**DETECTION OF DIABETES MELLITUS WITH DEEP LEARNING AND DATA AUGMENTATION TECHNIQUES ON FOOT THERMOGRAPHY**

**ABSTRACT**

There has been a substantial amount of research involving computer methods and technology for the detection and recognition of diabetic foot ulcers (DFUs), but there is a lack of systematic comparisons of state-of-the-art deep learning object detection frameworks applied to this problem. DFUC2020 provided participants with a comprehensive dataset consisting of 2,000 images for training and 2,000 images for testing. This paper summarises the results by comparing the deep learning-based algorithms proposed by the winning teams: Faster R-CNN, three variants of Faster R-CNN and an ensemble method; YOLOv3; YOLOv5; EfficientDet; and a new Cascade Attention Network. For each deep learning method, we provide a detailed description of model architecture, parameter settings for training and additional stages including pre-processing, data augmentation and post-processing. We provide a comprehensive evaluation for each method. All the methods required a data augmentation stage to increase the number of images available for training and a post-processing stage to remove false positives. The best performance was obtained from Deformable Convolution, a variant of Faster R-CNN, with a mean average precision (map) of 0.6940 and an F1-Score of 0.7434. Finally, we demonstrate that the ensemble method based on different deep learning methods can enhanced the F1-Score but not the map.

**Keyword:** Diabetic Foot, Ulcers, deep learning, convolutional neural network (CNN), Mobile net model, minutiae.

**CHAPTER 1**

**INTRODUCTION**

According to the International Diabetes Federation Saeedi et al. (2019), in 2019 there were approximately 463 million adults with diabetes worldwide. This number is expected to grow to 700 million by 2045. A person with diabetes has a 34% lifetime risk of developing a diabetic foot ulcer (DFU). In other words, 1 in every 3 people with diabetes will develop a DFU in their lifetime Armstrong et al. (2017). Infection of a DFU. Frequently leads to limb amputation, causing significant morbidity, psychological distress and reduced quality of life and life expectancy. This research is the first step of a future diabetic foot care project. Periodic monitoring of foot ulcers is important to assess the progress of ulcer healing, which is currently performed manually by clinicians. Many foot clinics take photographs of ulcers during initial evaluation and subsequent reviews for comparison of various stages of ulcer progression to boost the visual memory of clinicians. The current research aims to develop artificial intelligence-based deep learning algorithms for detection of ulcers without direct clinical intervention. This is especially important in the current COVID-19 climate, where social distancing is of paramount importance Technologies developed to enhance ulcer diagnostics and care plans have the potential to revolutionise diabetic foot care. Detection tasks can be challenging when taking into account the numerous environmental elements in real-world settings.

Examples of some observations include • Newly acquired and subtle early stages of ulceration can easily be missed by care personnel during visual assessment of priorly acquired conditions due to the short time designated for standard treatment • Low-quality images with bad focus, motion blur, occlusion, poor lighting, and backlight are common in wound documentation due to limited available time for treatment and documentation, even when performed by trained personnel • Malformed toenails, deep rhagades, folded amputation scars, and fresh epithelialization are examples for false positive detections that require manual correction, which can be time consuming when documenting DFU • Very small and very large and curved ulcers are problematic for certain detectors, but are common in typical wound care documentation It is essential to develop a technological solution capable of transforming current screening practices that have the potential to significantly reduce clinical time burdens. With the emerging growth of deep learning, automated analysis of DFU has become possible. However, deep learning requires large-scale datasets to achieve results comparable with those of human experts. Currently, medical imaging researchers are working in isolation and the majority of their research is not reproducible. To bridge the gap and to motivate data sharing amongst researchers and clinicians, Yap et al. Yap et al. (2020c,b) proposed the diabetic foot ulcer challenges. This paper presents an overview of the state-of-the-art computer methods in DFU detection, provides an overview of the publicly available datasets, presents a comprehensive evaluation of the popular object detection frameworks on DFU detection, proposes an ensemble method and Cascade Attention DetNet for DFU detection, and conducts a comprehensive evaluation of the deep learning algorithms trained on the DFUC2020 dataset

