

PLANT DISEASE DETECTION USING CONVOLUTIONAL NEURAL NETWORK

Submitted in partial fulfilment of the requirements
for the award of
Master of Engineering degree in Computer Science and Engineering

by

SOURAV RAY (40310005)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bona fide work of **Sourav Ray (40310005)** who carried out the project entitled "**Plant Disease Detection using convolutional Neural Network**" under my supervision from November 2021 to May 2022.

Internal Guide

Dr. B. BHARATHI, M.E., Ph.D.,

HoD

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **SOURAV RAY(Reg. No. 40310005)** hereby declare that the Project Report entitled “**PLANT DISEASE DETECTION USING CONVOLUTIONAL NEURAL NETWORK**” done by me under the guidance of **Dr. B. BHARATHI, M.E., PH.D.**, (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer Science Engineering.

DATE:

**PLACE:
OFTHECANDIDATE**

SIGNATURE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T. Sasikala M.E., Ph.D., Dean, School of Computing, Dr.S. Vigneshwari M.E., Ph.D. and Dr.L. Lakshmanan M.E., Ph.D. , Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.B. Bharathi, M.E, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. In our base paper image segregation and identification is done using matrix factorization and machine learning. But this create a trained model using a Convolutional Neural Network where the user-given image is detected by converting and comparing the images which are in matrix form. In our application, the user image is compared with our trained model to identify the specifics of images and sent to the client where machine learning helps to find the right fertilizers for the crop based on disease recognized in the image given by the user in the first place. This proposed method is way efficient and effective compared to traditional image recognition. And our application use weather report for any given location and soil report - suggestion for the cultivation of specific crop for best productivity.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No
	ABSTRACT	IV
	LIST OF FIGURES	V
1	INTRODUCTION	1
	1.1 CONVOLUTIONAL NEURAL NETWORK ALGORITHM	3
	1.3 SOFTWARE AND PROGRAMING TOOL	12
	1.4 PROGRAM STRUCTURE	14
	1.5 PLANT DISEASES	18
	1.6 MINERAL FERTILIZER	24
2	LITERATURE SURVEY	29
3	PROJECT OVERVIEW	44
	3.6 OBJECTIVE	44
	3.7 MOTIVATION	45
	3.8 EXISTING SYSTEM	46
	3.9 PROPOSED SYSTEM	47
	4.0 TYPES OF DISEASE LEAF	48
4	METHODOLOGY	49
	4.1 OVERALL ARCHITECTURE	49
	4.2 LANGUAGE DESCRIPTION	58
	4.3 HOW PYTHON INTERPRETER WORKS	58
	4.4 GUIDELINES FOR PYTHON CODE	59
	4.5 CHARACTERISTICS OF PYTHON	61

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
4.6	MYSQL DATABASE	63
4.7	WORKING OF MYSQL	64
4.8	CHARACTERISTICS OF MYSQL	65
4.9	AIRTIFICIAL INTELLIGENCE	66

TABLE OF CONTENTS

Chapter No.	TITLE	Page No
5	RESULT AND DISCUSSION	70
6	CONCLUSION AND FUTURE WORK	74
	REFERENCES	75
	APPENDIX	78
A.	PROJECT JOURNAL	

LIST OF FIGURES

Figure No.	Figure Name
Fig 4.1	Plant Collaboration
Fig 4.5	How Python interpreter work
Fig 4.4	Plant Use Case Diagram

CHAPTER 1

INTRODUCTION

This application plays a vital role in recognizing the crop's leaves image and finding out the disease cause and fertilize suggestions after comparing with the trained model using Convolution Neural Network helps to differentiate each image by converting them by process of matrix convolution. CNN represents an interesting method for adaptive image processing and forms a link between general feed-forward neural networks and adaptive filters. Any user who grows different crops may not know all the diseases caused by fungus and bacteria. By uploading, their affected crops leave users to benefit by understanding the disease name and cause then our client can suggest fertilizer by analyzing the composition between modeled images and user images. Our application creates a good environment for fertilizer suggestions and two more important processes where the user can have weather and soil reports.

Based on the location given by the user, a weather report is generated and a soil report contains the suggested crops to grow by the user based on the soil samples given by the user. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. The problem of efficient disease protection is closely associated with the problems of sustainable agriculture. Inexperienced pesticide usage can cause the event of long-term resistance of the pathogens, severely reducing the power to fight back. Timely and accurate diagnosis of plant diseases is one among the pillars of precision agriculture. It is crucial to stop unnecessary waste of monetary and other resources, thus achieving healthier production during this changing environment, appropriate and timely disease identification including early prevention has never been more important. There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms, or the effect becomes noticeable too late to act, and in those situations, a classy analysis is obligatory. However, most diseases generate some quite manifestation within the visible spectrum, therefore the eye examination of a trained professional is that the prime technique adopted in practice for disease detection.

To achieve accurate disease diagnostics a plant pathologist should possess good observation skills in order that one can identify characteristic symptoms. Variations in symptoms indicated by diseased plants may lead to an unsuitable diagnosis since unprofessional gardeners and hobbyists could have more difficulties determining it than knowledgeable plant pathologist. an automatic system designed to assist identify plant diseases by the plant's appearance and visual symptoms might be of great help to amateurs within the gardening process and trained professionals as a confirmation system in disease diagnostics. One of the main parts of Neural Networks is Convolutional neural networks (CNN). CNNs use image recognition and classification in order to detect objects, recognize faces, etc.

They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output. CNNs are primarily used to classify images, cluster them by similarities, and then perform object recognition. Many algorithms using CNNs can identify Plant disease, faces, street signs, animals, etc. Plant diseases can cause great damage to agricultural crops by significantly reducing their production because they restrict the growth of crops and lead to poor quality of products. Due to its lower yield and fiber, biofuel crops as agriculture struggles to keep up with the world's rapidly increasing population. An existing method for the detection and identification of plant leaf diseases is observed with the naked eye . Additionally, plants must be monitored in a consistent manner to avoid the spread of disease. Diseases, pests, and other undesirable substances present in crops can cause a sharp decline in agricultural production . The impact of these dangerous factors on crops has a direct impact on the decline of the quality and quantity of crops. To combat, control, and mitigate the effects of biological organisms and diseases. However, most diseases generate some kind of manifestation in the visible spectrum, so the naked eye examination of a trained professional is the prime technique adopted in practice for plant disease detection.

1.1 CONVOLUTIONAL NEURAL NETWORK ALGORITHM

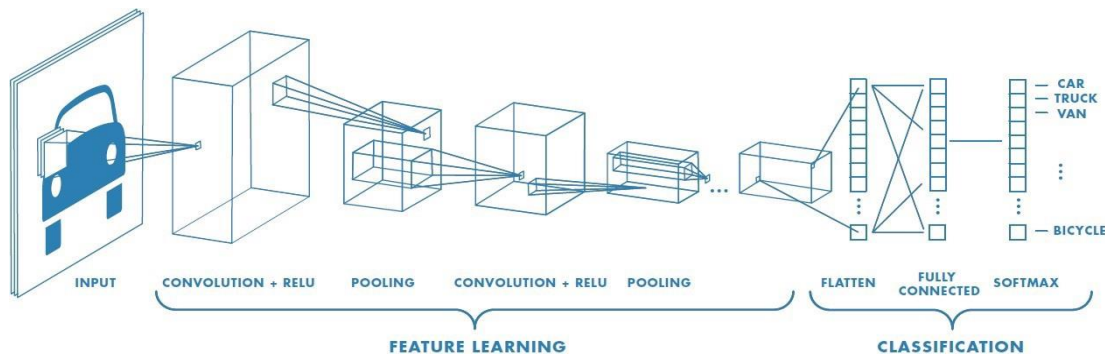


Fig 1.1: Feature Learning and Classification

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm-a Convolutional Neural Network.

They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks makes them prone to overfitting data.

1.2 CNN INTRODUCTION

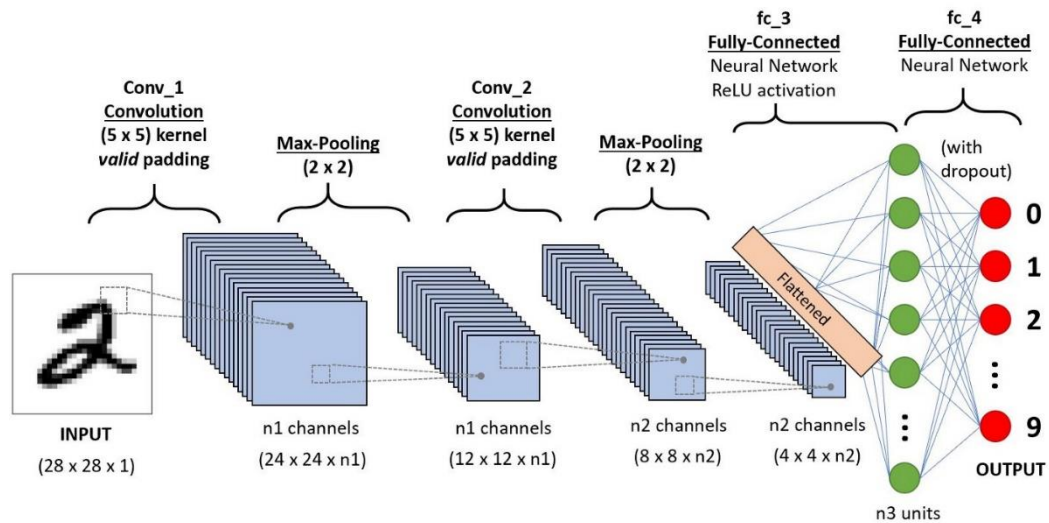


Fig 1.2: A CNN sequence to classify digits

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

It is a class of artificial neural network(ANN), most commonly applied to analyze visual imagery. They are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along features and provide translate-equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are only equivariant, as opposed to invariant, to translation.

They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interface, and financial time series. CNN are regularized versions of multiplayer perceptron. Multiplayer perceptron usually mean fully connected networks that is, each neuron in one layer is connected to all neurons on the next layer. The “full connectivity” of these networks makes them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms.

