Leafy AI

A project report submitted in partial fulfillment of the requirements for the degree

Of

BACHELOER OF SCIENCE IN INFORMATION TECHNOLOGY

Of

JORHAT INSTITUTE OF SCIENCE AND INFORMATION TECHNOLOGY

UNDER DIBRUGARH UNIVERSITY

2020



Submitted by

**NIHAR KASHYAP (Roll. No. 32)**

**PRACHURJYA GOGOI (Roll. No. 38)**

Guided by

HSUVAS BORKAKOTY JAMESON MUSHAHARY

INTERNAL GUIDE HEAD OF THE DEPARTMENT

**DEPARTMENT OF COMPUTER SCIENCE & IT**

**JORHAT INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JORHAT 785010**

**ASSAM**

**CERTIFICATE**

This is to certify that the project entitled “LeafyAI” of JIST submitted to Jorhat Institute of Science and Technology is partial fulfillment for the completion of the major project B.Sc in Information Technology Degree under Dibrugarh University. It is an original work carried out by Nihar kashyap and Prachurjya Gogoi Bsc IT 6th Semester under the guidance of Mr. Hsuvas Borkakoty of CS & IT Department, JIST.

Signature of Guide Signature of HOD

(Mr. Hsuvas Borkakoty) (Mr. Jameson Mushahary)

CS & IT Department, JIST CS & IT Department, JIST

**CERTIFICATE OF EXTERNAL**

This is to certify that Nihar Kashyap(Roll No 32) and Prachurjya Gogoi(Roll No 38) student of 6th semester, Department of Computer Science and Information Technology, Jorhat Institute of Science and Technology has carried out his Project work entitled “Plant Disease Management System” under the guidance of Mr. Hsuvas Borkakoty, Assistant Professor, Department of Computer Science and Information Technology, Jorhat Institute of Science and Technology, for the partial fulfillment of the requirement for the award of degree of Bachelor of Science in Information Technology.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner

Date: Date:

Place: Place:

**ACKNOWLEDGEMENT**

The presented project work is a humble and maiden effort of the work concerned with “LeafyAI” and has been a large success for the generous help and guidance received by us from several persons and quarters.

We are greatly indebted to Mr. Hsuvas Borkakoty, Department of Computer Science and Information Technology for ample guidance and encouragement which was an unfailing source of rewards experience.

We would also like to convey our sincere thanks to all who helped and encouraged us during the course of the project. Finally, we thank our parents for providing us with the means to complete this project and above all God for his blessings.

Thanking You

Nihar Kashyap

Prachurjya Gogoi

B.Sc IT 4th Semester

Jorhat Institiute of Science and Technology

**TABLE OF CONTENTS**

TITLE PAGE

Introduction………………………………………………………………………….

A Note on Machine Learning……………………………………………………….

Requirement Analysis……………………………………………………………….

Vgg16 Architecture………………………………………………………………….

**INTRODUCTION**

Agriculture is more than a feeding source for India; it is also the main source of economy for a huge section of people. Profit in the agricultural sector depends on high productivity. However pests and diseases pose a major threat to crops. Early treatment of these diseases is very important to save the crops from extensive damage. This is where early detection systems come into play. It is physically impossible for farmers to inspect each and every corner of the field. To overcome this problem we developed an image based disease detection system. This project titled ‘LeafyAI’ is Machine Learning based plant disease management system. It uses deep learning to predict the disease of a plant from the picture of an infected leaf. LeafyAI can predict the disease with 95 % accuracy and show the symptoms and remedies associated with the disease through a graphical user interface. If the user seeks further help he/she can post a question in the Forum section of the website. Our team of moderators and other expert users are dedicated to answer any questions posted on the forum as soon as possible.

**GENERAL TERMINOLOGIES RELATED WITH THE PROJECT**

CNN (Convolutional Neural Netwok) - It is a way of extracting features from images and make the computer understand them. A convolutional neural network is constructed of multiple convolutional layers. After each layer, we end up with a feature map that will be passed to the next layer. The first convolutional layer will only extract simple features such as edges in different orientations. When we go deeper in the network, the features will become clear parts of the object.

VGG16 (Visual Geometry Group 16) - VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. This model won the 1stand 2nd place in 2014 ILSVRC challenge.

Dataset - Collection of excel sheets, images, audio or video used to train the neural network. The quality of data present in the dataset is very important for the neural network to train properly.

Keras - It is an open-source neural-network library written in Python. Keras is capable of running on top of [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow), [Microsoft CognitiveToolkit](https://en.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Theano](https://en.wikipedia.org/wiki/Theano_(software)),or [PlaidML](https://en.wikipedia.org/wiki/PlaidML). Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

**REQUIREMENT ANALYSIS**

During this phase the requirements for completion of a project are identified. The tools specifically used for building this system are:

Hardware:

CPU: Intel Core i5 – 7300HQ @ 2.50 GHz

RAM: 4 GB DDR4

GPU: Nvidia GeForce GTX 1050

Hard Disk: 1 TB

Software:

Front End: HTML, CSS, JavaScript

Back End: Python Django

The requirements for this system are:

Any laptop or Desktop with mouse and keyboard accessibility

Processor: Intel i3 or above

Memory: Minimum 4 GB

Storage: 2 GB available hard disk space

Python Packages: Keras, TensorFlow

**FEASIBILITY STUDY**

Feasibility is the study of impact which happens in the organization by the development of the system. The impact can be either positive or negative. When the positive nominate the negative then the system is considered feasible. Here the feasibility study can be performed in three ways such Technical Feasibility, Economical feasibility, Operational Feasibility and behavioural feasibility.

1. ECONOMIC FEASIBILITY:-

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with cost, decisions is made to design and implement the system.

This part of feasibility study gives the top management the economic justification for the new system. This is an important input to the management because very often the top management does not get compounded by the various technicalities that bound be associated with a project of this time. A simple economic analysis that gives the actual comparison of cost and benefits is or much more meaningful in such cases.

The project we developed is economically feasible on the user’s part as it does not require anything more than a Smartphone or Computer and a stable internet connection to operate. On top of that the project was optimized to consume less data and load frequently required data from the cache memory. As such the bandwidth requirement of the project was also reduced. The admin part follows the same principle of low bandwidth and data requirement.

**2)** TECHNICAL FEASIBILITY:-

Technical feasibility centres on the technical existing manual system of the test management process and to what extent it can support the system. According to feasibility analysis procedure the technical feasibility of the system is analysed and the technical requirements such as software facilities, procedure, inputs are identified. It is also one of the important phases of the system development activities.

The system was designed with the latest and best technologies available on the market. It is fast, user friendly and easy to maintain. This project uses Python and SQL to function which are common platforms to work with now days. Technical staff should have no problem modifying and maintaining the project as and when required. With the bloom in web based services in the recent years this system is not only technically feasible but cost effective as well.

**3)** BEHAVIOURAL FEASIBILITY:-

This includes the following questions:

* Is there sufficient support for the user?
* Will the proposed system cause harm?

It is found that the most of the citizens of this country are well versed with web services and computers. Along with that the project provides tooltip at necessary places to help the user find his/her way around.

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and concluded that the system is feasible.

**SOFTWARE REQUIREMENT SPECIFICATION**

A software requirements specification (**SRS**) is a **document** that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs. For smaller problems or problems that can easily be comprehended; the specification activity might come after the entire analysis is complete. However, it is more likely that problem analysis and specifications are done concurrently. The transition from analysis should also not be expected to be straight forward, even if some formal modelling is used during analysis. Essentially what passes from requirement analysis activity to the specification activity is the knowledge acquired about the system. The modelling is essentially a tool to help obtain a thorough and complete knowledge about the system.

ANALYSIS OF FACTUAL DATA:

Analysis of data is a process of inspecting, cleaning, transforming, and modelling data with the goal of highlighting useful information, suggesting conclusion, and supporting decision making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science and social science domains.

Data mining is a particular data analysis technique that focuses on modeling and knowledge discovery foe predictive rather than purely descriptive purposes.

IDENTIFICATION OF ESSENTIAL REQUIREMENTS:

Identification of the essential requirement is an important task in developing of the project. In this system the essential requirements are identified through surveying. By surveying, the important needs of the user in our web application are known. In the surveying, the different possibilities of tour information that have to be included in the system is given by questionnaire.

Questions included like:

1. Need to change to the UI for better usability.
2. Is it advantageous over manual methods?
3. Need of better accuracy in detection.
4. Need for image uploading system

SELECTION OF REQUIREMENT STRATEGIES:

From the survey analysis graph it is clear that which are all the requirements that the user requires the most. It is decided to include the required information and omit the less priority ones.