**CHAPTER 2**

**LITERATURE REVIEW**

**Deep learning in diabetic foot ulcers detection: A comprehensive evaluation: Moi Hoon Yap, Ryo Hachiuma, Azadeh Alavi and team.**

There has been a substantial amount of research involving computer methods and technology for the detection and recognition of diabetic foot ulcers (DFUs), but there is a lack of systematic comparisons of state-of-the-art deep learning object detection frameworks applied to this problem. DFUC2020 provided participants with a comprehensive dataset consisting of 2,000 images for training and 2,000 images for testing. This paper summarises the results of DFUC2020 by comparing the deep learning-based algorithms proposed by the winning teams: Faster R-CNN, three variants of Faster R-CNN and an ensemble method; YOLOv3; YOLOv5; EfficientDet; and a new Cascade Attention Network. For each deep learning method, we provide a detailed description of model architecture, parameter settings for training and additional stages including pre-processing, data augmentation and post-processing. We provide a comprehensive evaluation for each method. All the methods required a data augmentation stage to increase the number of images available for training and a post-processing stage to remove false positives. The best performance was obtained from Deformable Convolution, a variant of Faster R-CNN, with a mean average precision (map) of 0.6940 and an F1-Score of 0.7434. Finally, we demonstrate that the ensemble method based on different deep learning methods can enhanced the F1-Score but not the map

**Summary:** We conduct a comprehensive evaluation of the performance of deep learning object detection networks for DFU detection. Deformable convolutions appear to work well in DFU detections and contribute to the improvement to the best performing method. While the overall results show the potential of localising DFUs using CNNs, the number of false positive results is significant, and the networks are not always able to effectively discriminate ulcers from other skin conditions. The introduction of a second classifier based on a negative dataset may provide a possible solution to this issue when training future networks. However, in reality, it may prove impossible to gather all possible negative examples for supervised learning algorithms. This approach could also impact network size and complexity, which could negatively impact inference speed.

**Faster R-CNN Approach for Diabetic Foot Ulcer Detection: Artur Leandro da Costa Oliveira, Andre Britto de Carvalho, b and Daniel Oliveira Dantas**

Diabetic Foot Ulcer (DFU) is one of the major health concerns about Diabetes. These injuries impair the patient’s quality of life, bring high costs to public health, and can even lead to limb amputations. The use of automatic tools for detection can assists specialists in the prevention and treatment of the disease. Some methods to address this problem based on machine learning have recently been presented. This article proposes the use of deep learning techniques to assist the treatment of DFUs, more specifically, the detection of ulcers through photos taken from the patient’s feet. We propose an improvement of the original Faster R-CNN using data augmentation techniques and changes in parameter settings. We used a training dataset with 2000 images of DFUs annotated by specialists. The training was validated using the Monte Carlo cross-validation technique. Our proposal achieved a mean average precision of 91.4%, a F1-score of 94.8%, and an average detection speed of 332ms which outperformed traditional detector implementations

**Summary:** We conduct a comprehensive evaluation of the performance of deep learning object detection networks for DFU detection. Deformable convolutions appear to work well in DFU detections and contribute to the improvement to the best performing method. While the overall results show the potential of localising DFUs using CNNs, the number of false positive results is significant, and the networks are not always able to effectively discriminate ulcers from other skin conditions. The introduction of a second classifier based on a negative dataset may provide a possible solution to this issue when training future networks. However, in reality, it may prove impossible to gather all possible negative examples for supervised learning algorithms. This approach could also impact network size and complexity.