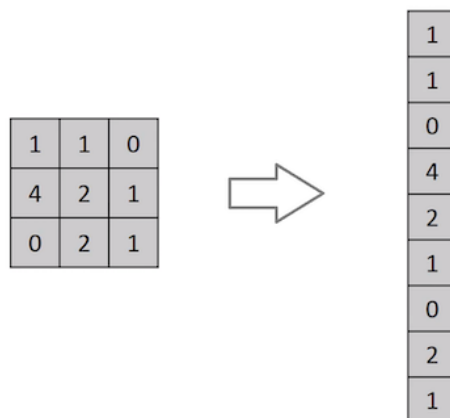


Fig 1.3: Flattering of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values. So flatten the image(e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification. In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout. A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

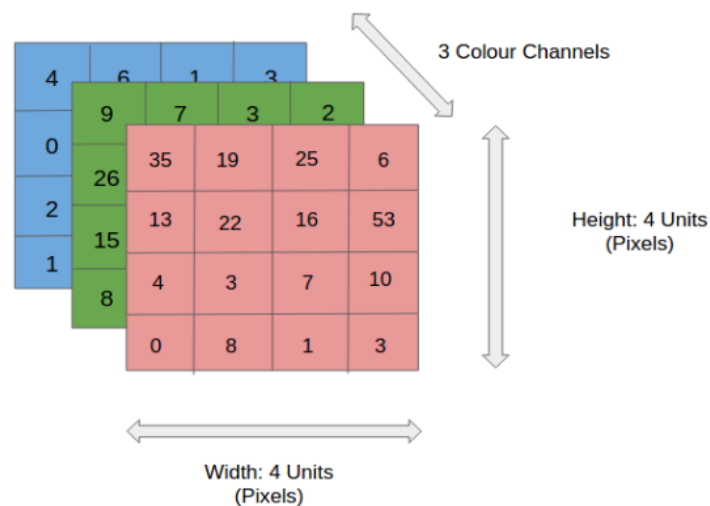


Fig 1.4: 4x4x3 RGB Image

In the above figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc. You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

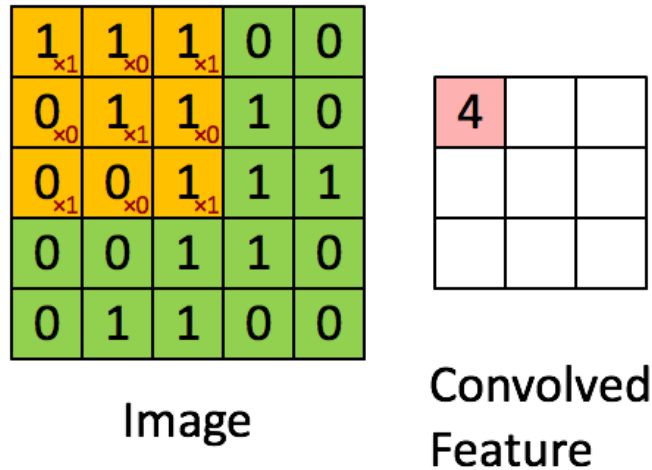


Fig 1.5: Convoluting a 5x5x1 image with a 3x3x1 kernel

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB). In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix**.

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.

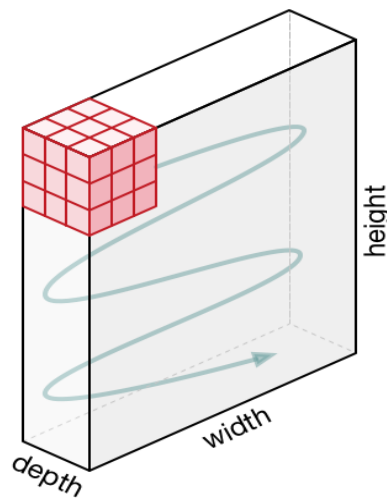


Fig 1.6: Movement of the kernel

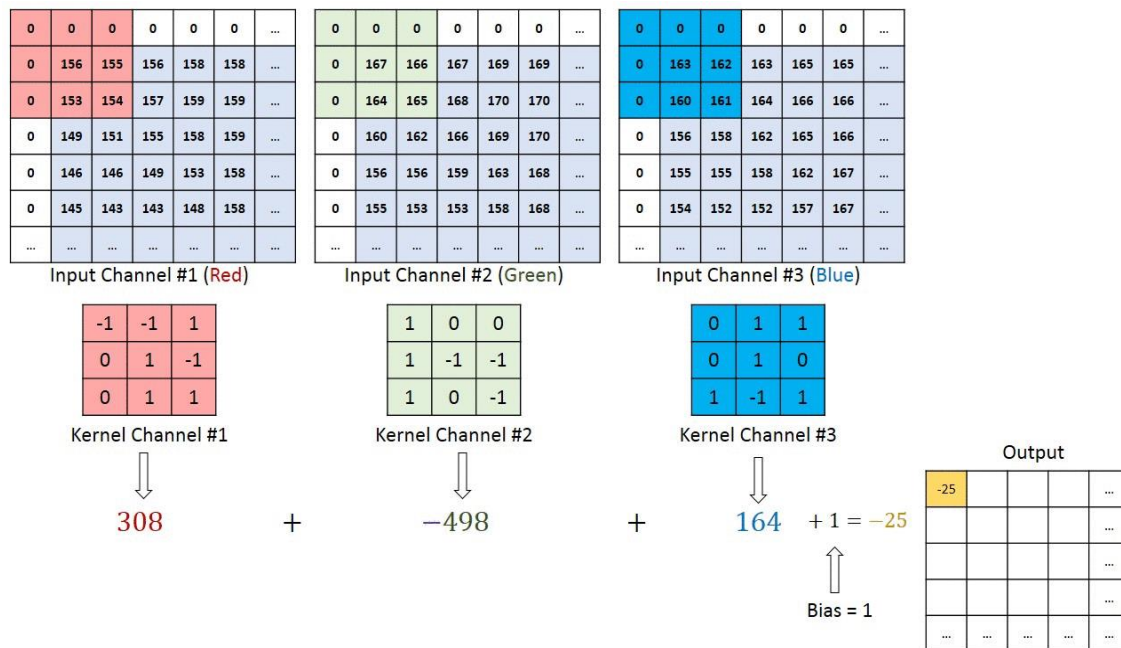


Fig 1.7: Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between K_n and I_n stack ($[K_1, I_1]; [K_2, I_2]; [K_3, I_3]$) and all the results are summed with the bias to give us a squashed one-depth channel Convolved Feature Output.

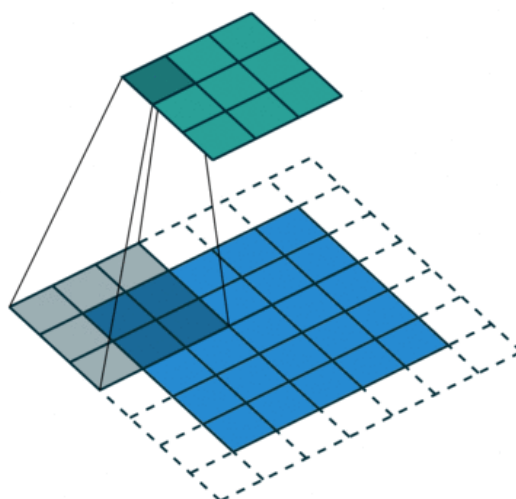


Fig 1.8: Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc.

With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would. There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

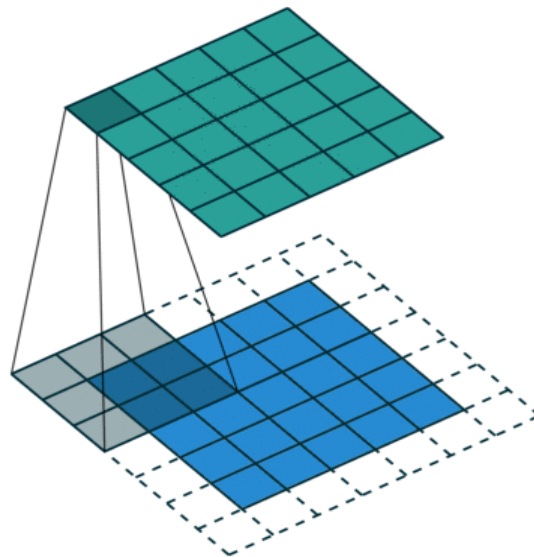


Fig 1.9: 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — **Same Padding**. On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — **Valid Padding**. The following repository houses many such GIFs which would help you get a better understanding of how Padding and Stride Length work together to achieve results relevant to our needs.

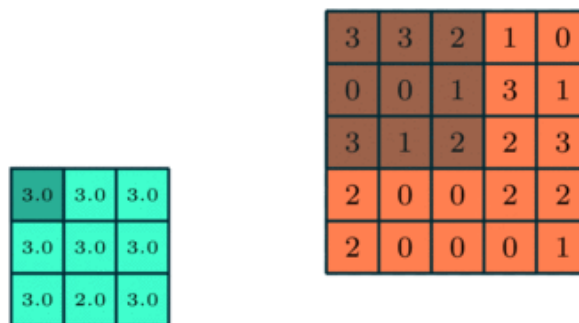


Fig 1.10: 3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

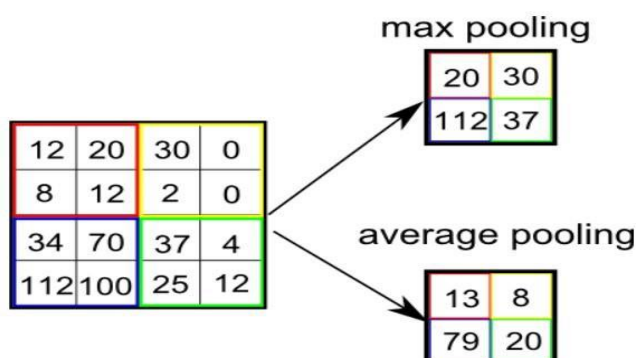


Fig 1.11 Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i -th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power. After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