DEFINATION OF PROCESSING REQUIREMENTS:-

The user interface for this system will have to be simple and clear. Most importantly, the pages must be easy to read, easy to understand, accessible. The colour scheme should be appropriate to provide familiarity with college.

There are many function the system can perform and this must be logically grouped or displayed in a intuitive order to allow the user to perform task quickly and efficiently, without getting lost in excessive amounts of text.

The system must also display large amount of information and to avoid confusion this must be displayed in categories or in different pages. Furthermore, a small amount of information may be displayed initially, for e.g. in a certain limit on date or amount, and the ability to view more in depth information on the subject should be apparent.

The system will provide different views for different users, allowing multiple access levels. For example, a standard user will only be able to see their own details and details of their own applications, whereas the administrator will be able to view all users, applications and statistics and will have many more privileges. Being an online system, it will naturally be viewable from any computer or smartphone with internet connection. This will provide far more accessibility than if it were written only for one specific platform such as Windows or Android.

OBJECTIVE OF SRS:-

The objective of this SRS document is to specify software requirements of the Online Application for the designers. It is intended to be a complete specification of what functionality the website provides. The main purpose of the system is to build a disease management system which can detect and provide solution to diseases in a fast and easy manner. Additionally it provides a platform for both agricultural experts and non-experts to mutually solve each other’s problems.

OVERVIEW OF SRS:-

The SRS document will include two sections.

**Overall Description** will describe major components of the system, interconnection and external interfaces.

**Specific Requirements** will describe the functions of actors, their role in the system and constraints.

OVERALL DESCRIPTION:

The SRS document will give further details on the overall product description, including the hardware, software, and communication interfaces, product functions, user characteristics, and any assumptions that will be made.

SPECIFIC REQUIREMENTS:

The SRS document will also include the specific requirements needed. This will include the function, performance, design, and software requirements. This document is organized in a logical manner and is easy to follow. Reader’s should refer to the table of content, appendices, or index if looking for something in specific. Otherwise, reading this document from start to finish will start with a vague description and get more specific in detail as changing section and reading further.

Software requirements: -

|  |  |
| --- | --- |
| OS-Platform | Windows 7,8,10, Linux, Android |
| Front-end | HTML, CSS, JavaScript |
| Back-end | Python Django |
| IDE | Visual Studio |
| Programming Language | Python |
| Additional Package | Tensorflow, RemoveBG, Keras |

Hardware requirements: -

|  |  |
| --- | --- |
| RAM | 4 GB DDR4 |
| Processor | Intel Core i5 – 7300HQ @ 2.50 GHz |
| GPU | Nvidia GeForce GTX 1050 |
| Storage | 2 GB available hard disk space |

**PROBLEM STATEMENT AND OBJECTIVE OF PROJECT**

**Problem Statement-**

“To design a disease management that will predict the disease from a given image and show the name of the disease along with symptoms and remedies to the user through a graphical user interface”

**Objective-**

The project tries to fulfill the following objectives –

* To create a Machine Learning Model to detect diseases in the leaves of Potato, Tomato, Corn, Pumpkin and Cherry
* Promote early detection of devastating crop diseases for the benefit of farmers.
* To provide suitable Remedy and Symptoms for the detected disease.
* Provide a graphical Interface for the user to upload image of the diseased leaf and retrieve the result.
* Provide a Forum Section for Users to ask their doubts and seek help relevant to the scope of the project

**A NOTE ON MACHINE LEARNING**

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. This project made extensive use of machine learning to achieve the objectives stated above.

**What is a CNN?**

Convolutional neural networks refer to a sub-category of neural networks: they, therefore, have all the characteristics of neural networks. However, CNN is specifically designed to process input images. Their architecture is then more specific: it is composed of two main blocks.

The first block makes the particularity of this type of neural network since it functions as a feature extractor. To do this, it performs template matching by applying convolution filtering operations. The first layer filters the image with several convolution kernels and returns “feature maps”, which are then normalized (with an activation function) and/or resized.

This process can be repeated several times: we filter the features maps obtained with new kernels, which gives us new features maps to normalize and resize, and we can filter again, and so on. Finally, the values of the last feature maps are concatenated into a vector. This vector defines the output of the first block and the input of the second.

The second block is not characteristic of a CNN: it is in fact at the end of all the neural networks used for classification. The input vector values are transformed (with several linear combinations and activation functions) to return a new vector to the output. This last vector contains as many elements as there are classes: Element i represents the probability that the image belongs to class i. Each element is therefore between 0 and 1, and the sum of all is worth 1. These probabilities are calculated by the last layer of this block (and therefore of the network), which uses a logistic function (binary classification) or a softmax function (multi-class classification) as an activation function.

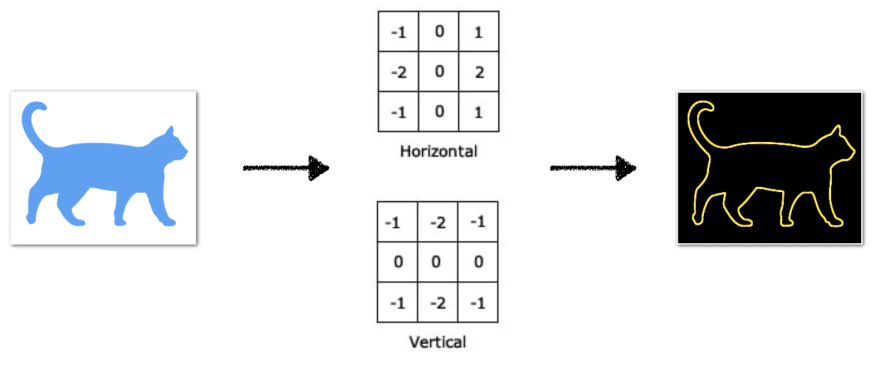
**Features in Machine Learning**

 A feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in pattern recognition, classification and regression. Feature extraction involves reducing the number of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power; also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples.

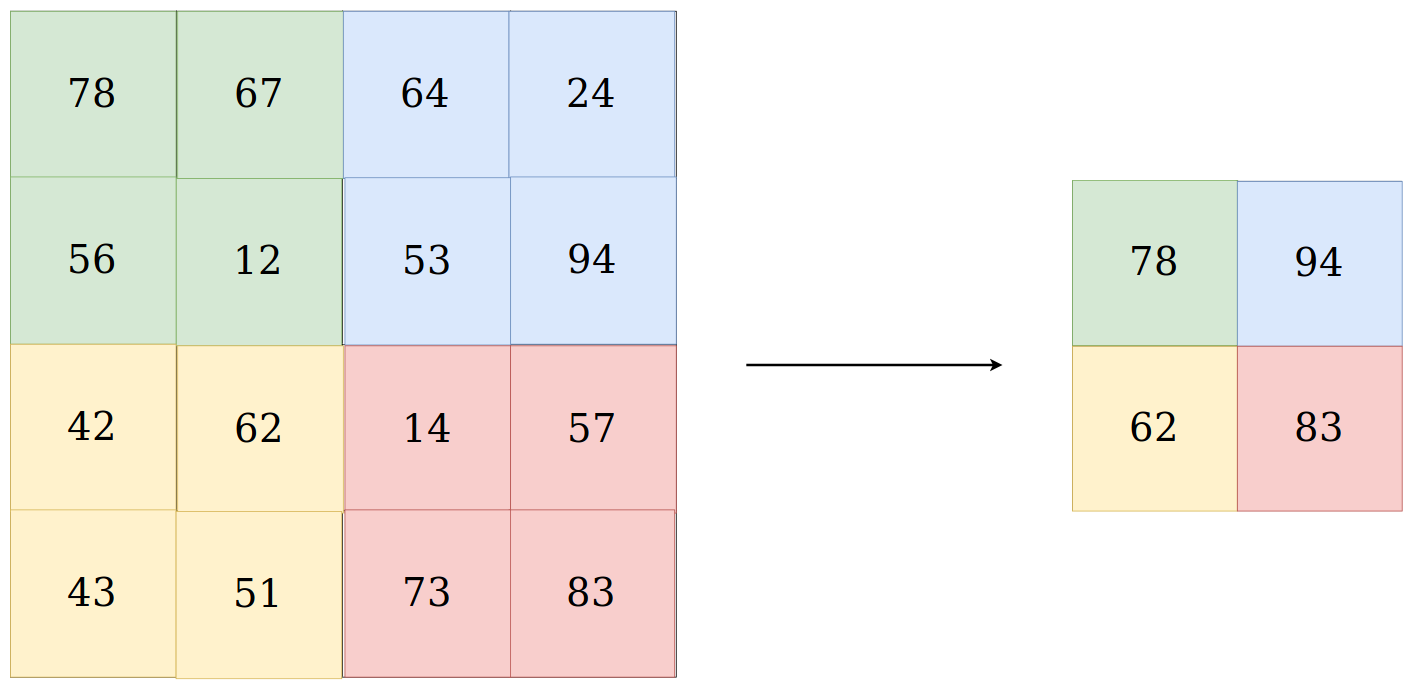
CNN works by storing features of images into an n-dimensional vector called feature vector. This feature can be as simple as pixel values of the image to more complex ones like hough transform. For our project features that were important for us were color, shape, texture, contours of diseases in the leaf. A reason why CNN is exceptionally good at working with images is that the large number of parameters present in the neural network allows it to “see” even the tiniest of features in the image; without us explicitly telling it to.

