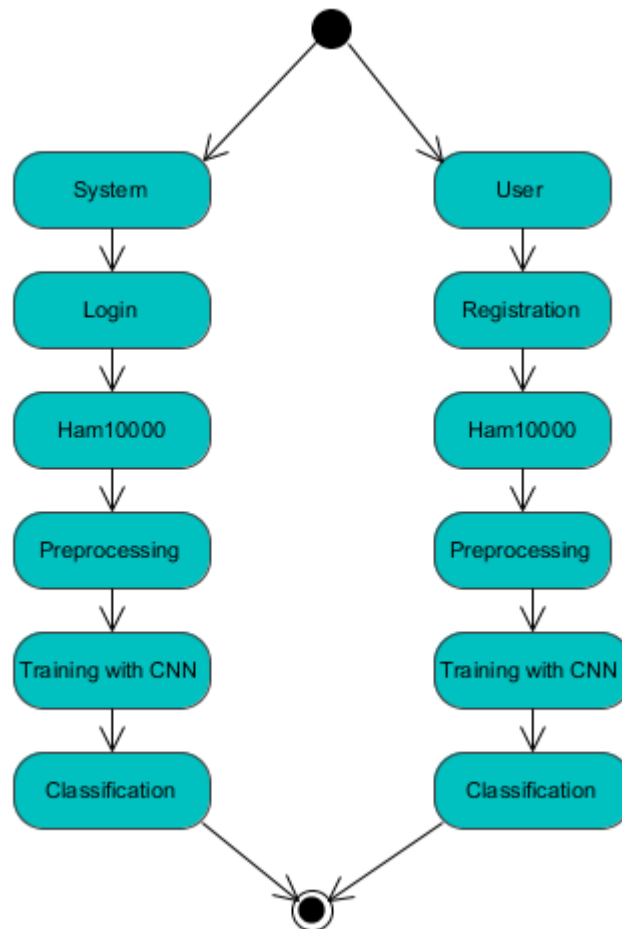


Fig 8.6 Activity Diagram

CHAPTER 9

ER & DATA FLOW DIAGRAMS

9.1 ER DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

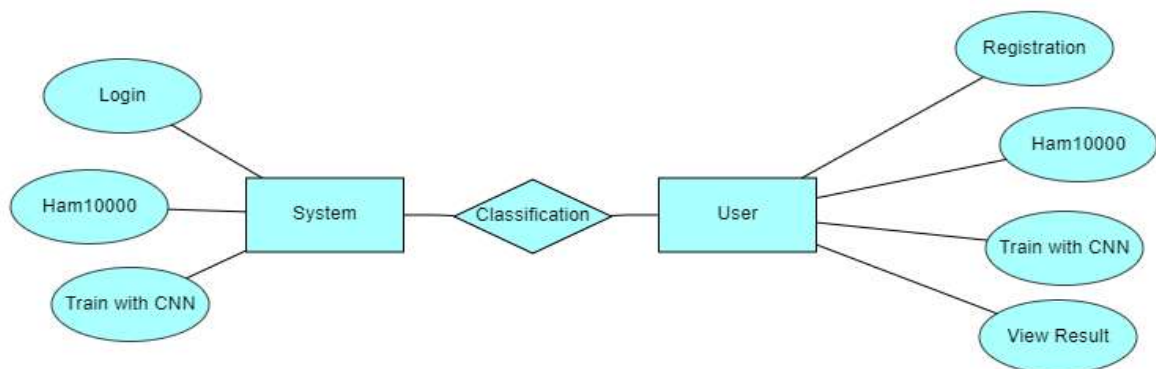


Fig 9.1 ER Diagram

9.2 DFD DIAGRAM

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

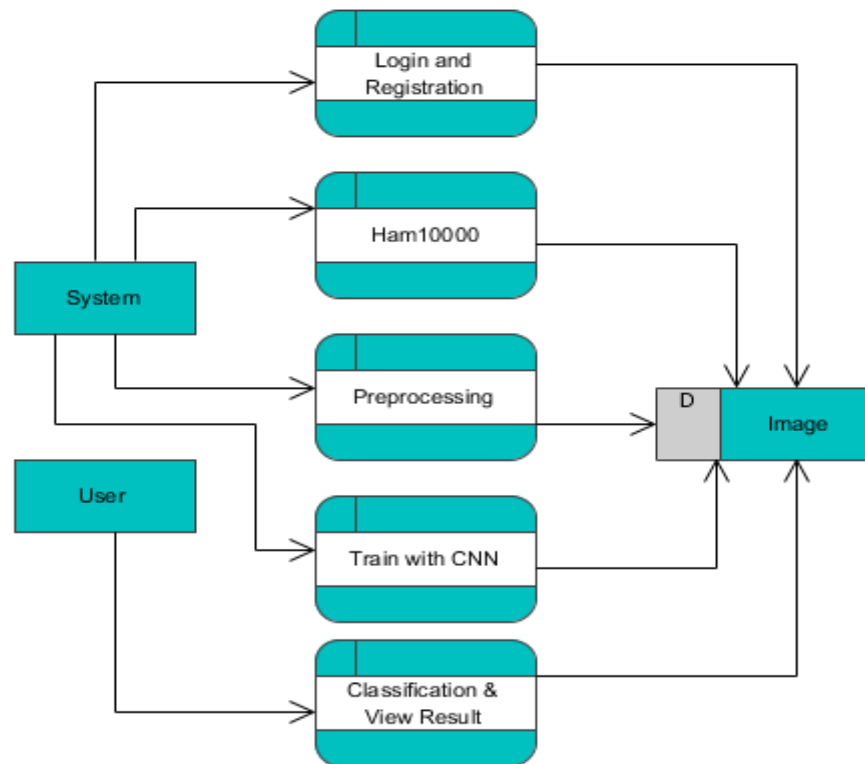


Fig 9.2 Data Flow Diagram

CHAPTER 10

EXECUTION RESULTS

OUTPUT SCREEN SHOTS WITH DESCRIPTION.

Home: In our project, we are classifying the presence of Skin Disease Classification, with the help of CNN and Transfer learning.



Fig10.1: Home

About Project: Here the user will get a breif idea about the project.