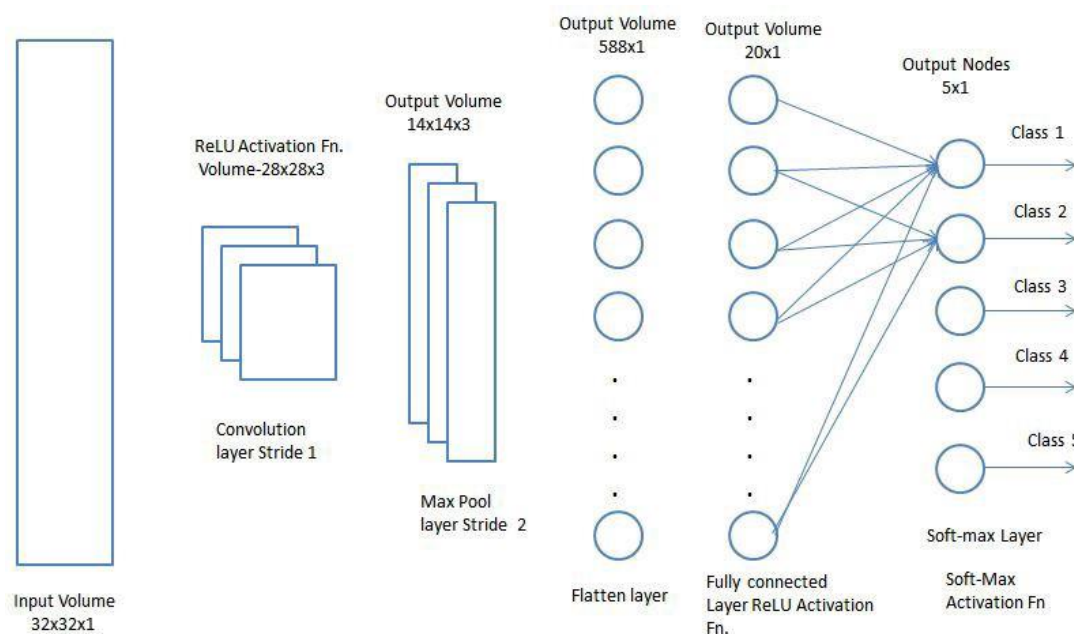


Fig 1.12: Classification-Fully Connected layer

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **SoftMax Classification** technique

1.3 SOFTWARE PROGRAMING TOOL

PyCharm is hybrid platform developed by JetBrains as an IDE for Python. It is commonly used Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE. We can run PyCharm on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers.

1.2.1 Features of PyCharm

➤ Intelligent Code Editor

- It helps us write high-quality codes!
- It consists of color schemes for keywords, classes, and functions. This helps increase the readability and understanding of the code.
- It helps identify errors easily.
- It provides the autocomplete feature and instructions for the completion of the code.

➤ Code Navigation

- It helps developers in editing and enhancing the code with less effort and time.
- With code navigation, a developer can easily navigate to a function, class, or file.
- A program can locate an element, a symbol, or a variable in the source code within no time.
- Using the lens mode, further, a developer can thoroughly inspect and debug the entire source code.

➤ **Refactoring**

- It has the advantage of making efficient and quick changes to both local and global variables.
- Refactoring in PyCharm enables developers to improve the internal structure without changing the external performance of the code.
- It also helps split up more extended classes and functions with the help of the extract method.

➤ **Assistance for Many Other Web Technologies**

- It helps developers create web applications in Python.
- It supports popular web technologies such as HTML, CSS, and JavaScript.
- Developers have the choice of live editing with this IDE. At the same time, they can preview the created/updated web page.
- The developers can follow the changes directly on web browser.
- PyCharm also supports AngularJS and NodeJS for developing web applications.

➤ **Support for Popular Python Web Frameworks**

- PyCharm supports web frameworks such as Django.
- It provides the autocomplete feature and suggestions for the parameters of Django.
- It helps in debugging the codes of Django.
- It also assists web2py and Pyramid, the other popular web frameworks.
- It consists of interactive graphs that help developers understand data.
- It is capable of integrating with various tools such as Python, Django, and Pytest. This integration helps innovate unique solutions.

1.4 PROGRAM STRUCTURE



Fig. 1.13 WAMPSEVER

WampServer is meant by constituting WAMP + SERVER, now here Wamp is a short form for “**Windows, Apache, MySQL and PHP**” whereas the Server simply means a computer program that provides services to other applications or clients.

Windows which is an ultimate platform for beginners and advanced users to operate, process and manage the different day to day computing tasks, however, if you are a developer and want to experience some of the most powerful software without paying a single penny then you should think about **Linux**. There are so many software packages that are only designed to run efficiently on Linux platforms such as Apache web server, PHP interpreter and MySQL database.

WAMP stands for Windows (operating system), Apache (webserver), MySQL (database), and PHP (coding language). WAMP can be compared to two other close variants, namely:

- LAMP (Linux, Apache, MySQL, and PHP) for Linux Systems.
- MAMP (Mac, Apache, MySQL, and PHP) for Mac OS operating systems.

WAMP, therefore, is a localhost server for Windows Operating Systems that lets you manage, create, test, and develop websites without having to use a remote web server.

Let's take *WordPress* as an example. If you want to build a website on the *WordPress* platform, for example, you can use the WAMP server to create a local environment that allows you to work offline.

Here you can experiment with code, plugins, design, as well as test the different features of your website before taking it live. It's important to note that Apache, PHP, and MySQL are all open-source software. This means that you can install and configure each service individually. However, this process is not easy, even for skilled developers. Luckily, a WAMP server automates the configuration process giving you a local working environment in a matter of minutes. With that in mind, now let's look at **how to install a WAMP server**. WampServer also supports phpMyAdmin, MariaDB, Adminer, PhpSysInfo. You don't have to enter complicated SQL statements to manage the MYSQL database, you can manage it directly from phpMyAdmin which provides a graphical user interface to create, manage and delete databases

➤ **Use of WampServer**

- In general, developers or users use the WampServer for testing various web applications or websites locally before making them live using WAMP. For example, you want to create a website on WordPress but before going live your website or purchasing hosting you can learn how to install WordPress, setting themes etc. locally using WampServer.
- Apache Web server or Apache HTTP server to allow users testing web pages or apps in Windows browser.
- PHP- Hypertext Pre-processor scripting language for web development and can be embedded into HTML.
- Developers or users use the WampServer for testing various web applications or websites locally before making them live using WAMP. For example, you want to create a website on WordPress but before going live your website or purchasing hosting you can learn how to use WampServer.

1.4.1 TENSORFLOW

It is an open-source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perceptron, Understanding, Discovering, Prediction and Creation. TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models.

You can use the TensorFlow library do to numerical computations, which in itself doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges. The name "TensorFlow" is derived from the operations which neural networks perform on multidimensional data arrays or tensors! It's literally a flow of tensors. For now, this is all you need to know about tensors, but you'll go deeper into this in the next sections.

➤ **Use Case of TensorFlow:**

- Voice/Sound Recognition.
- Texted Based Applications.
- Image Recognition.

○ **Voice/Sound Recognition.**

One of the most well-known uses of TensorFlow are Sound based applications. With the proper data feed, neural networks are capable of understanding audio signals. These can be:

- Voice recognition – mostly used in IoT, Automotive, Security and UX/UI
- Voice search – mostly used in Telecoms, Handset Manufacturers
- Sentiment Analysis – mostly used in CRM
- Flaw Detection (engine noise) – mostly used in Automotive and Aviation.

- **Texted Based Applications.**

Further popular uses of TensorFlow are, text-based applications such as sentimental analysis (CRM, social media), Threat Detection (Social Media, Government) and Fraud Detection (Insurance, Finance) Language Detection is one of the most popular uses of text based applications.

We all know Google Translate, which supports over 100 languages translating from one to another. The evolved versions can be used for many cases like translating jargon legalese in contracts into plain language. Text Summarization.

- **Image Recognition.**

Mostly used by Social Media, Telecom and Handset Manufacturers; Face Recognition, Image Search, Motion Detection, Machine Vision and Photo Clustering can be used also in Automotive, Aviation and Healthcare Industries. Image Recognition aims to recognize and identify people and objects in images as well as understanding the content and context. TensorFlow object recognition algorithms classify and identify arbitrary objects within larger images. This is usually used in engineering applications to identify shapes for modelling purposes (3D space construction from 2D images) and by social networks for photo tagging (Facebook's Deep Face).

By analyzing thousands of photos of trees for example, the technology can learn to identify a tree it has never seen before. Image recognition refers to the task of inputting an image into a neural network and having it output some kind of label for that image. The label that the network outputs will correspond to a pre-defined class. There can be multiple classes that the image can be labeled as, or just one. If there is a single class, the term "*recognition*" is often applied, whereas a multi-class recognition task is often called "*classification*". A subset of image classification is object detection, where specific instances of objects are identified as belonging to a certain class like animals, cars, or people. In order to carry out image recognition/classification, the neural network must carry out feature extraction. Features are the elements of the data that you care about which will be fed through the network. In the specific case of image recognition, the features are the groups of pixels, like edges and points, of an object that the network will analyze for pattern.

1.5 PLANT DISEASES

1.5.1 CEDER APPLE RUST (APPLE)



FIG 1.14 CEDER APPLE RUST

Cedar apple rust (*Gymnosporangium juniperi-virginianae*) is a fungal disease that requires juniper plants to complete its complicated two-year life-cycle. Spores overwinter as a reddish-brown gall on young twigs of various juniper species. In early spring, during wet weather, these galls swell and bright orange masses of spores are blown by the wind where they infect susceptible apple and crab-apple trees. The spores that develop on these trees will only infect junipers the following year.

From year to year, the disease must pass from junipers to apples to junipers again; it cannot spread between apple trees. On apple and crab-apple trees, look for pale yellow pinhead sized spots on the upper surface of the leaves shortly after bloom. These gradually enlarge to bright orange-yellow spots which make the disease easy to identify. Orange spots may develop on the fruit as well. Heavily infected leaves may drop prematurely. There are some ways by which we can provide treatment of Cedar apple rust:

- Choose resistant cultivars when available.
- Rake up and dispose of fallen leaves and other debris from under trees.
- Remove galls from infected junipers. In some cases, juniper plants should be removed entirely.
- Apply preventative, disease-fighting fungicides labeled for use on apples weekly, starting with bud break, to protect trees from spores being released by the juniper host. This occurs only once per year, so additional applications after this springtime spread are not necessary.
- On juniper, rust can be controlled by spraying plants with a copper solution (0.5 to 2.0 oz/ gallon of water) at least four times between late August and late October.
- Safely treat most fungal and bacterial diseases with SERENADE Garden. This broad-spectrum bio-fungicide uses a patented strain of *Bacillus subtilis* that is registered for organic use. Best of all, SERENADE is completely non-toxic to honey bees and beneficial insects.
- Containing sulfur and pyrethrins, Bonide® Orchard Spray is a safe, one-hit concentrate for insect attacks and fungal problems. For best results, apply as a protective spray (2.5 oz/ gallon) early in the season.
- Since the juniper galls are the source of the spores that infect the apple trees, cutting them is a sound strategy if there aren't too many of them. While the spores can travel for miles, most of the ones that could infect your tree **are within a few hundred feet**. The best way to do this is to prune the branches about 4-6 inches below the galls. You will want to disinfect your pruning shears, so you don't spread the infection. Dip them in 10% bleach or alcohol for at least 30 seconds between cuts.
- If your tree has a history of infection with cedar apple rust, you will want to get ahead of the infection and take preemptive measures. This is critical in the spring, when the juniper galls are releasing their spores. The time to treat your tree is between the pink stage of the blossoms (when the leaves are turning green) to the period when the petals drop.