**Layers of a CNN –**

1) The convolutional layer - The first layer in a CNN is always a Convolutional Layer. The convolutional layer is the key component of convolutional neural networks, Its purpose is to detect the presence of a set of features in the images received as input. The convolution layer’s parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.



2) The pooling layer – It is common to periodically insert a Pooling layer in-between successive Conv layers in ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.



More generally, the pooling layer:

Accepts a volume of size W1×H1×D1W1×H1×D1

Requires two hyperparameters:

* their spatial extent F,
* the stride S,

Produces a volume of size W2×H2×D2W2×H2×D2 where:

W2= (W1−F)/S+1W2=(W1−F)/S+1

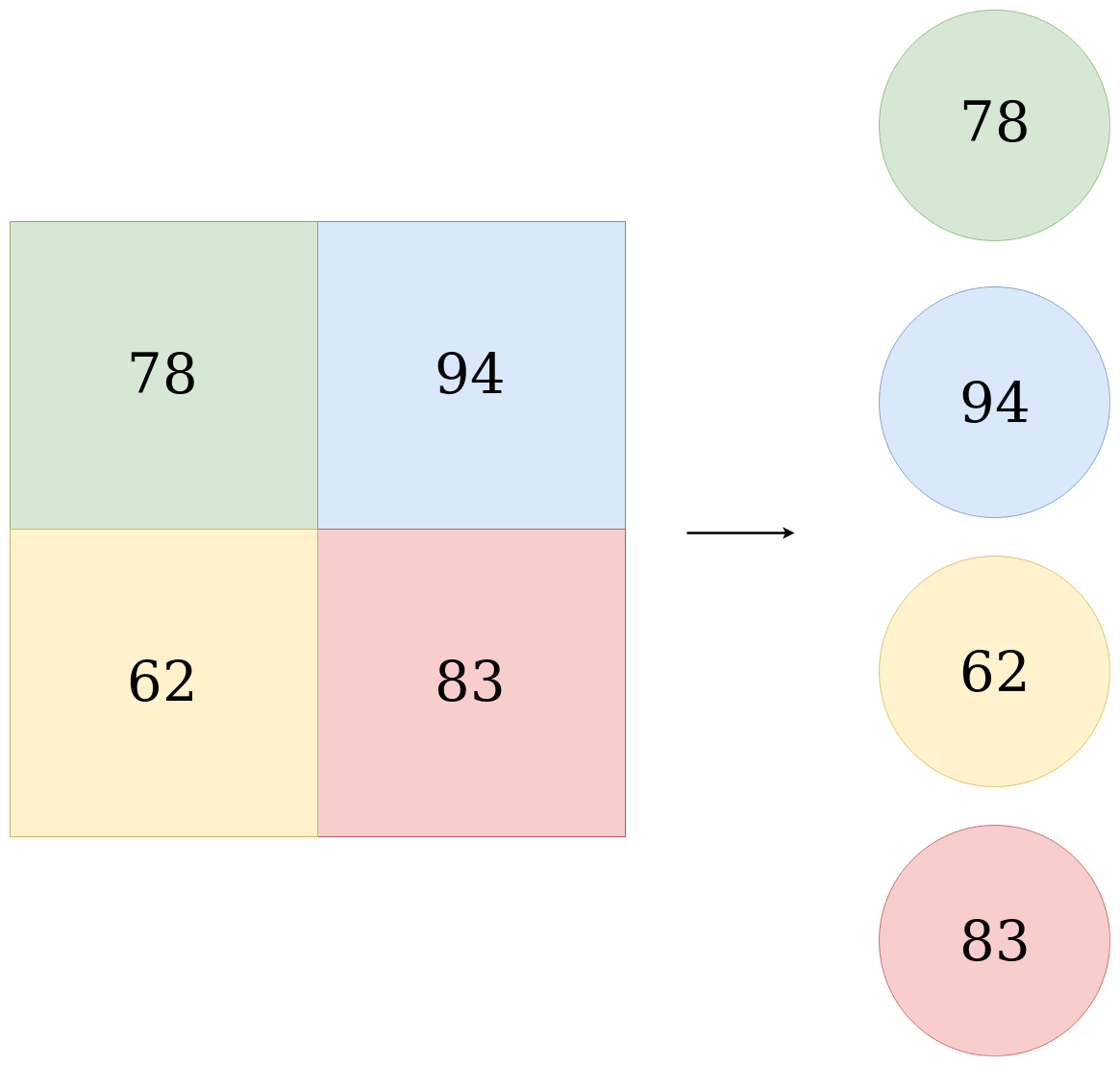
H2= (H1−F)/S+1H2=(H1−F)/S+1

D2=D1D2=D1

Introduces zero parameters since it computes a fixed function of the input

For Pooling layers, it is not common to pad the input using zero-padding.

3) Fully connected layer - Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.



**ACTIVATION FUCNTION**

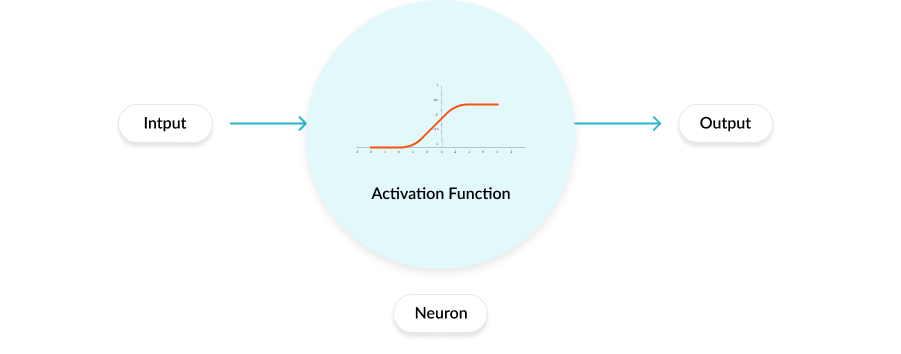
Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

**Role of the activation function in a neural network model**

In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer.

The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

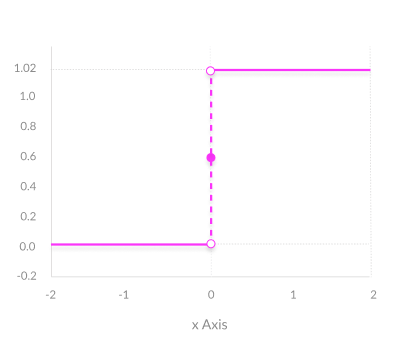


Increasingly, neural networks use non-linear activation functions, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.

**Types of activation function –**

1. Binary Step Function

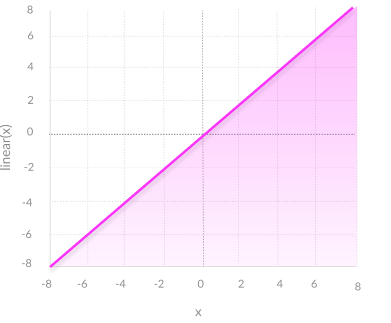
A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.



This function has the drawback of not able to classify more than two categories.

2. Linear Activation Function

It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input. In one sense, a linear function is better than a step function because it allows multiple outputs, not just yes and no.



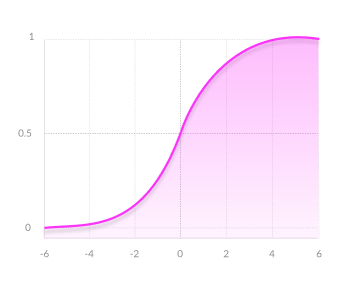
Although better than step functions this type of function cannot be used for back propagation.

3. Non-Linear Activation Functions

Modern neural network models use non-linear activation functions. They allow the model to create complex mappings between the network’s inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensionality.

Almost any process imaginable can be represented as a functional computation in a neural network, provided that the activation function is non-linear.