**A Deep Learning Method for Early Detection of Diabetic Foot Using Decision Fusion and Thermal Images: Khairul Munadi, Khairun Saddami, Maulisa Oktiana and team**

Diabetes mellitus (DM) is one of the major diseases that cause death worldwide and lead to complications of diabetic foot ulcers (DFU). Improper and late handling of a diabetic foot patient can result in an amputation of the patient’s foot. Early detection of DFU symptoms can be observed using thermal imaging with a computer-assisted classifier. Previous study of DFU detection using thermal image only achieved 97% of accuracy, and it has to be improved. This article proposes a novel framework for DFU classification based on thermal imaging using deep neural networks and decision fusion. Here, decision fusion combines the classification result from a parallel classifier. We used the convolutional neural network (CNN) model of Shuffle Net and MobileNetV2 as the baseline classifier. In developing the classifier model, firstly, the MobileNetV2 and Shuffle Net were trained using plantar thermo gram datasets. Then, the classification results of those two models were fused using a novel decision fusion method to increase the accuracy rate. The proposed framework achieved 100% accuracy in classifying the DFU thermal images in binary classes of positive and negative cases. The accuracy of the proposed Decision Fusion (DF) was increased by about 3.4% from baseline Shuffle Net and MobileNetV2. Overall, the proposed framework outperformed in classifying the images compared with the state-of-the-art deep learning and the traditional machine-learning-based classifier.

**Summary:** In this article, we proposed a novel framework to classify the thermal images of diabetic foot ulcers (DFU) based on the fusion schema of two CNN classification results. The CNNs used were MobileNetV2 and Shuffle Net. We applied decision fusion on the CNN classification results and dual-stage validation. Decision-fusion-based classification provides information of the negative and positive labels, which improves the generalization model. Shuffle Net is designed to detect the positive images while MobileNetV2 is used to recognize the negative images. The proposed framework (DF) achieves the maximum values on all performance metrics, namely, accuracy, recall, specificity, precision, and FMeasure. Fusing MobileNetV2 and Shuffle Net also results in a model with fewer learning parameters and a smaller model size. Fusion has shown to be a good solution to achieve a higher performance with a smaller model size. In a future study, enriching the dataset to improve generalization and exploring more fusion strategies to develop smaller CNN models, suitable to be embedded on a mobile device, are potential prospects.

**Classification of Diabetic Foot Ulcers Using Class Knowledge Banks: Yi Xu, Kang Han, Yongming Zhou, Jian Wu, team.**

Nowadays, there has been an increase in security concerns regarding fingerprint biometrics. This problem arises due to technological advancements in bypassing and hacking methodologies. This has sparked the need for a more secure platform for identification. In this paper, we have used a deep Convolutional Neural Network as a pre-verification filter to filter out bad or malicious fingerprints. As deep learning allows the system to be more accurate at detecting and reducing false identification by training itself again and again with test samples, the proposed method improves the security and accuracy by multiple folds. The implementation of a novel secure fingerprint verification platform that takes the optical image of a fingerprint as input is explained in this paper. The given input is pre-verified using Google’s pre-trained inception model for deep learning applications, and then passed through a minutia-based algorithm for user authentication. Then, the results are compared with existing models.

**Summary:** In this paper, we proposed the method called the class knowledge banks (CKBs) which can effectively extract class knowledge from the training data and explicitly leverage the class knowledge in the testing. The proposed method is an alternative means to produce the logits instead of the usual linear classifiers in the literature. The CKBs leverage their units to extract and represent class knowledge from different perspectives and the similarities between the representation of the input and the corresponding CKBs can be regarded as the logits of the input. The CKB can be trained through back-propagation and be easily embedded into existing image classification models. Experimental results on the DFU infection and ischaemia datasets demonstrate the effectiveness of the proposed CKB in DFU image classifications

**SYSTEM ANALYSIS & FEASIBILITY STUDY**

**Existing Method:**

* In the existing there are methods implemented to classify Diabetic foot ulcer and normal foot in deep learning. In method we are performing the classification classify Diabetic foot ulcer and normal foot identification using Resnet, vgg19, vgg16 of deep learning along with the Machine learning methods. As image analysis based approaches for classification of Diabetic foot ulcer and normal foot.

.

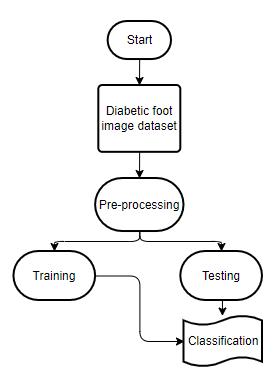
**Disadvantages:**

* Less accuracy.
* More Losses.