1.5.2 GRAY LEAF SPOT (CORN)



FIG 1.15 GRAY LEAF SPOT

Gray leaf spot is the number one disease in all corn production. The fungus survives the winter on residue from the prior corn crop, providing a primary source of the inoculum, particularly when growers plant continuous corn or use a no-till soybean/corn rotation. Spores from the fungus are splashed onto the lower leaves early in the season. When these resulting infections produce spores, the inoculum can be carried by air currents higher into the canopy or into neighboring fields. Mature lesions from gray leaf spot are rectangular, as they are restricted by the leaf veins.

As every veteran corn producer knows, corn is susceptible to a myriad of foliar fungal diseases. The most aggressive of these diseases, gray leaf spot, has been increasing in economic impact in many regions of the world over the past 10 years. Because gray leaf spot reduces a plant's photosynthetic surface area, the fungus significantly diminishes both the yield and quality of the crop. Field trials have shown potential yield losses from gray leaf spot range from 5 to 40 bu/A; however, losses as high as 100 percent are documented when early blighting of the leaves above the ear leaf occurs.

As reduced tillage became widespread in the 1980s and '90s, gray leaf spot disease grew commonplace in the heart of the Corn Belt. Farmers eager to capitalize on the increasing value and demand for corn augmented the amount of Midwest acreage in continuous corn. Both trends have contributed to an increase in the amount of corn residue in fields—a circumstance that can lead to severe infections the following year. The planting of susceptible hybrids and weather conditions that favor rapid spread of the disease have contributed to the fact that gray leaf spot is now endemic in many regions of the world.

Future yield losses will correlate with environmental conditions and the overwintering of the fungus in crop residue. If the humidity and frequency of heavy rainfall forecasted for many regions of the U.S. materializes, pathogens are expected to become even more problematic. Gray leaf spot initially appears as spots on the leaves that are round or oval, tan in color, and have a dark brown border. When the leaves are wet or humidity is high, the leaf spots turn gray and fuzzy with profuse spore production. In time, the leaf spots expand and girdle the leaf, causing it to die back from the tip. Significant damage to the turf stand may occur as the disease continues to progress.

1.5.3 EARLY BLIGHT (POTATO)

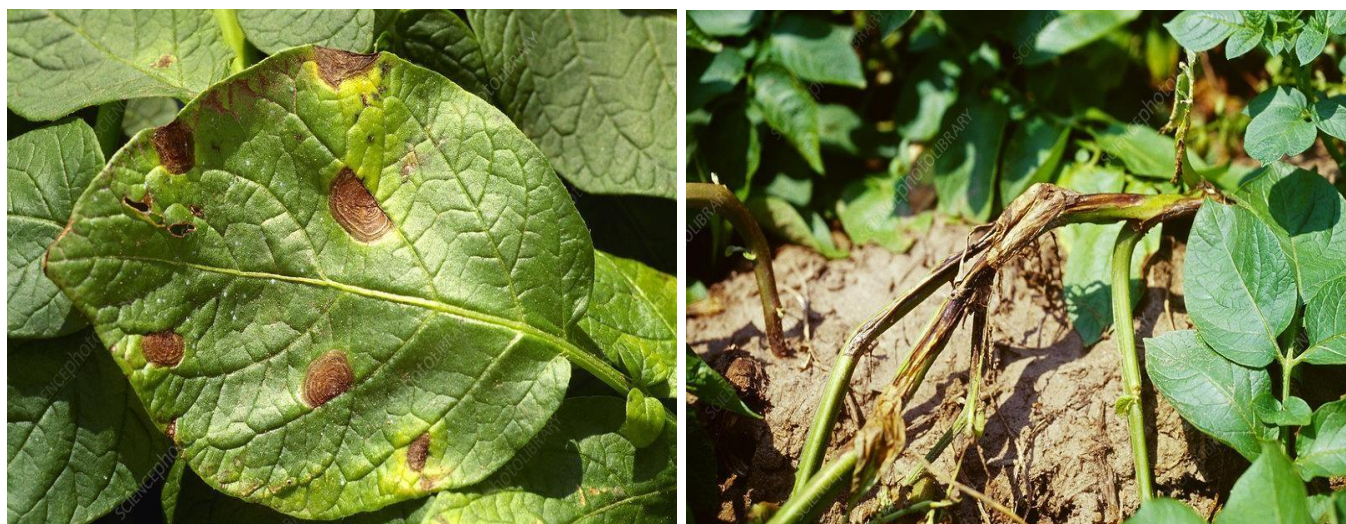


FIG 1.16 EARLY BLIGHT

Early blight of potato is caused by the fungal pathogen *Alternaria solani*. The disease affects leaves, stems and tubers and can reduce yield, tuber size, storability of tubers, quality of fresh-market and processing tubers and marketability of the crop.

In most production areas, early blight occurs annually to some degree. The severity of early blight is dependent upon the frequency of foliar wetness from rain, dew, or irrigation; the nutritional status of the foliage; and cultivar susceptibility. Across the Midwestern U.S., foliar infection is the most problematic phase of the disease, whereas in the western portion of the country, tuber infection can be more damaging. The disease first develops on mature and senescing foliage, and early maturing cultivars are the most susceptible. Potato is the primary host, but the disease also can be severe on tomatoes, and occur on other *solanaceous* plants such as hairy nightshade.

Symptoms of early blight occur on fruit, stem and foliage of tomatoes and stem, foliage and tubers of potatoes. Initial symptoms on leaves appear as small 1-2 mm black or brown lesions and under conducive environmental conditions the lesions will enlarge and are often surrounded by a yellow halo. Lesions greater than 10 mm in diameter often have dark pigmented concentric rings. This so-called “bullseye” type lesion is highly characteristic of early blight (Figure 4). As lesions expand and new lesions develop entire leaves may turn chlorotic and dehisce, leading to significant defoliation. Lesions occurring on stems are often sunken and lens-shaped with a light center, and have the typical concentric rings (Figure 5). On young tomato seedlings lesions may completely girdle the stem, a phase of the disease known as “collar rot,” which may lead to reduced plant vigor or death. Symptoms of early blight occur on fruit, stem and foliage of tomatoes and stem, foliage and tubers of potatoes. Initial symptoms on leaves appear as small 1-2 mm black or brown lesions and under conducive environmental conditions the lesions will enlarge and are often surrounded by a yellow halo (Figures 2 and 3). Lesions greater than 10 mm in diameter often have dark pigmented concentric rings. This so-called “bullseye” type lesion is highly characteristic of early blight.

1.5.4 TOMATO LATE BLIGHT

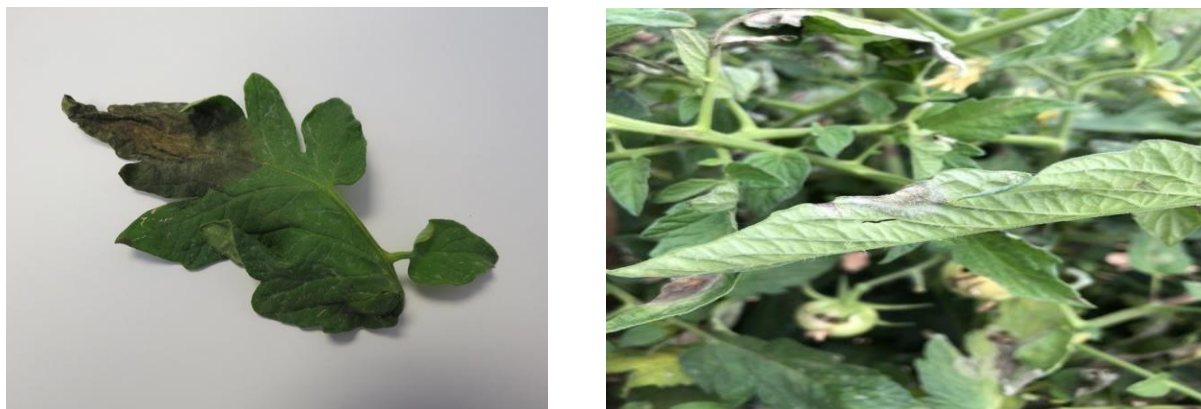


FIG 1.17 TOMATO LATE BLIGHT

Tomato late blight is caused by the oomycete pathogen *Pythophthora infestans*(*P. infestans*).The pathogen is best known for causing the devastating Irish potato famine of the 1840s,which killed over a million people, and caused another million to leave the country.

The first symptoms of late blight on tomato leaves are irregularly shaped, water-soaked lesions, often with a lighter halo or ring around them; these lesions are typically found on the younger, more succulent leaves in the top portion of the plant canopy. During high humidity, white cottony growth may be visible on underside of the leaf (Figure 2), where sporangia form .

Spots are visible on both sides of the leaves. As the disease progresses, lesions enlarge causing leaves to brown, shrivel and die. Late blight can also attack tomato fruit in all stages of development. Rotted fruit are typically firm with greasy spots that eventually become leathery and chocolate brown in color (Figure 4); these spots can enlarge to the point of encompassing the entire fruit. "Look alike" diseases: Other species of *Phytophthora*, specifically *P. nicotianae* and *P. capsici*, can also cause lesions on tomato and tomato fruit similar to those caused by late blight. However, these two species are soilborne (as opposed to ariborne) and typically only cause disease after the plants have been flooded or if the plant had been in contact with wet soil. Late blight can be differentiated from early blight on tomato and other foliar fungal diseases by where they occur on the plant.