**Seven common nonlinear activation functions -**



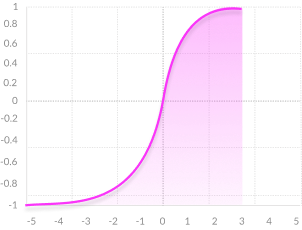
**1. Sigmoid / Logistic**

Advantages:

* Smooth gradient, preventing “jumps” in output values.
* Output values bound between 0 and 1, normalizing the output of each neuron.
* Clear predictions—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

Disadvantages:

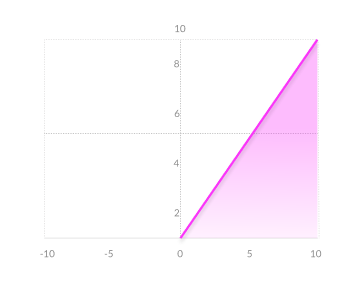
* Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
* Outputs not zero centered.
* Computationally expensive



**2. TanH / Hyperbolic Tangent**

Advantages:

* Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
* Otherwise like the Sigmoid function.



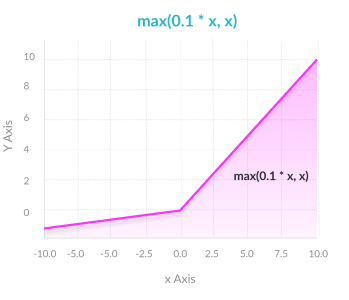
**3. ReLU (Rectified Linear Unit)**

Advantages:

* Computationally efficient—allows the network to converge very quickly
* Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

Disadvantages:

* The Dying ReLU problem—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.



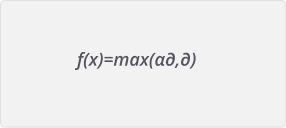
**4. Leaky ReLU**

Advantages:

* Prevents dying ReLU problem—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
* Otherwise like ReLU

Disadvantages:

* Results not consistent—leaky ReLU does not provide consistent predictions for negative input values.



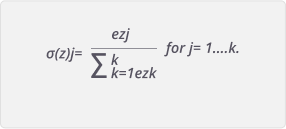
**5. Parametric ReLU**

Advantages:

* Allows the negative slope to be learned—unlike leaky ReLU, this function provides the slope of the negative part of the function as an argument. It is, therefore, possible to perform backpropagation and learn the most appropriate value of α.
* Otherwise like ReLU

Disadvantages:

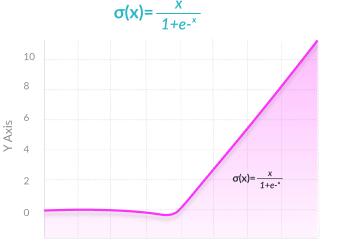
* May perform differently for different problems.



**6. Softmax**

Advantages:

* Able to handle multiple classes —normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.
* Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.



**7. Swish**

Swish is a new, self-gated activation function discovered by researchers at Google. It performs better than ReLU with a similar level of computational efficiency. In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.

**OPTIMIZATION FUNCTIONS**

The goal of machine learning and deep learning is to reduce the difference between the predicted output and the actual output. This is also called as a Cost function(C) or Loss function. Optimizers update the weight parameters to minimize the loss function. Loss function acts as guides to the terrain telling optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum.

**Gradient Descent**

Gradient descent is an iterative machine learning optimization algorithm to reduce the cost function. This will help models to make accurate predictions.

Gradient indicates the direction of increase. As we want to find the minimum point in the valley we need to go in the opposite direction of the gradient. We update parameters in the negative gradient direction to minimize the loss.

**Learning Rate**

Learning rate controls how much we should adjust the weights with respect to the loss gradient. Learning rates are randomly initialized.

Lower the value of the learning rate, slower will be the convergence to global minima.

A higher value for learning rate will not allow the gradient descent to converge

Since our goal is to minimize the cost function to find the optimized value for weights, we run multiple iterations with different weights and calculate the cost to arrive at a minimum cost

**Types of Optimizers**

**1. Momentum**

Momentum is like a ball rolling downhill. The ball will gain momentum as it rolls down the hill.

Momentum helps accelerate Gradient Descent(GD) when we have surfaces that curve more steeply in one direction than in another direction. It also dampens oscillations.

For updating the weights it takes the gradient of the current step as well as the gradient of the previous time steps. This helps us move faster towards convergence.

Convergence happens faster when we apply momentum optimizer to surfaces with curves.

**2. Nesterov accelerated gradient(NAG)**

Nesterov acceleration optimization is like a ball rolling down the hill but knows exactly when to slow down before the gradient of the hill increases again.

We calculate the gradient not with respect to the current step but with respect to the future step. We evaluate the gradient of the looked ahead and based on the importance then update the weights.

NAG is like we are going down the hill where we can look ahead in the future. This way we can optimize our descent faster. It works slightly better than standard Momentum.

**3. Adagrad — Adaptive Gradient Algorithm**

We need to tune the learning rate in Momentum and NAG which is an expensive process.

Adagrad is an adaptive learning rate method. In Adagrad we adopt the learning rate to the parameters. We perform larger updates for infrequent parameters and smaller updates for frequent parameters.

It is well suited when we have sparse data as in large scale neural networks. GloVe word embedding uses adagrad where infrequent words required a greater update and frequent words require smaller updates.

For SGD, Momentum, and NAG we update for all parameters *θ* at once. We also use the same learning rate η. In Adagrad we use different learning rate for every parameter *θ* for every time step *t*

Adagrad eliminates the need to manually tune the learning rate.

In the denominator, we accumulate the sum of the square of the past gradients. Each term is a positive term so it keeps on growing to make the learning rate η infinitesimally small to the point that algorithm is no longer able learning. Adadelta, RMSProp, and adam tries to resolve Adagrad’s radically diminishing learning rates.

**4. Adadelta**

* Adadelta is an extension of Adagrad and it also tries to reduce Adagrad’s aggressive, monotonically reducing the learning rate
* It does this by restricting the window of the past accumulated gradient to some fixed size of w. Running average at time *t* then depends on the previous average and the current gradient
* In Adadelta we do not need to set the default learning rate as we take the ratio of the running average of the previous time steps to the current gradient

**5. RMSProp**

* RMSProp is Root Mean Square Propagation. It was devised by Geoffrey Hinton.
* RMSProp tries to resolve Adagrad’s radically diminishing learning rates by using a moving average of the squared gradient. It utilizes the magnitude of the recent gradient descents to normalize the gradient.
* In RMSProp learning rate gets adjusted automatically and it chooses a different learning rate for each parameter.
* RMSProp divides the learning rate by the average of the exponential decay of squared gradients

**6. Adam — Adaptive Moment Estimation**

* Another method that calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients.
* It also reduces the radically diminishing learning rates of Adagrad
* Adam can be viewed as a combination of Adagrad, which works well on sparse gradients and RMSprop which works well in online and nonstationary settings.
* Adam implements the exponential moving average of the gradients to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients
* Adam is computationally efficient and has very little memory requirement
* Adam optimizer is one of the most popular gradient descent optimization algorithms

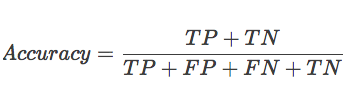
**7. Nadam- Nesterov-accelerated Adaptive Moment Estimation**

* Nadam combines NAG and Adam
* Nadam is employed for noisy gradients or for gradients with high curvatures
* The learning process is accelerated by summing up the exponential decay of the moving averages for the previous and current gradient

**PERFORMANCE METRICS USED IN MACHINE LEARNING**

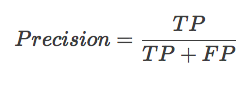
**1. Accuracy** — Ratio of correct predictions to total predictions made. Suitable only when there are about equal number of observations in each class.

Formula :



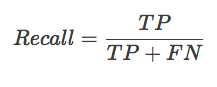
**2.** **Precision** — How many samples classified as positive class *(TP + FP)* i.e. “*Predicted Positive*”, truly belong to positive class *(TP)*.

Formula:



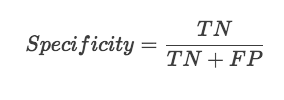
**3. Recall** — Of all the samples that are positive *(TP + FN) i.e. “Actually Positive”*, how many were correctly classified as positive *(TP)*.

Formula:

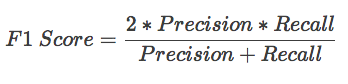


**4. Specificity** — Of all the samples that are negative *(TN + FP) i.e. “Actually Negative”*, how many were correctly classified as negative *(TN)*.

Formula:



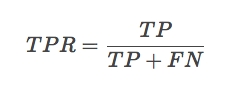
**5. F1 Score** — Harmonic mean of Precision and Recall. It is a single score that represents both Precision and Recall.



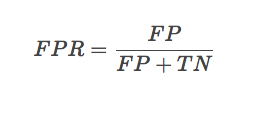
**6. AUC-ROC (Area Under Curve — Receiver Operating Curve)**

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. Axes for ROC are TPR (True Positive Rate) and FPR (False Positive Rate).