Fig 10.2: About Project

User Home



Fig 10.3: User Home

Image Upload Page



Fig 10.4 Upload Image

Actinic keratosis Image

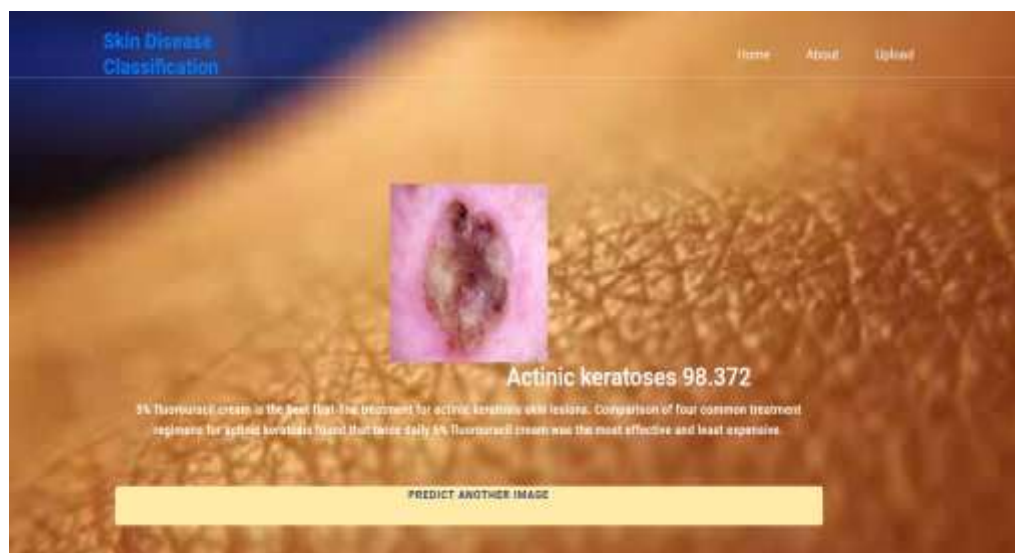


Fig 10.5 Actinic keratosis Image

Basal cell carcinoma

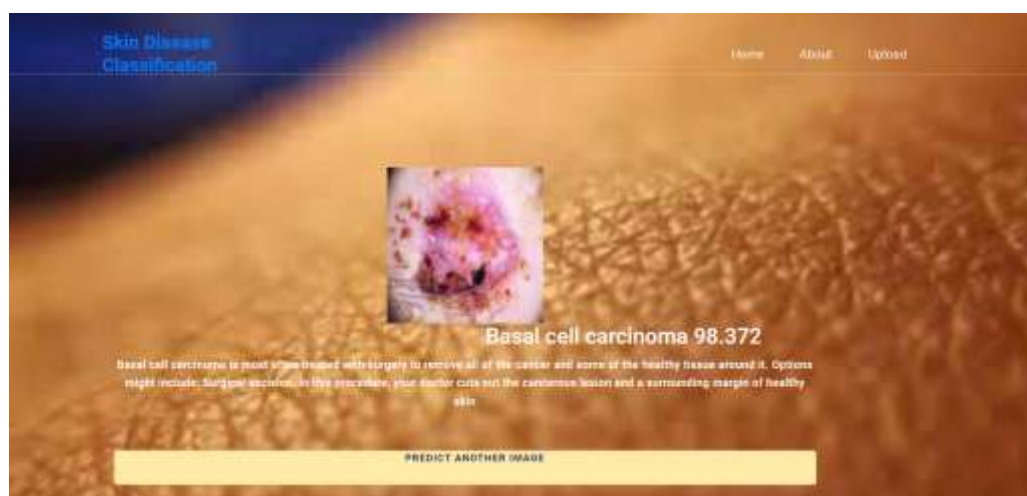


Fig 10.6 Basal cell carcinoma

Benign keratosis-like lesions



Fig 10.7 Benign keratosis-like lesions

Dermatofibroma



Fig 10.8 Dermatofibroma

Melanoma



Fig 10.9 Melanoma

Vascular lesions

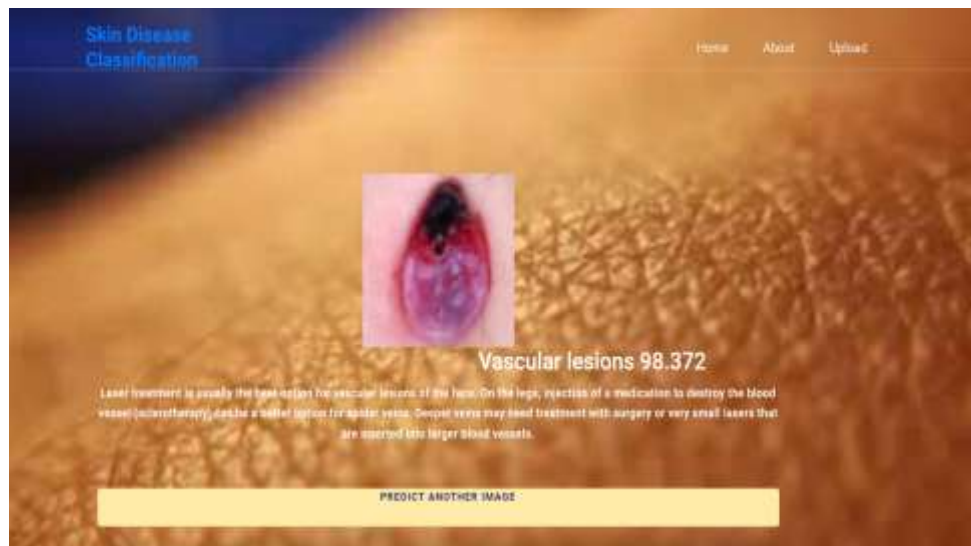


Fig 10.10 Vascular Lesions

Conclusion

These are the required screenshots of the classification of diseases regarding execution of project.