Leaf symptoms of late blight first appear as small, water-soaked areas that rapidly enlarge to form purple-brown, oily-appearing blotches. On the lower side of leaves, rings of grayish white mycelium and spore-forming structures may appear around the blotches. Entire leaves die and infections quickly spread to petioles and young stems. Infected fruit turn brown but remain firm unless infected by secondary decay organisms; symptoms usually begin on the shoulders of the fruit because spores land on fruit from above.

1.6 MINERAL FERTILIZERS

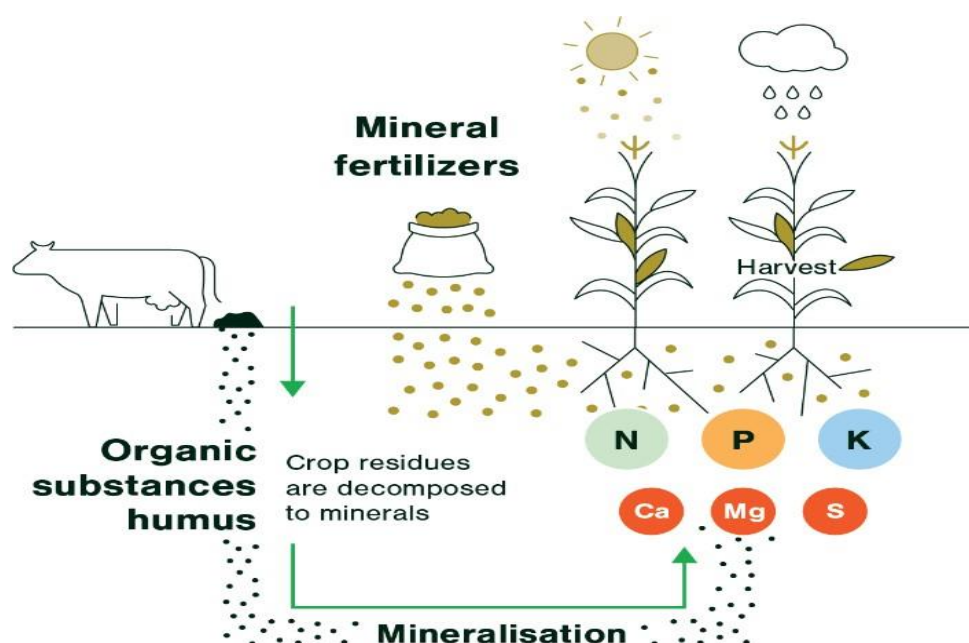


FIG 1.18 MINERAL FERTILIZERS

Mineral fertilizers are **materials, either natural or manufactured, containing nutrients essential for the normal growth and development of plants**. Plant nutrients are food for plants some of which are used directly for human food, others to feed animals, supply natural fibers or produce timber.

1.6.1 NITROGEN FERTILIZERS



FIG 1.19 NITROGEN FERTILIZERS

Nitrogen fertilizer means a substance containing nitrogen that is used for its plant nutrient content, is designed for use or claimed to have value in promoting plant growth, and requires a guaranteed analysis under Minnesota Statutes, section 18C.215. Nitrogen fertilizer does not include animal and vegetable manures that are not manipulated, or marl, lime, limestone, biosolids, industrial-by-product, industrial wastewater, irrigation water, or other products exempted by the commissioner.

It contains 24% Sulphur. Store well. It is applied at planting. Nitrogen will be released slowly. Takes about a month for all NH_4 to convert to NO_3 . The SO_4 radical combines with Ca and forms CaSO_4 radical combines with Ca and forms CaSO_4 which is soluble and hence Ca is removed & soil becomes acidic. It is Hygroscopic. So fertilizer bag should not be kept open for long time, it is most suitable for foliar spraying (1 to 5%). It undergoes hydrolysis, converted to Ammonium carbonate. Then to NH_4 and NO_3 . It is also used as food in small quantities for animals. In India it is manufactured to a larger extent. Highly hygroscopic, highly explosive thus all bags are tightly sealed until ready to use.

Keep all bags indoor in a dry place. Pile the stacks not higher than 6 bags. It is not manufactured in India. When it is allowed to be in stores it seeks into hard cake which requires to be broken up. It is more hygroscopic than Urea and ammonium nitrate. It is not suitable for paddy. It is not manufactured in India. Stable salt, stores well. Toxic to plants before it is converted into available form. Non leach able. It is a synthetic organic fertilizer.

1.6.2 PHOSPHORUS FERTILIZERS



FIG 1.20 PHOSPHORUS FERTILIZERS

Phosphorus is **an essential nutrient required for plant growth**. It helps in root development, plant maturation, and seed development. Along with nitrogen and potassium, phosphorus is one of the most important elements for plant life. The function of phosphorus in plants is very important.

It helps a plant convert other nutrients into usable building blocks with which to grow. Phosphorus is one of the main three nutrients most commonly found in fertilizers and is the “P” in the NPK balance that is listed on fertilizers. Phosphorus is essential to a plant’s growth, but what does it mean if you have high phosphorus in your soil, or a phosphorus deficiency? Keep reading to learn more about the importance of phosphorus in plant growth. The easiest way to tell is to look at the plants. If your plants are small, are producing little or no flowers, have weak root systems or a bright green or purplish cast, you have a phosphorus deficiency. Since most plants in the garden are grown for their flowers or fruit, replacing phosphorus in the soil if it is lacking is very important. Mineral phosphorus (P) fertilizers support high crop yields and contribute to feeding the teeming global population. However, complex edaphic processes cause P to be immobilized in soil, hampering its timely and sufficient availability for uptake by plants. The resultant low use efficiency of current water-soluble P fertilizers creates significant environmental and human health problems. Current practices to increase P use efficiency have been inadequate to curtail these problems.

1.6.3 POTASSIUM FERTILIZERS



FIG 1.21 POTASSIUM FERTILIZERS

Potassium is **the third key nutrient of commercial fertilizers**. It helps strengthen plants' abilities to resist disease and plays an important role in increasing crop yields and overall quality. Potassium fertilizers may also influence Cd availability to crops, through effects on plant growth, soil pH, ionic strength of the soil solution complexation, and/or ion competition (Grant et al., 1996). Under field conditions, KCl increased Cd concentrations in barley (Grant et al., 1996). The accompanying ion in the potassium fertilizer may have a strong effect on Cd availability. In pot studies with spring wheat, Cd accumulation and concentrations in plant shoots increased with increasing application rates of KCl and potassium sulfate (K_2SO_4) from 0 to 166 mg K kg^{-1} with no significant differences occurring between the two K sources (Zhao et al., 2004). However, shoot Cd concentrations were lower with KNO_3 than with KCl or K_2SO_4 and were only increased by the highest rate of KNO_3 application. In fields studies in Tasmania, Cd concentrations in potato tubers were higher with KCl application than with K_2SO_4 (Sparrow et al., 1994). As discussed in Section 3.1.2, Cl^- will increase the availability of Cd in soils through formation of chlorocomplexes (Smolders et al., 1998; Smolders and McLaughlin, 1996a,b). Therefore, application of Cl^- in KCl fertilizers may play an important role in its effect on Cd availability.

1.7 IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs offer intelligent code completion feature. Some IDEs contain a compiler, interpreter, or both, such as Net Beans and Eclipse, PyCharm; others do not, such as Sharp Develop and Lazarus.

The boundary between an integrated development environment and other parts of the broader *software development environment* is not well-defined. Sometimes a version control system and various tools are integrated to simplify the construction of a Graphical User Interface (GUI). Many modern IDEs also have a class browser, an object browser, and class diagram, for use in object-oriented software development. PyCharm is the most popular IDE used for Python scripting language. This chapter will give you an introduction to PyCharm and explains its features. PyCharm offers some of the best features to its users and developers in the following aspects – Code completion and inspection, Advanced debugging, Support for web programming and frameworks such as Django and Flask. When coding in Python, queries are normal for a developers. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one. An IDE typically contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interface (GUI). The user writes and edits source code in the code editor. The compiler translates the source code into a readable language that is executable for a computer. And the debugger tests the software to solve any issues or bugs. An IDE's toolbar looks much like a word processor's toolbar. The toolbar facilitates color-based organization, source-code formatting, error diagnostics and reporting, and intelligent code completion

CHAPTER 2

A Literature Survey: Plant Leaf Diseases Detection Using Convolutional Neural Network

Sourav Ray¹, Dr. B. Bharathi., ME., Phd²

Abstract: *This paper provides survey on different classification techniques that can be used for plant leaf diseases classification. Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. In our base paper image segregation and identification is done using matrix factorization and machine learning. But we create a trained model using a Convolutional Neural Network where the user-given image is detected by converting and comparing the images which are in matrix form. In our application, the user image is compared with our trained model to identify the specifics of images and sent to the client where machine learning helps to find the right fertilizers for the crop based on diseases recognized in the image given by the user in the first place. Our proposed method is way efficient and effective compared to traditional image recognition, and our application use weather report for any given location and soil report-suggestion for the cultivation of specific crop for best productivity.*

Keywords: *CNN, Machine Learning, Matrix Factorization.*

I. Introduction

Developing countries like India the economy is mainly depends on agriculture. Due to plant diseases the quality and quantity of agriculture product is reduced. Our application plays a vital role in recognizing the crop's image and finding out the diseases cause and fertilize suggestions after comparing with the trained model using Convolution Neural Network helps to differentiate each image by converting them by process of matrix convolutional. CNN represents an interesting method for adaptive image processing and forms a link between general feed-forward neural networks and adaptive filters. Any user who grows different crops leave user can benefit by understanding the disease name and cause then our client can suggest fertilizer by analyzing the composition between modelled images and user images. Our application creates a good environment for fertilizer suggestion and two more important processes where the user based on the soil samples given by the user. The pre-processing required in ConvNet is much lower as compared to other classification algorithms.

II. Literature Survey

In this section describes various approaches for detecting the disease in plant leaf using image processing technique

Ahmed, K., Shahidi, T.R & et al...[1] As one of the top ten rice producing and consuming countries in the world, Bangladesh depends greatly on rice for its economy and for meeting its food demands.