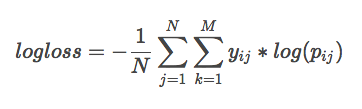
TPR Formula:



FPR Formula:



**7. Logarithmic loss** — Log Loss quantifies the accuracy of a classifier by penalizing false classifications. Mathematically Log Loss is defined as —



**TRANSFER LEARNING**

Transfer learning uses a pretrained neural network and modifies or adds extra neurons in order work with the given task. We used the VGG16 Neural Network which was pretrained to identify objects and patterns in images. We removed the last neuron layer and added our own neurons. The number of neurons to be added for optimum result was determined through hit and trial method. Transfer Learning is an effective way to train machine learning models when the number of available training sample is less.

**Data Collection**

The accuracy of a neural network largely depends on the quantity and quality of training data provided during the training phase.

Each Step of Data collection is given below-

Image Acquisition – We gathered about 1000 images of each disease for training and validation purpose. The images were downloaded from internet as well as taken manually from the fields of Assam Agricultural University. The authenticity of the disease in each image was verified by experts. Care was taken not to include inconsistent or unclear images that might cause the Neural Network to train improperly.

Image Segmentation – Image segmentation is the process of removing unwanted parts of the image. In our case the background of the images was removed as they did not have any significant contribution towards training. Moreover we were only interested in the diseased portion of a leaf so the corresponding healthy parts were removed as well. Further to increase the computational efficiency of our Model the dimension of the images were scaled down to 256 x 256.

Labeling – Once the images were processed they were put in folders which were named according to the class they belong to. The folders were separated as training and validation. The training folder contained 700 images of each disease while validation consists of 300 images of each class.

**Training**

Training a network is a process of finding kernels in convolution layers and weights in fully connected layers which minimize differences between output predictions and given ground truth labels on a training dataset. Backpropagation algorithm is the method commonly used for training neural networks where loss function and gradient descent optimization algorithm play essential roles. A model performance under particular kernels and weights is calculated by a loss function through forward propagation on a training dataset, and learnable parameters, namely kernels and weights, are updated according to the loss value through an optimization algorithm called backpropagation and gradient descent.

**VGG16 ARCHITECTURE**



VGG-16 is a convolutional neural network architecture proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. is trained on ImageNet dataset which has over 14 million images and 1000 classes, and achieves 92.7% top-5 accuracy. The name VGG-16 comes from the fact that it has 16 layers. Its layers consist of Convolutional layers, Max Pooling layers, Activation layers, Fully connected layers. Conv 1 has number of filters as 64 while Conv 2 has 128 filters, Conv 3 has 256 filters while Conv 4 and Conv 5 has 512 filters.

For our project we modified the VGG16 architecture such that it gives better results on our dataset. We removed the last pre trained layer and added the following layers:

* One Dense layer of 512 neurons and relu activation
* One final Dense layer of 11 neurons softmax activation

Convolution blocks 1 through 5 are primarily used as feature extractors. By the end of Conv 5 the image was transformed into a set of features. After that the tensor is flattened into a one dimensional array and fed into the classification layer.

The last Dense layer of 11 neurons was our classification layer that produced the prediction result. The number of neurons in this layer reflects the number of prediction classes we have. The softmax function associated with it gives us the output as a value between 0 and 1 for each class. This value corresponds to the probability of each class.

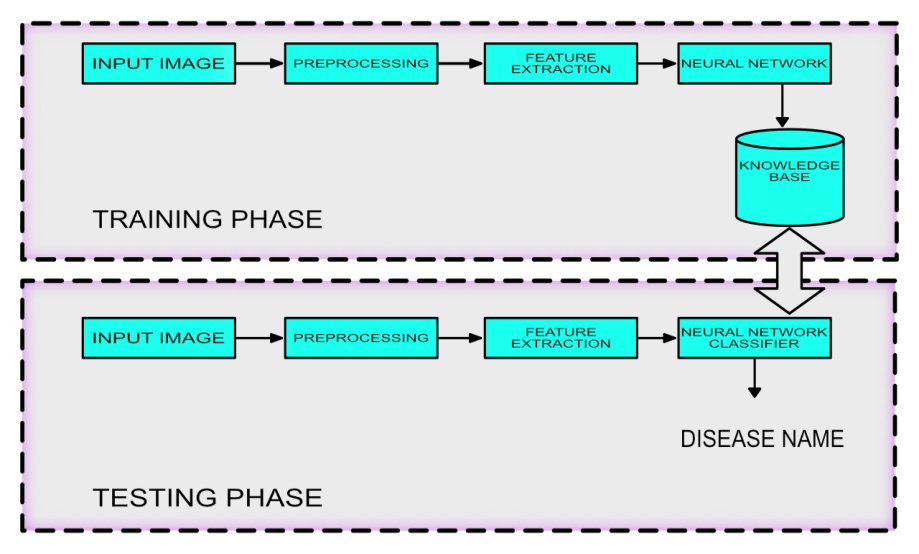
**METHODOLOGY**

To find out whether the leaf is diseased or healthy, certain steps must be followed. i.e., Preprocessing, Feature extraction, Training of classifier and Classification. Preprocessing of image is bringing all the images size to a reduced uniform size. Then comes extracting features of a preprocessed image which is done with the help of convolution layers. Extracted features can be as simple as edges, simple colors to more complex ones like entropy, energy etc. In our case more important features were the shape of leaf, the color and texture of the leaf.

The first few layers of the neural network are primarily focused on edge detection. At first it tries to determine the type of crop the leaf belongs to. As we move deeper more complex features are calculated.

The last layer is a fully connected layer to the end of the network. This layer basically takes an input volume and outputs an N dimensional vector where N is the number of classes that the program has to choose from. In our case N is 11 since we have 11 classes of crops for classification. Each number in this N dimensional vector represents the probability of a certain class. For example, if the resulting vector is [0 .75 .25], then this represents a 0% probability that the image belongs to class 1, 75% probability that the disease belongs to class 2 and 25% probability that the disease belongs to class 3

**Algorithmic Depiction**



**Training of Neural Network –**

Training for the Neural Network was carried through Keras library in Python. Before feeding our data to the neural network we need to make sure they are in the proper format. We used ImageDataGenerator method to resize our images into 224 by 224 pixels and separate them into batches of 10 images each. Separating them into batch is required because processing all the images at once will put considerable strain on the processor and memory. After that we load the pre trained vgg16 model weights and add two fully connected layer each of 512 and 11 neurons respectively. The neurons need to be connected into one single mesh for the algorithm to work. This step is called compiling the neural network. Compiling requires the optimizer type to use, the loss function and learning rate to train the model. Through extensive trial and error we found out that the Stochastic Gradient Descent(SGD) optimizer with a learning rate of 0.001 works best for our model. The loss function; since it is a multiclass classification needs to be categorical\_crossentropy. After all the hyperparameters are set correctly we trained the model using model.fit() function for 400 epochs. Training took approximately 7 days to complete. Testing the neural network using real world data is very important to assess its performance. So we took a set of 10 images of each class that the neural network has never seen before. And then passed on these images for classification using the model.predict() function. The image went through the usual pre processing steps before classification. The results of this testing are mentioned below in the Results section

**IMPLEMENTATION AND RESULTS**

**Inputs during Training –**

Training was carried out using the above mentioned python script. As training data 7998 images were fed as input to the program. The images belonged to 11 classes namely Cherry Powdery Mildew, Corn Common Rust, Corn healthy, Corn Northern Leaf Blight, Potato Early Blight, Potato Healthy, Potato Late blight, Squash powdery mildew, Tomato healthy, Tomato Late blight, Tomato Yellow Leaf Curl virus.

The images were programmatically reduced to (224,224) dimensions. Further the images were rotated and sheared by different factors to produce variations which would increase the number of training samples. We trained for 400 epochs or till the model stopped improving whichever was earlier. In our case the peak accuracy was attained at 250th epoch. The script was kept running for 7 days straight.

**Inputs during testing -**

At the end of training our model scored an accuracy of 94 % in training and 90 % in validation. With this accuracy we moved to testing the model with real world data. We gathered images of crops that the model has never seen before. With a prediction() function the images were scaled into the required dimensions and fed one by one into the model and the top predicted label was recorded. Testing on real world images the model gave an accuracy of 80%. Further summarizations of class based results are given in the results subsection. It may be noted that the model performed better when the images had uniform colored background in them, preferably black.