**Proposed System:**

In purposed method we are performing the classification of either the image is Diabetic foot ulcer or normal foot identification using Convolution Neural Network (CNN) and Mobile net of deep learning along with the Machine learning methods. As image analysis based approaches for Diabetic foot ulcer and normal foot classification and authentication. Hence, proper classification is important for the Diabetic foot ulcer and normal foot that which will be possible by using our proposed method. Block diagram of proposed method is shown below.

**Block Diagram:**

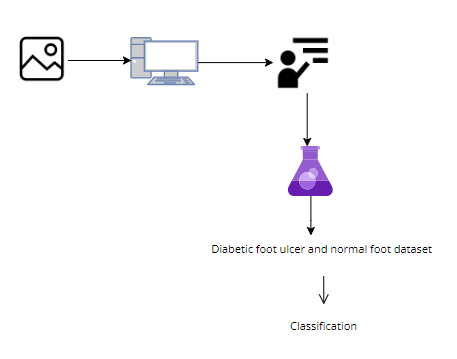


**Fig 1. Block diagram of proposed method**

**Advantages**:

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

**ARCHITECTURE**



**METHODOLOGY AND ALGORITHMS:**

**CNN:**

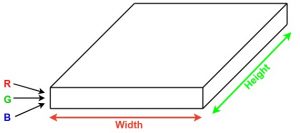
It is assumed that the reader knows the concept of neural networks.  
When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.  
Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

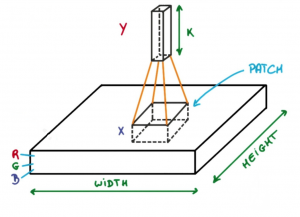
The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we back propagate into the model by calculating the derivatives. This step is called back propagation which basically is used to minimize the loss.  
Here’s the basic python code for a neural network with random inputs and two hidden layers.

**Convolution Neural Network**

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as images generally have red, green, and blue channels). 



Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights. 



*Image source: Deep Learning Udacity*

Now let’s talk about a bit of mathematics that is involved in the whole convolution process. 

* Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
* For example, if we have to run convolution on an image with dimension 34x34x3. The possible size of filters can be axax3, where ‘a’ can be 3, 5, 7, etc but small as compared to image dimension.
* During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
* As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together and as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

**Layers used to build ConvNets**

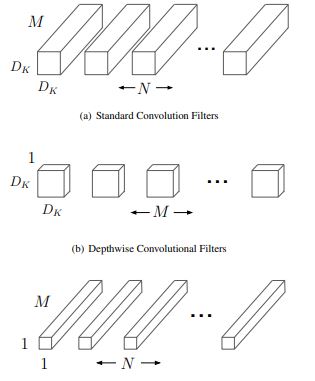
A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.   
**Types of layers:**   
Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3. 

1. **Input Layer:** This layer holds the raw input of the image with width 32, height 32, and depth 3.
2. **Convolution Layer:** This layer computes the output volume by computing the dot product between all filters and image patches. Suppose we use a total of 12 filters for this layer we’ll get output volume of dimension 32 x 32 x 12.
3. **Activation Function Layer:** This layer will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.
4. **Pool Layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

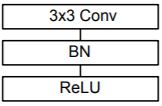
**Mobile Net architecture**.

Convolutional Neural Networks (CNN) have become very popular in computer vision. However, in order to achieve a higher degree of accuracy modern CNNs are becoming deeper and increasingly complex. Such networks cannot be used in real applications like robots and self driving cars.In this section we will be discussing a CNN architecture that aims to effectively tackle this problem.

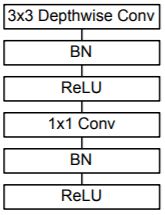
MobileNet is an efficient and portable CNN architecture that is used in real world applications. Mobile Nets primarily use **depth wise separable convolutions** in place of the standard convolutions used in earlier architectures to build lighter models.MobileNets introduce two new global hyper parameters(width multiplier and resolution multiplier) that allow model developers to trade off **latency** or **accuracy** for speed and low size depending on their requirements.