To ensure healthy and proper growth of the rice plants it is essential to detect any disease in time and prior to applying required treatment to the affected plants. Since manual detection of diseases costs a large amount of time and labour, it is inevitable prudent to have an automated system. This paper presents a rice leaf disease detection system using machine learning approaches. Three of the most common rice plant disease namely leaf smut, bacterial leaf blight and brown spot disease are detected in this work. Clear images of affected rice leaves with white background were used as the input. After necessary pre-processing, the dataset was trained on with a range of different machine learning algorithms including that o KNN(K- Nearest Neighbors), J48(Decision Tree), Naïve Bayes and Logistic Regression. Decision tree algorithm, after 10-fold cross validation, achieved an accuracy of over 97% when applied on the dataset.

Ashraf, T., Khan, Y.N & et al.... [2] Weeds are normally sprayed uniformly across field irrespective of their density in the field. Automatic methods can be developed that help us to locate and spray weed according to its density. Automatic weed detection is necessary for precision spraying that can improve rice yield and reduce production cost. This aids farmers to get an idea of local weed infested areas of field. This paper proposes two classification techniques to distinguish images based on their weed density. The weed covered by our work is of category grass and is called nutgrass in rice. We aim to classify images into three classes according to grass density. The first technique uses texture features extracted from gray level cooccurrence matrix (GLCM) and produces an accuracy of 73% using Radial Basis Function (RBF) kernel in Support Vector Machine (SVM). Another technique proposed uses features such as moments that are invariant to scale and rotation to classify grass density. The Second technique outperforms the first one with an accuracy of 86% using Random Forest classifier.

Sanga, S.L., D. Machuve., Jomanga, K., & et al... [3] In Tanzania, smallholder farmers contribute significantly to banana production and Kagera, Mbeya, and Arusha are among the leading regions. However, pests and diseases are a threat to food security. Early detection of banana diseases is important to identify the diseases before too much damage is done on the plants. In this paper, a tool for early detection of banana diseases by using a deep learning approach is proposed. Five deep learning architectures, namely Vgg16, Resnet18, Resnet50, Resnet152 and InceptionV3 were used to develop models for banana diseases detection, achieving all high accuracies, varying from 95.41% for InceptionV3 to 99.2% for Resnet152. InceptionV3 was selected for mobile deployment because it demands much less memory. The developed tool was capable of detecting diseases with a confidence of 99% of the captured leaves from the real environment. This tool will help smallholder farmers conduct early detection of banana diseases and improve their productivity.

V.K Tewari, C.M Pareek, Gurdeep Lal, L.K Dhruw, Naseeb Singh & et al... [4] The agrochemical application with conventional sprayers results in wastage of applied chemicals, which not only increases the economic losses but also pollutes the environment. In order to overcome these drawbacks, an image processing based real-time variable-rate chemical spraying system was developed for the precise application of agrochemicals in diseased paddy crop based on crop diseases severity information. The developed system comprised of web cameras for image acquisition, laptop for image processing, microcontroller for controlling the system functioning, and solenoid valve assisted spraying nozzles. The chromatic aberration(CA) based image segmentation method was used to detect the diseased severity level of paddy plants, based on which the solenoid valves remained on for a specific time duration so that the required amount of agrochemical could be sprayed on the diseased paddy plants. Field performance of developed sprayer prototype was evaluated in the variable-rate application(VRA) and constant-rate application (CRA) modes. The field testing results showed a minimum 33.88% reduction in applied chemical while operating in the VRA mode as compared with the CRA mode. Hence, the developed system appears promising and could be used extensively to reduce the cost of pest management as well as to control environment pollution due to such agrochemicals.

Prabira Kumar Sethy, Nalini Kanta Barpanda, Amiya Kumar Rath, Santi Kumari Behera & et al... [5] Features are the vital factor for image classification in the field of machine learning. The advancement of deep convolutional neural network (CNN) shows the way for identification of rice diseases using deep features with the expectation of high returns. This technology introduced 5932 on-field images of four types of rice leaf diseases, namely bacterial blight, blast, brown spot and tungro. In addition, the performance evaluation of 11 CNN models in transfer learning approach and deep feature plus support vector machine (SVM) was carried out. The simulation results show the deep feature plus SVM perform better classification compared to transfer learning counterpart. Also, the performance of small CNN models such as mobilenetv2 and shuffle net was examined. The performance evaluation was carried out in terms of accuracy, sensitivity, specificity, false positive rate (FPR), F1 Score and training time. Again, the statistical analysis was performed to choose the better classification model.

Yang Li, Jiachen Yang & et al... [6] Xinjiang is the major cotton-producing area in China, also well known in the world for its high-quality cotton. The growth and quality of cotton are closely related to the pest attack, but it is difficult for farmers to manually recognize all the types of pests because of the similar appearance. To solve this problem, we propose a few-shot cotton pest recognition method, which only needs few raw training data, quite different from the typical deep learning methods. We use two datasets to verify the effectiveness and feasibility of the few-shot model, one is the National Bureau of Agricultural Insect Resources (NBAIR), the other is a dataset with the natural scenes. The convolutional neural network (CNN) is adopted to extract feature vectors of images. The CNN feature extractor is trained by the triplet loss to distinguish different pest species to ensure the system robustness. Furthermore, the few-shot recognition model is finally running in an embedded terminal, based on the compiled convolutional and max-pooling circuit in the FPGA and the control program in the ARM. The running speed reaches 2 frames per second and can be faster by the further parallelism in hardware.

Sue Han Lee, Herve Goeau , Pierre Bonnet, Alexis Joly & et al... [7] The control of plant diseases is a major challenge to ensure global food security and sustainable agriculture. Several recent studies have proposed to improve existing procedures for early detection of plant diseases through modern automatic image recognition systems based on deep learning . In this article , we study these methods in details, especially those based on convolutional neural networks. We first examine whether it is more relevant to fine-tune a pre-trained model on a plant identification task rather than a general object recognition task. In particular, we show, through visualization techniques, that the characteristics learned differ according to the approach adopted and that they do not necessarily focus on the part affected by the diseases. Therefore, we introduce a more intuitive method that considers diseases independently of crops, and we show that it is more effective than the classic crop-disease pair approach, especially when dealing with diseases involving crops that are not illustrated in the training database. This finding therefore encourages future research to rethink the current de facto paradigm of crop disease categorization.

Krishnaswamy Rangarajan Arvind, Purushothaman Raja& et al... [8] The biotic stress of agricultural crops is a major concern across the globe. Especially, its major effects are felt in economically poor countries where advanced facilities for diagnosis of a disease is limited as well as lack of awareness among the farmers. A recent revolution in smartphone technology and deep learning techniques have created an opportunity for automated classification of disease. In this study images acquired through smartphone are transmitted to a personal computer via wireless Local Area Network (LAN) for classification of ten different diseases using transfer learning in four major agricultural crops which are least explored. Six pre-trained a Convolutional Neural Network(CNN) have been used namely Alex Net, Visual Geometry Group 16(VGG16), VGG19, Google Net , ResNet101 and DenseNet201 with its corresponding results explored. Google Net resulted in the best validation accuracy of 97.3%. The misclassification was mainly due to Tobacco Mosaic Virus(TMV) and two-spotted spider mite. In test conditions, images were classified in real-time and prediction scores have been evaluated for each disease class. It depicted a reduction in accuracy in all models with VGG16 resulting in the best accuracy of 90%.

Jatin Arora, Utkarsh Agrawal & et al... [9] Apart from being relied upon for feeding the entire world, the agricultural sector is also responsible for a third of the global Gross-Domestic-Product (GDP). Additionally, a majority opportunities for a significant fraction of the poor. This calls for methods to ensure the accurate and efficient diagnosis of plant disease, to minimize any adverse effects in the produce. This paper proposes the recognition and classification of maize plant leaf diseases by application of the Deep Forest algorithm. The Automated novel approach and accurate classification using the Deep Forest technique are a significant step-up from the existing manual classification and other technique with less accuracy. The proposed approach has outperformed Deep Neural models and other traditional machine learning algorithms in terms of accuracy. It justifies its low dependency on extensive Hyper-parameter tuning and the size of the dataset as against other Deep Learning Models based on neural networks.

David Opeoluwa Oyewola, Emmanuel Gbenga Dada, Sanjay Misra, Robertas Damasevicius&et al... [10] For people in developing countries cassava is a major source of calories and carbohydrates. However, Cassava Mosaic Disease (CMD) has become a major cause of concern among farmers in sub-Saharan Africa countries, which rely on cassava for both business and local consumption. The article proposes a novel deep residual convolution neural network (DRNN) for CMD detection in cassava leaf images. With the aid of distinct block processing, we can counterbalance the imbalanced image dataset of the cassava diseases and increase the number of images available for training and testing. Moreover, we adjust low contrast using Gamma correction and decorrelation stretching to enhance the color separation of an image with significant band-to-band correlation. Experimental results demonstrate that using a balanced dataset of images increases the accuracy of classification. The proposed DRNN model outperforms the plain convolutional neural network (PCNN) by a significant margin of 9.25% on the Cassava Disease Dataset from Kaggle. Cassava or manioc is a starchy root vegetable or tuber. People in developing countries depend largely on it for consumption due to a large supply of carbohydrates. It grows in the world's tropical regions because it can withstand harsh weather or unfavorable climatic conditions.

Weihui Zeng, Miao Li & et al... [12] The characteristics of the complex background in crop disease image, the small disease area, and the small contrast between the disease region and the background that easily causes confusion between them, seriously affect the recognition robustness and accuracy. To address these issues, we propose a Self-Attention Convolutional Neural Network (SACNN), which extracts effective features of crop disease spots to identify crop diseases. Our SACNN includes a basic network and a self-attention network: the basic network is for extracting the global features of the image, and the self-attention network: the basic network is for extracting the global features of the image, and the self-attention network is for obtaining the local features of the lesion area. Extensive experimental results show that the recognition accuracy of SACNN on AES-CD9214 and MK-D2 is 95.33% and 98.0%, respectively. The recognition accuracy of SACNN on MK-D2 has outperformed the state-of-the-art method by 2.9%, which implies that the CNN with self-attention can focus on the important areas of the image, and thus can improve the recognition accuracy. Adding different levels of noise to the AES-CD9214 test set shows the anti-interference ability and the strong robustness of SACNN. In addition, we discuss the influence of the location selection, channel size setting, network number and other aspects of the self-attention network on the recognition performance, in order to show the self-attention network working mechanism and provide inspiration for future research.