TEST CASES:

Input	Output	Result
Input text	Tested for the classification of Skin Diseases Classification	Success

TEST CASES MODEL BUILDING:

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the dataset.	Dataset path.	Dataset need to read successfully.	Dataset fetched successfully.	P
2	Performing pre-processing on the dataset	Pre-processing part takes place	Pre-processing should be performed on dataset	Pre-processing successfully completed.	P
3	Model Building	Model Building for the clean data	Need to create model using required algorithms	Model Created Successfully.	P
4	Classification	Input image provided.	Output should be the Identification of Skin Diseases Classification	Model classified successfully	P

CHAPTER 11

BIBLIOGRAPHY

SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:

11.1 Installing Python:

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



11.1 Installing Python

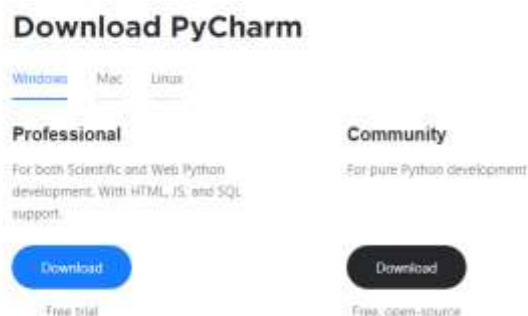
Once the download is complete, run the exe for install Python. Now click on Install Now.

You can see Python installing at this point.

When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

11.2 Installing PyCharm:

- To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and click the "DOWNLOAD" link under the Community Section.



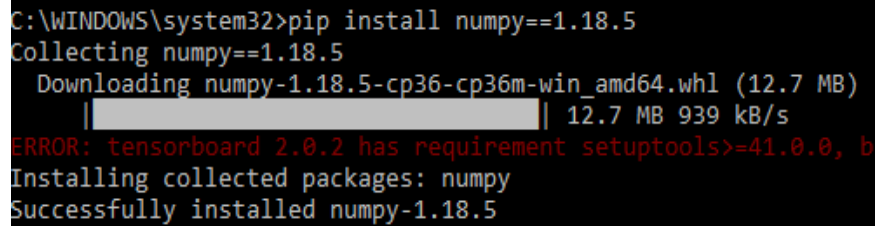
- Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
- On the next screen, Change the installation path if required. Click “Next”.
- On the next screen, you can create a desktop shortcut if you want and click on “Next”.
- Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
- Wait for the installation to finish.
- Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
- After you click on "Finish," the Following screen will appear.



Fig 11.2 Downloading Pycharm

- You need to install some packages to execute your project in a proper way.
- Open the command prompt/ anaconda prompt or terminal as administrator.
- The prompt will get open, with specified path, type “pip install package name” which you want to install (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: Pip install NumPy



```
C:\WINDOWS\system32>pip install numpy==1.18.5
Collecting numpy==1.18.5
  Downloading numpy-1.18.5-cp36-cp36m-win_amd64.whl (12.7 MB)
    | 12.7 MB 939 kB/s
ERROR: tensorboard 2.0.2 has requirement setuptools>=41.0.0, b
Installing collected packages: numpy
Successfully installed numpy-1.18.5
```

11.3 INTRODUCTION TO PYTHON

Python

Script

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad*, *Microsoft WordPad*, *Microsoft Word*, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how's and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open-source general-purpose language.
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Easy-is to interface with C++ (via SWIG)
- Great interactive environment
- Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

- Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read – Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain – Python's source code is fairly easy-to-maintained.
- A broad standard library – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable – you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases – Python provides interfaces to all major commercial databases.
- GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are

listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is. For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an

integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

20 Python libraries

1. Requests. The most famous http library written by Kenneth remits. It's a must have for every python developer.
2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.
3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQL Alchemy. A database library. Many love it and many hate it. The choice is yours.
6. Beautiful Soup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.
8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.
9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made
13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI's for my python scripts.
14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.
15. Scaly. A packet sniffer and analyzer for python made in python.
16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

Numpy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numpy dimensions are called *axes*. The number of axes is *rank*.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

- High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

- The class statement creates a class object and gives it a name. This creates a new namespace.
- Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: `ClassName.Attribute`.
- Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.
- Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

- Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
- Using the term `self` identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by subclassing a class, you can change the behaviour of the program by simply

adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods

Exceptions

Essentially, exceptions are events that modify program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non-existent dictionary key will raise a Key Error exception.
- Searching a list for a non-existent value will raise a Value Error exception
- Calling a non-existent method will raise an Attribute Error exception.
- Referencing a non-existent variable will raise a Name Error exception.
- Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing.

Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/else statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won't have to do this very often but it's nice to have the option when necessary.

However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

His entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.

There are three types of functions in python:

Help (), min (), print ().

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard. XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces. Simple API for XML SAX: Here, you register call-backs for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the

file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree – based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](#) where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Jingo vs. Flask vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language agnostic Reddit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

11.4 System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTING

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM_TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CONCLUSION

In this project we have successfully classified the images of Identification of Skin Diseases, are affected with the using the deep learning and Transfer learning. Here, we have considered the dataset of ham10000 images which will be of different types Diseases and trained using CNN along with some ResNet50 and xception transfer learning method. After the training we have tested by uploading the image and classified it.

Future Scope

This can be utilized in future to classify the types of different Diseases to prevent the Precautions to the patients.

Motivation

To develop a solution that can help dermatologists better support their diagnostic accuracy by ensembling contextual images and patient-level information, reducing the variance of predictions from the model.

REFERENCES

1. Bi X, Wang H. Early Alzheimer's disease diagnosis based on EEG spectral images using deep learning. *Neural Netw* 2019;114:119-35.
2. Brosch T, Tam R, Initiative for the Alzheimers Disease Neuroimaging. Manifold learning of brain MRIs by deep learning. *Med Image Comput Comput Assist Interv* 2013;16:633-40.
3. Prasoon A, Petersen K, Igel C, Lauze F, Dam E, Nielsen M. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. *Med Image Comput Comput Assist Interv* 2013;16:246-53.
4. Brosch T, Tang LY, Youngjin Yoo, et al. Deep 3D Convolutional Encoder Networks With Shortcuts for Multiscale Feature Integration Applied to Multiple Sclerosis Lesion Segmentation. *IEEE Trans Med Imaging* 2016;35:1229-39.
5. Cheng JZ, Ni D, Chou YH, et al. Computer-Aided Diagnosis with Deep Learning Architecture: Applications to Breast Lesions in US Images and Pulmonary Nodules in CT Scans. *Sci Rep* 2016;6:24454.
6. Gulshan V, Peng L, Coram M, et al. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA* 2016;316:2402-10.
7. Esteva A, Kuprel B, Novoa RA, et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 2017;542:115-8.
8. Haenssle HA, Fink C, Schneiderbauer R, et al. Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Ann Oncol* 2018;29:1836-42.
9. Masood A, Al-Jumaily AA. Computer aided diagnostic support system for skin cancer: a review of techniques and algorithms. *Int J Biomed Imaging* 2013;2013:323268.
10. Burrioni M, Corona R, Dell'Eva G, et al. Melanoma computer-aided diagnosis: reliability and feasibility study. *Clin Cancer Res* 2004;10:1881-6.
11. Erickson BJ. Machine Learning: Discovering the Future of Medical Imaging. *J Digit Imaging* 2017;30:391.

11. Fishbein AB, Silverberg JI, Wilson EJ, et al. Update on Atopic Dermatitis: Diagnosis, Severity Assessment, and Treatment Selection. *J Allergy Clin Immunol Pract* 2020;8:91-101.
12. Tan M, Le Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML*, 2019.
13. Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge. *Int J Comput Vis* 2015;115:211-52.
14. Hu J, Shen L, Albanie S, et al. Squeeze-and-Excitation Networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* 2018:7132-41.
15. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition* 2016:2818-26.
16. Bain EE, Shafner L, Walling DP, et al. Use of a Novel Artificial Intelligence Platform on Mobile Devices to Assess Dosing Compliance in a Phase 2 Clinical Trial in Subjects With Schizophrenia. *JMIR Mhealth Uhealth* 2017;5:e18.
17. Shao J, Xue S, Yu G, et al. Smartphone-controlled optogenetically engineered cells enable semiautomatic glucose homeostasis in diabetic mice. *Sci Transl Med* 2017. doi: 10.1126/scitranslmed.aal2298.
18. Liang H, Tsui BY, Ni H, et al. Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence. *Nat Med* 2019;25:433-8.