**Standard Convolution layer:**   
A single standard convolution unit (denoted by **Conv** in the table above) looks like this:



**Depth wise separable Convolution layer**  
A single Depth wise separable convolution unit (denoted by **Conv dw** in the table above) looks like this:

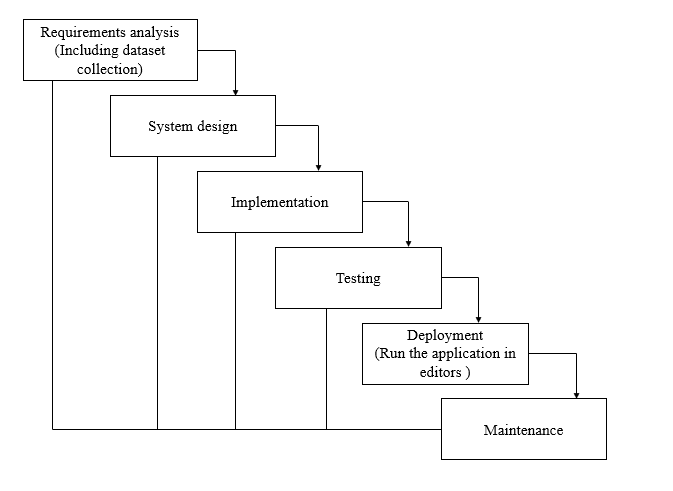


**Width Multiplier**  
Width muliplier (denoted by α) is a global hyperparameter that is used to construct smaller and less computionally expensive models.Its value lies between 0 and 1.For a given layer and value of α, the number of input channels 'M' becomes α \* M and the number of output channels 'N' becomes α \* N hence reducing the cost of computation and size of the model at the cost of performance.The computation cost and number of parameters decrease roughly by a factor of α2.Some commonly used values of α are 1,0.75,0.5,0.25.

**Resolution Multiplier**  
The second parameter introduced in MobileNets is called resolution multiplier and is denoted by ρ.This hyper parameter is used to decrease the resolution of the input image and this subsequently reduces the input to every layer by the same factor. For a given value of ρ the resolution of the input image becomes 224 \* ρ.This reduces the computational cost by a factor of ρ2.

**SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

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



**Fig 3**: 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.

**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

**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.

**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.

**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.

**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.

**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

**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

**SYSTEM DESIGN:**

## 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.

**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 input 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.

**MODULES:**

**System**

**User**

**1. System:**

1.1 Create Dataset:

The dataset containing images of the Diabetic foot ulcers and normal foot 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 and Mobile net Deep learning algorithm along with some of the transfer learning methods.

1.4 Classification:

The results of our model are display of Diabetic foot ulcers and normal foot images classification.

**2. User:**

2.1 Upload Image

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

2.2 View Results

The classified image results are viewed by user.

**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.

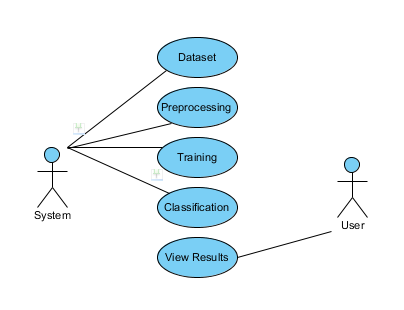
**GOALS:**

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.

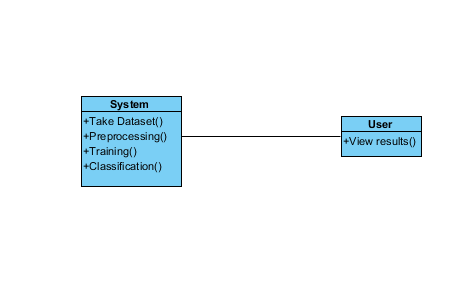
**USE CASE DIAGRAM**

* 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.