Yafeng Zhao, Zhen Chen, Xua Gao, Wenlong Song, Qiang Xiong, Junfeng Hu, Zhichao Zhang & et al..[13] Plant leaves can be used to effectively detect plant diseases. However, the number of images of unhealthy leaves collected from various plants is usually unbalanced. It is difficult to detect diseases using such an unbalanced dataset. We used Double GAN (a double generative adversarial network) to generate images of unhealthy plant leaves to balance such datasets. We proposed using Double GAN is divided into two stages. In stage 1, we used healthy leaves and unhealthy leaves as inputs. First, the healthy leaf images were used as inputs for the WGAN (Wasserstein generative adversarial network) to obtain the pretrained model. Then, unhealthy leaves were used for the pretrained model to generate 64*64 pixel images of unhealthy leaves.

Jaafar Abdulridha, Yiannis Ampatzidis, Sri Charan Kakarla, Pamela Roberts & et al... [14] Early and accurate diagnosis is a critical first step in mitigating losses caused by plant diseases. An incorrect diagnosis can lead to improper management decisions, such as selection of the wrong chemical application that could potentially result in further reduced crop health and yield. In tomato, initial diseases symptoms may be similar even if caused by different pathogens, for example early lesions of target spot (TS) caused by the fungus *Corynespora cassicola* and bacterial spot (BS) caused by *Xanthomonas perforans*. In this study, hyperspectral imaging (380-1020 nm) was utilized in laboratory and field (collected by an unnamed aerial vehicle; UAV) settings to detect both diseases. Tomato leaves were classified into four categories: healthy, asymptomatic, early and late diseases development stages. Thirty-five spectral vegetation indices (VIs) were calculated to select an optimum set of indices for diseases detection and identification.

Two classification methods were utilized: (i) multiplayer perceptron neural network (MLP), and (ii) stepwise discriminant analysis (STDA). Best wavebands selection was considered in blue (480-420 nm), red (630-650 nm) and red edge (730-750nm). The most significant VIs that could distinguish between healthy leaves and diseased leaves were photochemical reflectance index (PRI) for both diseases, the normalized difference vegetation index (NDVI850) for BS in all stages, and the triangular vegetation index (TVI), NDVI850 and chlorophyll index green (Chl green) for TS asymptomatic, TS early and TS late disease stage respectively. The MLP classification method had an accuracy of 99%, for both BS and TS, under field (UAV-based) and laboratory conditions. The MLP classification method had an accuracy of 99%, for both BS and TS, under field (UAV-based) and laboratory conditions. Fresh-market tomato plants face challenges from numerous diseases in Florida production. Foliar diseases on fresh-market tomato, specifically TS and BS, are intensively managed in Florida through cultural and chemical means (Sharma and Bhattarai 2019; Onofre et al. 2019).

Agarwal, M. & et al... [15] The tomato crop is an important staple in the Indian market with high commercial value and is produced in large quantities. Diseases are detrimental to the plant's health which in turn affects its growth. To ensure minimal losses to the cultivated crop, it is crucial to supervise its growth. There are numerous types of tomato diseases that target the crop's leaf at an alarming rate. This paper adopts a slight variation of the convolutional neural network model called LeNet to detect and identify diseases in tomato leaves. The main aim of the proposed work is to find a solution to the problem of tomato leaf disease detection using the simplest approach while making use of minimal computing resources to achieve results comparable to state of the art techniques. Neural network models employ automatic feature extraction to aid in the classification of the input image into respective disease classes. This proposed system has achieved an average accuracy of 94-95 % indicating the feasibility of the neural network approach even under unfavorable conditions.

AL-Sadi, A. & et al... [16] Plant diseases have always been a challenge to plant growth and crop production in several parts of the world. Plant diseases can affect plants by interfering with several processes such as the absorbance and translocation of water and nutrients, photosynthesis, flower and fruit development, plant growth and development and cell division and enlargement. Plant diseases can be caused by different types of fungi, bacteria, phytoplasma, viruses, viroid's, nematodes and other agents. The severity of diseases caused by these pathogens varies from mild symptoms to decline of the infected plants, depending on the aggressiveness of the pathogen, host resistance, environmental conditions, duration of infection and other factors. Plant disease symptoms vary with the infecting pathogen and the infected part and can include leaf spots, leaf blights, root rots, fruit rots, fruit spots, wilt, dieback and decline. Although infection of plants by pathogens can have serious consequences on plant health, human health can be affected by one of the several ways. Viruses, bacteria and fungi that infect plants do not usually cause infection in humans. However, a study reported that Pepper mild mottle virus may react with the immune system of humans and induce a clinical symptom, but the study did not provide a clear evidence on the pathogenic role of this plant virus in humans.

Ault, R. & et al... [17] Machine Learning is an important and growing field within Artificial Intelligence. It is particularly useful in situations where developing an algorithm to perform the task in a conventional way would be extremely difficult. Instead of being programmed specifically to complete a task, a program embodies a trained model that can recognize patterns present in given example data, and is able use that model to make predictions on future data. Neural networks are a prominent example of machine learning models used for this purpose. Neural networks are models that are based on how brains work, with massive numbers of connected processing elements called nodes. Training the model is a process of iteratively assigning different weights to node connections; in effect, the network “learns” to recognize patterns. This process of training a neural network to perform a task involves repeatedly testing it with example data and using the results to modify the parameters or weights within the model in such a way as to minimize error. The component that makes these changes is called the optimizer. As the field of machine learning has grown and matured, many different types of optimizers have been proposed and developed, each with their own set of advantages and disadvantages. The goal of this research project is to conduct a performance evaluation of different optimizers, quantifying the speed and accuracy with which they perform an image classification task.

Abbas, A.; Jain, S.; Gour, M.; Vankudothu, S. & et al... [18] Plant diseases and pernicious insects are a considerable threat in the agriculture sector. Therefore, early detection and diagnosis of these diseases are essential. The ongoing development of profound deep learning methods has greatly helped in the detection of plant diseases, granting a vigorous tool with exceptionally precise outcomes but the accuracy of deep learning models depends on the volume and the quality of labeled data for training. In this paper, we have proposed a deep learning-based method for tomato disease detection that utilizes the Conditional Generative Adversarial Network (C-GAN) to generate synthetic images of tomato plant leaves. Thereafter, a DenseNet121 model is trained on synthetic and real images using transfer learning to classify the tomato leaves images into ten categories of diseases. The proposed model has been trained and tested extensively on publicly available Plant Village dataset.

R. I. Hasan, S. M. Yusuf, and L. Alzubaidi & et al... [19] Deep learning (DL) represents the golden era in the machine learning (ML) domain, and it has gradually become the leading approach in many fields. It is currently playing a vital role in the early detection and classification of plant diseases. The use of ML techniques in this field is viewed as having brought considerable improvement in cultivation productivity sectors, particularly with the recent emergence of DL, which seems to have increased accuracy levels. Recently, many DL architectures have been implemented accompanying visualization techniques that are essential for determining symptoms and classifying plant diseases. This review investigates and analyses the most recent methods, developed over three years leading up to 2020, for training, augmentation, feature fusion and extraction, recognizing and counting crops, and detecting plant diseases, including how these methods can be harnessed to feed deep classifiers and their effects on classifier accuracy. Currently, there are widespread applications of deep learning (DL) around the world. These applications include health care, visual data processing, social network analysis, and audio and speech processing (e.g., recognition and enhancement). The efficacy of DL models such as reinforcement learning, long short-term memory and auto-encoders for solving dimensionality problems has been proven. In contrast with previous machine learning (ML) algorithms that employed several analytical measures for feature extraction, DL techniques learn features directly and represents them successively in the hierarchical architectures.

V. V. Adit, C. V. Rubesh, S. S. Bharathi, G. Santhiya, and R. Anuradha, & et al... [20] The Plant Disease is a major threat to Farmers, but faster identification remains a difficult task due to lack of incorporating latest technologies. The classification of plant disease is very challenging as it has only minimal features and hard to differentiate. The proposed technique uses a public dataset of 76,000 images of diseased plant leaves. We train the model using LeNet, AlexNet and Inception V3 for categorizing 38 varieties of plant diseases. The proposed model gives a comparison study of above three deep learning models and from the results we can predict Inception V3 performs better with an accuracy of 98% when compared to other models.

M. Loey, A. ElSawy, and M. Afify & et al... [21] Deep learning has brought a huge improvement in the area of machine learning in general and most particularly in computer vision. The advancements of deep learning have been applied to various domains leading to tremendous achievements in the areas of machine learning and computer vision. Only recent works have introduced applying deep learning to the field of using computers in agriculture. The need for food production and food plants is of utmost importance for human society to meet the growing demands of an increased population. Automatic plant disease detection using plant images was originally tackled using traditional machine learning and image processing approaches resulting in limited accuracy results and a limited scope. Using deep learning in plant disease detection made it possible to produce higher prediction accuracies as well as broadened the scope of detected diseases and plant species considered. This article presents a survey of research papers that presented the use of deep learning in plant disease detection, and analyzes them in terms of the dataset used, models employed, and overall performance achieved.

S. Hernandez and J. L. Lopez, & et al... [22] Climate change is having an enormous impact on crop production in Latin America and the Caribbean. This problem not only concerns the volume of crop production but also the quality and safety of the food industry. Several research studies have proposed deep learning for plant disease detection. However, there is little information about the confidence of the prediction on unseen samples. Therefore, uncertainty in models of plant disease detection is required for effective crop management. In particular, uncertainty arising from sample selection bias makes it difficult to scale automatic plant disease detection systems to production. In this paper, we develop a probabilistic programming approach for plant disease detection using state-of-the-art Bayesian deep learning techniques and the uncertainty as a misclassification measurement. The results show that Bayesian inference achieves classification performance that is comparable to the standard optimization procedures for fine-tuning deep learning models. At the same time, the proposed method approximates the posterior density for the plant disease detection problem and quantify the uncertainty of the predictions for out-of-sample instances.