**Results –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Serial Number | Image Class | True Prediction | False Prediction | Total Images | Accuracy |
| 1 | Cherry Powderry mildew | 10 | 0 | 10 | 100% |
| 2 | Corn Common rust | 9 | 1 | 10 | 90% |
| 3 | Corn healthy | 9 | 1 | 10 | 90% |
| 4 | Corn Northern Leaf blight | 9 | 1 | 10 | 90% |
| 5 | Potato Early blight | 10 | 0 | 10 | 100% |
| 6 | Potato healthy | 8 | 2 | 10 | 80% |
| 7 | Potato Late blight | 9 | 1 | 10 | 90% |
| 8 | Squash Powdery mildew | 10 | 0 | 10 | 100% |
| 9 | Tomato healthy | 9 | 1 | 10 | 90% |
| 10 | Tomato Late blight | 10 | 0 | 10 | 100% |
| 11 | Tomato Yellow leaf curl virus | 8 | 2 | 10 | 80% |

**SYSTEM TESTING**

Every application needs to be thoroughly tested to eliminate runtime errors. This is a significant step in the software development cycle. LeafyAI was tested through two stages. The first one being white box testing and the second was second black box testing. The tests are described below:

**White Box Testing:**

This testing was carried out during the development phase exclusively by the developers. The modules were tested to see if they correctly perform the functionality meant for. Codes of each module were also checked to correct logical errors in the code. Optimizing the code was one of the chief goals of this phase. Duplicate codes were removed with clean non-repetitive functions.

**Black Box Testing:**

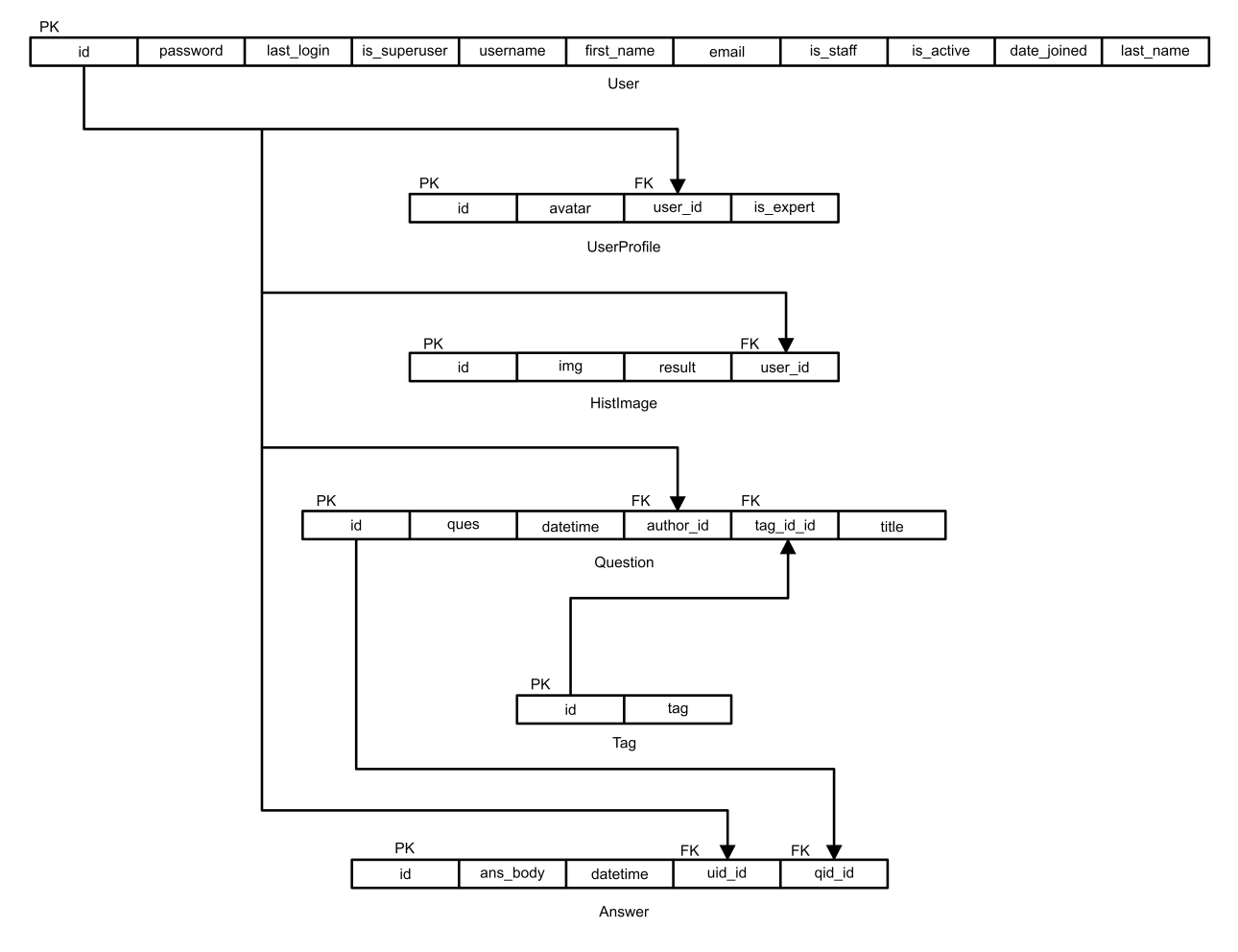
During this phase the application was distributed to a closed community of Users such as the guide for this project, friends and family members. All the participants here had little to no knowledge about the workings of the system. They were simply asked to provide inputs and record their observations. The main goal of this phase was to identify corner cases and invalid inputs.

Summary of the test carried out is given below:

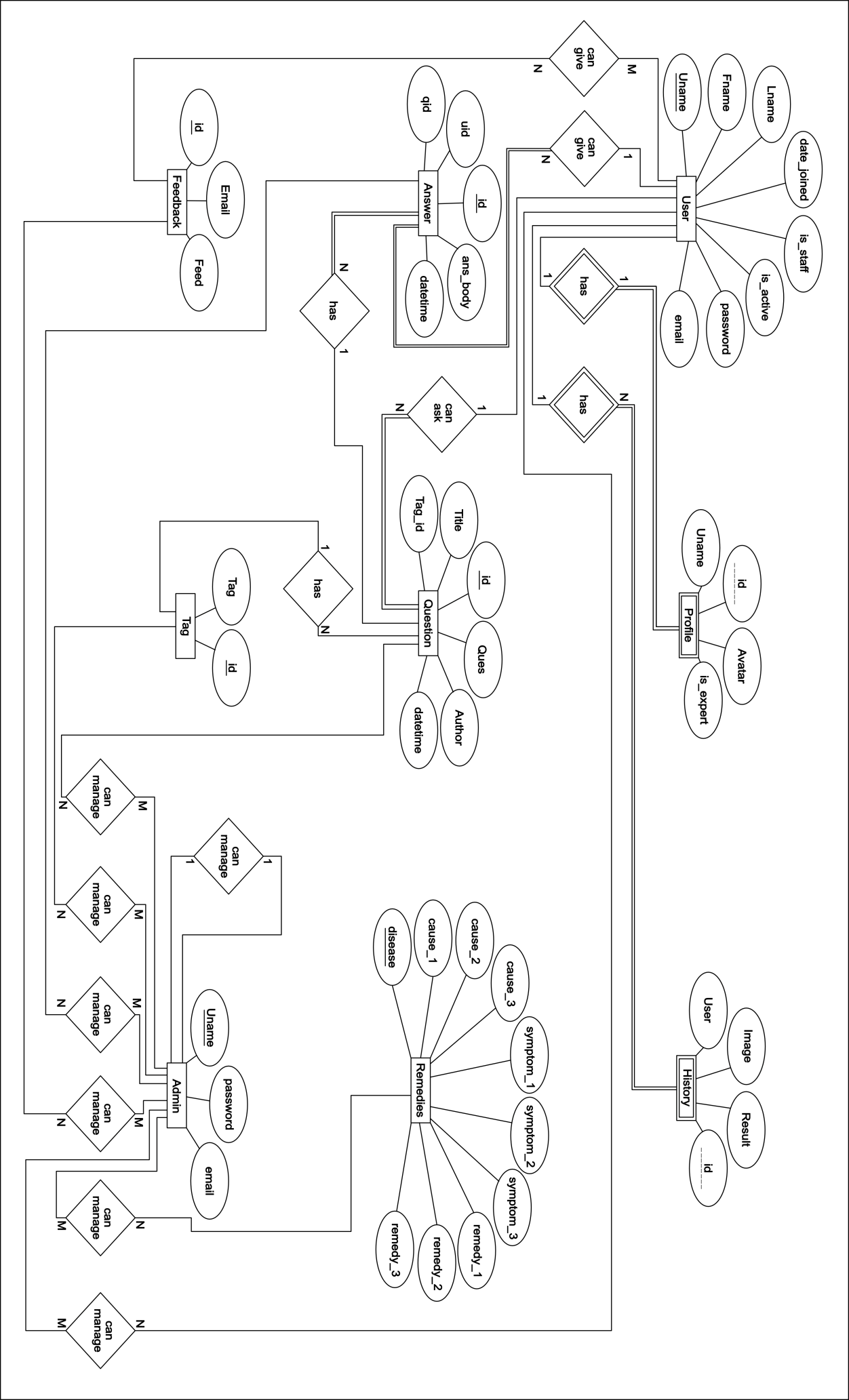
|  |  |  |  |
| --- | --- | --- | --- |
| Functions | Test cases Executed | Test cases Passed | Percentage of pass |
| Login | 100 | 90 | 90% |
| Register | 100 | 92 | 92% |
| Prediction | 100 | 90 | 90% |
| Edit profile | 100 | 95 | 95% |
| Feedback | 100 | 100 | 100% |
| Post Question | 100 | 96 | 96% |
| Edit Question | 100 | 99 | 99% |
| Delete Question | 100 | 100 | 100% |
| Post Answer | 100 | 97 | 97% |
| Edit Answer | 100 | 99 | 99% |
| Delete Answer | 100 | 100 | 100% |