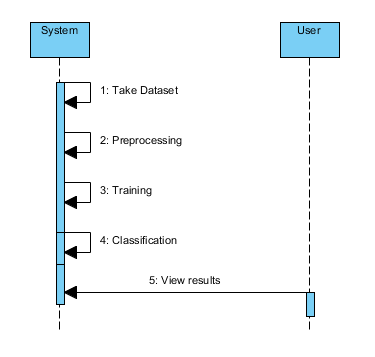
**CLASS DIAGRAM**

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



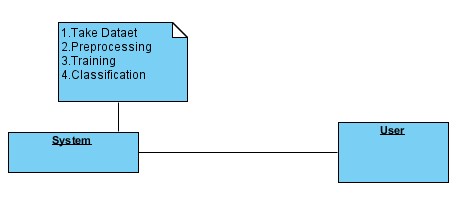
**SEQUENCE DIAGRAM**

* 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



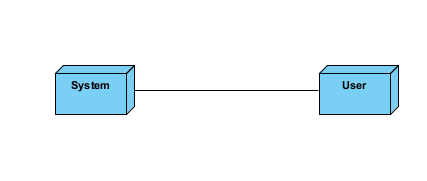
**COLLABORATION DIAGRAM:**

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.



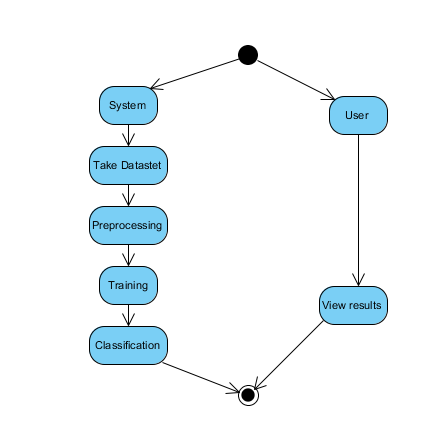
**DEPLOYMENT DIAGRAM**

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.



**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.



**COMPONENT DIAGRAM**:

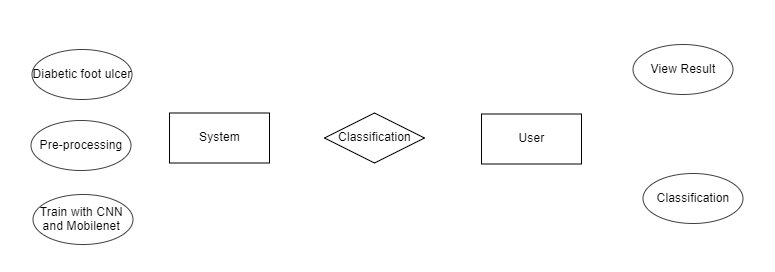
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned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



**ER DIAGRAM:**

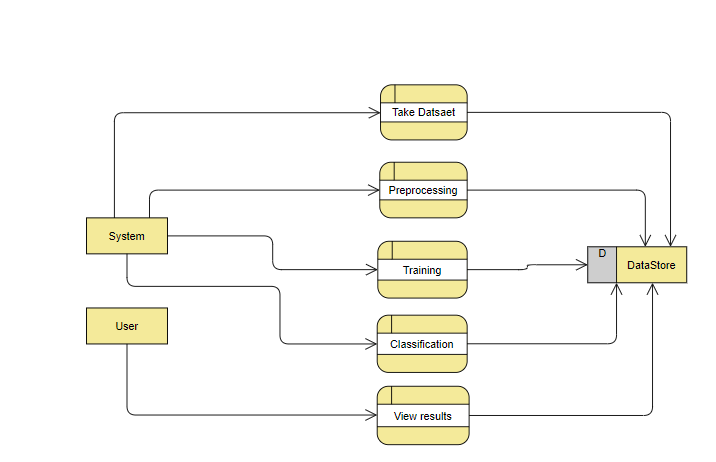
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity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

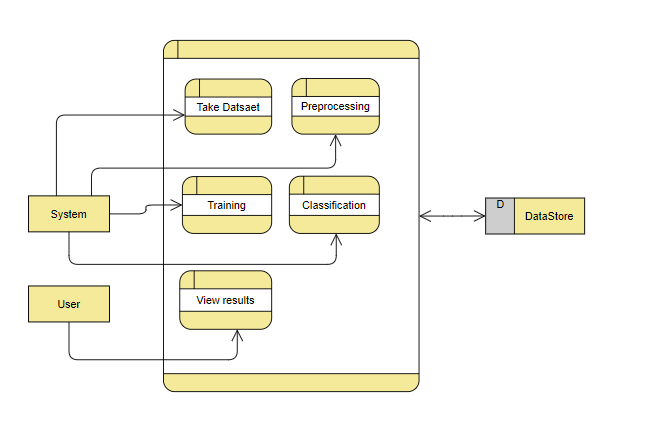
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete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