**ANALYSIS AND DESIGN**

**Relationship Model**

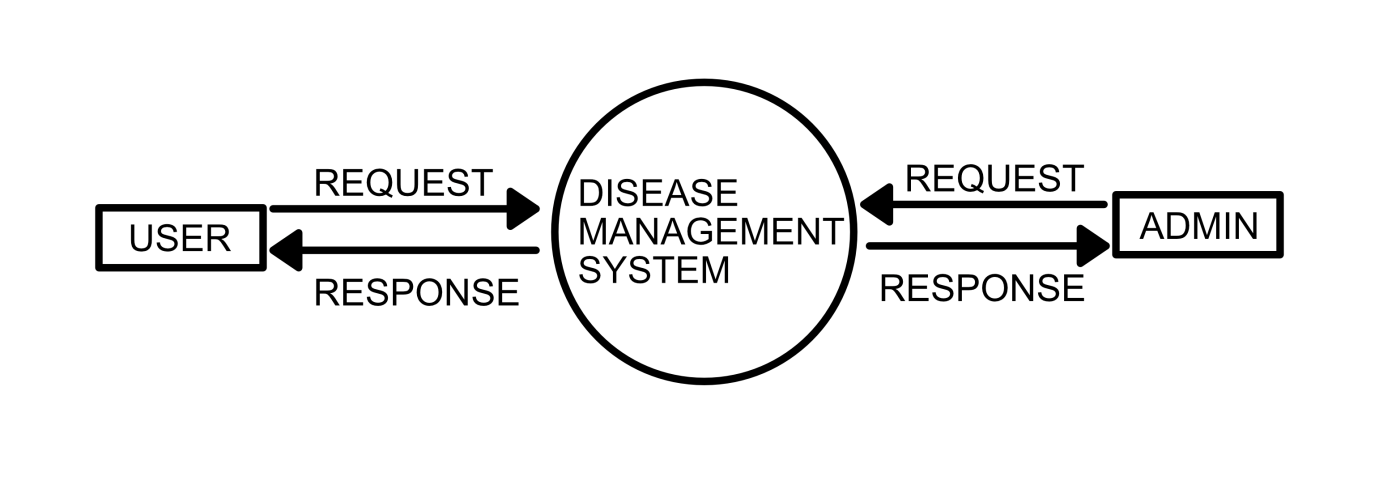


**Entity Relationship Diagram**

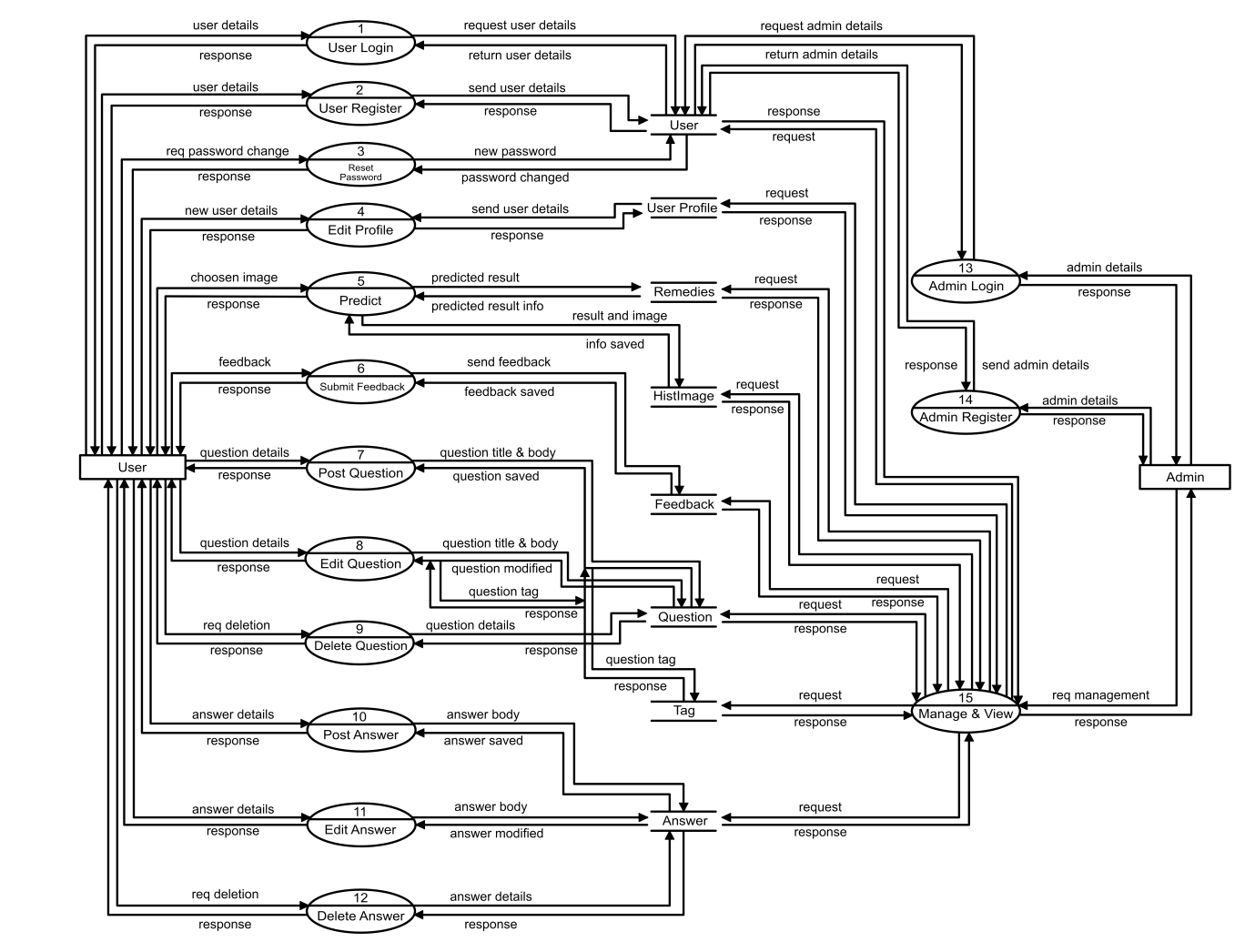


**DATAFLOW DIAGRAM**

**DFD Level 0**

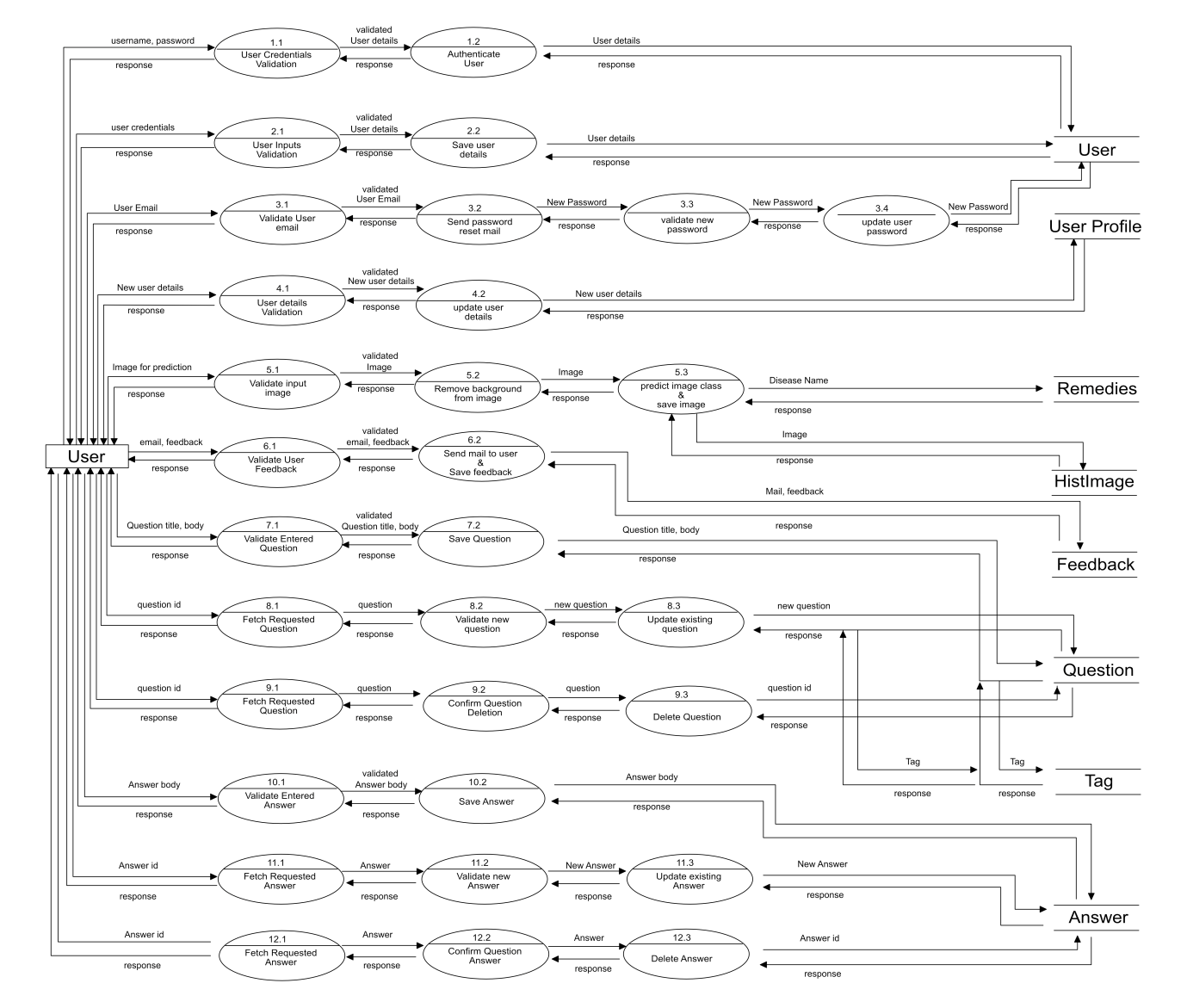


**DFD Level 1**

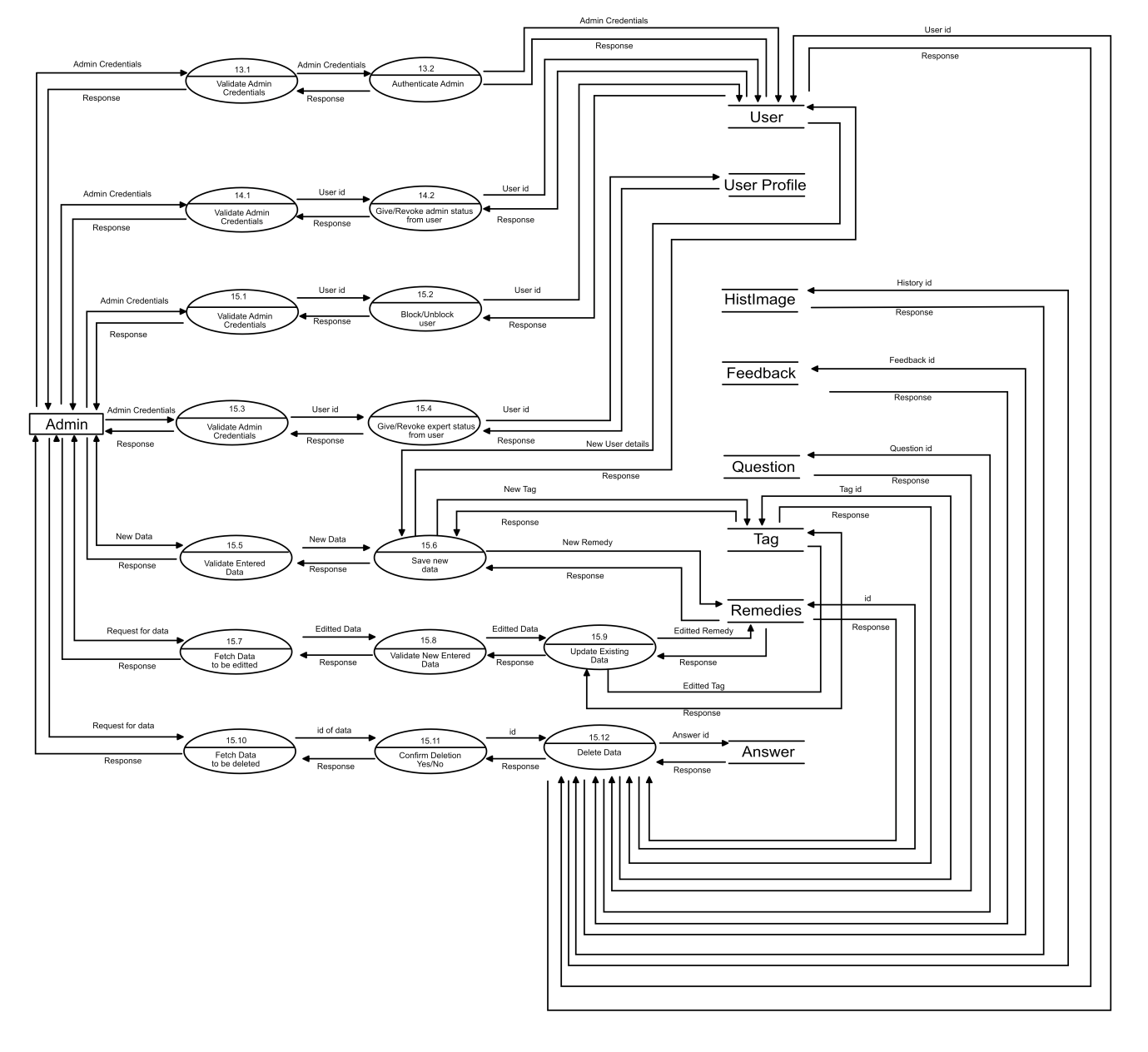


**DFD Level 2**

**Level 2 DFD for User**



**Level 2 DFD for Admin**



**SNAPSHOTS OF WEBSITE**

**BIBLIOGRAPHY**

* D. Lee, Kent, 2014. Python Programming Fundamentals. Springer - Verlag London 2014. pp 115 -136
* Cleophas, Tony J and Zwinderman, Aeilko H., 2015. Machine Learning in Medicine- A Complete Overview. Springer International Publishing Switzerland 2015. pp 445- 469
* Burger, Wilhelm. Burge, Mar J., 2016. Digital Image Processing. Springer - Verlag London 2016, 2018. pp 37 – 406
* Skansi, Sandro, 2018. Introduction to Deep Learning. Springer International Publishing AG, part of Springer Nature 2018. pp 51 - 130
* Aggarwal Charu C. Neural Networks and Deep Learning. Springer International Publishing AG, part of Springer Nature 2018. pp 17- 373
* Elmasri Ramez and Navathe Shamkant B. Fundamentals of Database Systems. Pearson Education 2011. pp 29 – 49
* Sun Xiaolin and Wei Jiangshu. 2019. Identification of maize disease based on transfer learning. IOP Publishing 2019.
* Ferentinos Konstantinos P. 2018. Deep learning models for plant disease detection and diagnosis. Elsevier B.V. 2018.
* Saha Anubhav, Krisabh, Dey Sougata, Hema D. Deva. 2019. Mulberry Leaf Disease Detection using Deep Learning. Blue Eyes Intelligence Engineering & Sciences publication 2019. pp 3368 - 3369

**WEB REFERENCES**

* <https://docs.djangoproject.com/en/3.0/>
* <https://medium.com/@randylaosat/a-beginners-guide-to-machine-learning-dfadc19f6caf>
* <https://www.youtube.com/watch?v=g-aS9oVY-DA>
* <https://medium.com/fintechexplained/how-to-fine-tune-your-machine-learning-models-to-improve-forecasting-accuracy-e18e67e58898>
* <https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a>
* <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>
* <https://towardsdatascience.com/how-to-deploy-a-machine-learning-model-dc51200fe8cf>
* <https://medium.com/machine-learning-through-visuals/performance-metrics-machine-learning-through-visuals-84ebd7cdcc20>
* <https://stackoverflow.com/questions/tagged/keras>
* <https://www.tensorflow.org/api_docs/python/tf/keras>
* <https://libguides.cu-portland.edu/Chicago_style/bibliography>