**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.





**TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Result** |
| Input text | Tested for the classification of Diabetic foot ulcer and foot | 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 Diabetic foot ulcer Classification. | Model classified successfully | P |

**CONCLUSION:**

In this project we have successfully classified the images of Diabetic foot ulcer and normal foot , are either matching with each other using the deep learning and Transfer learning. Here, we have considered the dataset of Diabetic foot ulcer and normal foot which will be of different types trained using CNN and Mobile net, transfer learning method. After the training we have tested by uploading the image and classified.

**FUTURE SCOPE:**

This can be utilized in future to classify and detect the different types of ulcers and make a difference in identifying and early treatment..

**REFERENCES:**

1. D.G. Armstrong, A.J. Boulton, S.A. Bus, Diabetic foot ulcers and their recurrence, N. Engl. J. Med. 376 (2017) 2367–2375, <https://doi.org/10.1056/nejmra1615439>
2. R. Brown, B. Ploderer, L.S. Da Seng, P. Lazzarini, J. van Netten, MyFootCare: a mobile self-tracking tool to promote self-care amongst people with diabetic foot ulcers, in: Proceedings of the 29th Australian Conference on Computer-Human Interaction (OzCHI’17), Association for Computing Machinery, 2017, pp. 462–466, <https://doi.org/10.1145/3152771.3156158>.
3. T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ ar, C. L. Zitnick, Microsoft COCO: common objects in context, in: European Conference on Computer Vision (ECCV) 2014, Springer International Publishing, 2014, pp. 740–755, <https://doi.org/10.1007/978-3-319-10602-1_48>.
4. Y. Cao, K. Chen, C.C. Loy, D. Lin, Prime sample attention in object detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, pp. 11580–11588, <https://doi.org/10.1109/cvpr42600.2020.01160>.
5. K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916, <https://doi.org/10.1109/tpami.2015.2389824>.
6. H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: beyond empirical risk minimization, in: 6th International Conference on Learning Representations (ICLR) 2018, ICLR., 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb. Retrieved on 2021-04-28
7. M.H. Yap, N.D. Reeves, A. Boulton, S. Rajbhandari, D. Armstrong, A.G. Maiya, B. Najafi, E. Frank, J. Wu, Diabetic Foot Ulcers Grand Challenge 2020, 2020, <https://doi.org/10.5281/zenodo.3715016>.
8. X. Zhu, H. Hu, S. Lin, J. Dai, Deformable ConvNets V2: more deformable, better results, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 9300–9308, https://doi.org/10.1109/ CVPR.2019.00953.
9. Z. Cai, N. Vasconcelos, Cascade R-CNN: high quality object detection and instance segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 1483–1498, <https://doi.org/10.1109/tpami.2019.2956516>
10. J.H. Ng, M. Goyal, B. Hewitt, M.H. Yap, The effect of color constancy algorithms on semantic segmentation of skin lesions, in: Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging, International Society for Optics and Photonics, SPIE, 2019, pp. 138–145, https://doi.org/ 10.1117/12.2512702

**BIBLIOGRAPHY:**

**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:**

**Installing Python:**

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



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

**Installing PyCharm:**

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



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
5. Wait for the installation to finish.
6. 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”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. 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



# **INTRODUCTION TO PYTHON**

* Python

### What Is a 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 Possum 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 Possum 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](https://github.com/fogleman/Minecraft) 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

Humpy’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 numbly 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 sub classing 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

I’ve talked about exceptions before but now I will talk about them in depth. 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/elf 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 tinplating
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](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, tinplating 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. Flash 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 Reedit 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](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

2. SYSTEM STUDY

### 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 TESTS

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 satisfaction, as shown by successfully 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 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.

6.1 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.

# 6.2 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.

6.3 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.