M. H. Saleem, S. Khanchi, J. Potgieter, and K.M Arif & et al... [23] The identification of plant disease is an imperative part of crop monitoring systems. Computer vision and deep learning(DL) techniques have been proven to be state-of-the-art to address various agricultural problems. This research performed the complex tasks of localization and classification of the disease in plant leaves. In this regard, three DL meta-architecture including the single Shot MultiBox Detector (SSD), Faster Region-based Convolutional Neural Network (RCNN), and Region-based Fully Convolutional Networks (RCNN) were applied by using the TensorFlow object detection framework. All the DL models were trained/tested on a controlled environment dataset to recognize the disease in plant species. Moreover, an improvement in the mean average precision of the best-obtained deep learning architecture was attempted through different state-of-the-art deep learning optimizers. The SSD model trained with an Adam optimizer exhibited the highest mean average precision (mAP) of 73.07%. The successful identification of 26 different types of defected and 12 types of healthy leaves in a single framework proved the novelty of the work. In the future, the proposed detection methodology can also be adopted for other agricultural applications. Moreover, the generated weights can be reused for future real-time detection of plant disease in a controlled/uncontrolled environment.

U. Atila, M. Ucar, K. Akyol, and E.Ucar, & et al... [24] Most plant diseases show visible symptoms, and the technique which is accepted today is that an experienced plant pathologist diagnoses the disease through optical observation of infected plant leaves. The fact that the disease diagnosis process is slow to perform manually and another fact that the success of the diagnosis is proportional to the pathologist's capabilities makes this problem an excellent application area for computer-aided diagnostic systems. Instead of classical machine learning methods, in which manual feature extraction should be flawless to achieve successful result, there is a need for model that does not need pre-processing and can perform a successful classification. In this study, Efficient Net deep learning architecture was proposed in in plant leaf disease classification and the performance of this model was compared with other state-of-the-art-deep learning models.

J. G. M. Esgario, R. A. Krohling, and J.A. Ventura, & et al... [25] Biotic stress consists of damage to plants through other living organisms. The efficient control of biotic agents such as pests and pathogens (viruses, fungi, bacteria, etc.) is closely related to the concept of agricultural sustainability. Agricultural sustainability promotes the development of new technologies that allow the reduction of environment impacts, greater accessibility to farmers and , consequently, increased productivity. The use of computer vision with deep learning methods allows the early and correct identification of the stress-causing agent. So, corrective measures can be applied as soon as possible to mitigate the problem. The objective of this work is to design an effective and practical system capable of identifying and estimating the stress severity caused by biotic agents on coffee leaves. The proposed approach consists of a multi-task system based on convolutional neural networks. In addition, we have explored the use of data augmentation techniques to make the system more robust and accurate. Computational experiments performed with the proposed system using the ResNet50 architecture obtained an accuracy of for the biotic stress classification and for severity estimation. Moreover, it was found that by classifying only the symptoms, the results were greater than . The experimental results indicate that the proposed system might be a suitable tool to assist both experts and farmers in the identification and quantification of biotic stresses in coffee plantations.

R. Chalapathy and S.Chawla, & et al... [26] Anomaly detection is an important problem that has been well-studied within diverse research areas and application domains. The aim of this survey is two-fold, firstly we present a structured and comprehensive overview of research methods in deep learning-based anomaly detection. Furthermore, we review the adoption of these methods for anomaly across various application domains and assess their effectiveness. We have grouped state-of-the-art research technique's into different categories based on the underlying assumptions and approach adopted. Within each category we outline the basic anomaly detection technique, along with its various and present key assumption's, to differentiate between normal and anomalous behaviors.

Y. Guo, J. Zang, C. Yin & et al... [27] The identification of plant disease is the premise of the prevention of plant disease efficiently and precisely in the complex environment. With the rapid development of the smart farming, the identification of plant disease becomes digitalized and data-driven, enabling advanced decision support , smart analysis, and planning. This paper proposes a mathematical model of plant disease detection and recognition based on deep learning, which improves accuracy, generality, and training efficiency. Firstly, the region proposal network (RPN) is utilized to recognize and localize the leaves in complex surroundings. Then, images segmented based on the results of RPN algorithm contain the feature of symptoms through Chan–Vese (CV) algorithm. Finally, the segmented leaves are input into the transfer learning model and trained by the dataset of diseased leaves under simple background. Furthermore, the model is examined with black rot, bacterial plaque, and rust diseases. The results show that the accuracy of the method is 83.57%, which is better than the traditional method, thus reducing the influence of disease on agricultural production and being favorable to sustainable development of agriculture. Therefore, the deep learning algorithm proposed in the paper is of great significance in intelligent agriculture, ecological protection, and agricultural production.

J. S. H. Al-bayati and B. B. Ustundag & et al... [28] Apple leaf disease is the foremost factor that restricts apple yield and quality. Usually, much time is taken for disease detection with the existing diagnostic techniques; therefore, farmers frequently miss the best time for preventing and treating diseases. The detection of apple leaf diseases is a significant research problem, and its main aim is to discover an efficient technique for disease leaf image diagnosis. This article has made an effort to propose a method that can detect the disease of apple plant leaf using deep neural network (DNN). Plant diseases detection system (PDDS) architecture is designed. Speeded up robust feature (SURF) is used for feature extraction and Grasshopper Optimization Algorithm (GOA) for feature optimization, which helps to achieve better detection and classification accuracy. Classification parameters, such as Precision, Recall, F-measure, Error, and Accuracy is computed, and a comparative analysis has been performed to depict the effectiveness of the proposed work.

CHAPTER 3

PROJECT OVERVIEW

3.6 OBJECTIVES

These applications play a vital role in recognition the crop's leaves image and finding out the disease cause and fertilize suggestions after comparing with the trained model using Convolutional Neural Network helps to differentiate each image by converting them by process of matrix convolution. CNN represents an interesting method for adaptive image processing and forms a link between general feed-forward an interesting method for adaptive image processing and forms a link between general feed-forward networks and adaptive filters. Any user who grows different crops may not know all the diseases caused by fungus and bacteria. By uploading, their affected crops leave users to benefit by understanding the disease name and cause then our client can suggest fertilizer by analyzing the composition between modeled images and user images. application creates a good environment for fertilizer suggestions and two more important processes where the user can have weather and soil reports. Based on the location given by the user, a weather report is generated and a soil report contains the suggested crops to grow by the user based on the soil samples given by the user. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. Plant is exposed to many attacks from various micro-organism, bacterial disease and pests. The symptoms of the attacks are usually distinguished through the leaves, stem or fruit inspection. Disease that are commonly attack plants are Powdery Mildew and Leaf Blight and it may cause severe damaged if not controlled in early stages. The method is going to used in this project has widely being used for identification, detection, grading and quality inspection in the agriculture field. Detection and identification disease of a plant is very important especially, n producing an affected leaf.

3.7 MOTIVATION

Plant diseases are a very devastating threat to the agriculture industry and have the potential of pushing the whole of human society into starvation if not detected early. With the implementation of Convolutional Neural Network models, the detection of plant diseases will become easier and cheaper helping many farmers in the timely detection of plant diseases, preventing wastage of plants and protecting the transmission of diseases from diseased to healthy plants. A lot of the research that has been carried out on plant disease present a comparative study using different CNN models but fail to explain the predictions made by their models.

In this project we not only provide a comparative study between the workings of a simple and complex model, but we also aim at providing explainability for the predictions made by the models. The motivation for including explainability lies in the fact that most of the CNN models widely used in this project for detecting the diseases of the leaf, which leads to the users using this model for predicting in not trusting and understanding how this CNN models works and make their predictions. The Artificial Intelligence techniques allows the users to understand their model predictions better and make decisions on their own as to whether, trust their model or not. The use of

- TensorFlow
- BS4
- Pillow
- Django
- PIP

this server helps a great deal in the applications of plants disease detection as the transparency and explainability of the models being is vital in gaining the trust of the workers working in the agriculture industry as their livelihood is dependent on their production of healthy plants.

3.7.1 REQUIREMENTS

➤ SOFTWARE REQUIREMENTS:

- PYCHARM IDE
- PYTHON
- MYSQL DATABASE
- SERVER: TENSORFLOW, MYSQL CLIENT, BS4, PILLOW, DJANGO, PIP.

In this project we are going to make a **CONSOLIDATED SYSTEM Application Using Convolutional Neural** Network for finding the diseases of the plants and the perfect place of soil where plants can grow without any diseases

3.8 EXISTING SYSTEM

In general, critical factors for image analysis and pattern recognition, there are many models to initialize the process. To learn a robust distance metric, we need abundant side information which is usually unavailable in practice due to the high labelling cost.

A process like a matrix factorization, distance metric learning, and Clustering method finds similar pixels to classify into clusters or classes, yet these distance metric learning problems in unsupervised or weakly supervised clustering tasks, especially for the unified theoretic schemes and optimization algorithms. There is no proper image recognition using a huge number of images trained into a model which can be achieved by a Convolution neural network. Basically, in existing system there is no such feature by which we can detect the affected leaf, for which there is huge cause of damage of different types of crops and plants. In recent times due to bad weather and less information about soil farmers facing huge loss. In crops different types of substances are present like diseases, pests, etc., The impact on these dangerous factors on crops has a appearance, morphology, and other characteristics of the leaves. By using the existing system it is impossible to find a affected leaf and cure for it.

3.9 PROPOSED SYSTEM

In the proposed system, any farmers who want to find the cause and disease formed in their crops are benefited. When the farmer uploads images of the affected crop by fungus and bacteria, the user cannot conclude what caused it and what fertilizer should be used to mitigate the problem in near future. Admin has a trained model which contains a huge amount of images by matrix conversion using CNN then admin compares the user given images with the trained model. So, initial images are converted into an input layer which is a grayscale image.

The output layer is a binary or multi-class label. Hidden layers consist of convolution layers, ReLU (rectified linear unit) layers, the pooling layers, and a fully connected Neural Network. The above process takes care of image conversion 4*4 matrix to segregate each block according to the density of grayscale, compare it with the same kind of format and recognize disease using the trained model. Then the client suggests fertilizer required for the crop which is beneficial for any farmers. Our application also suggests which crop to grow according to soil input given by farmers, then the user can have weather reports according to the location they are given. These are achieved by web scraping and BeautifulSoup.

➤ FEATURES

- Disease Detection.
- Image Recognition using CNN.
- ReLU (Rectifier Linear Unit).
- Fertilizer Suggestion.
- Soil suggestion for crop to grow.
- Weather report according to the location provided by the user.
- Web Scraping.
- BeautifulSoup Python package.

3.10 TYPES OF DISEASE LEAF USED



Apple Leaf



Corn Leaf



Potato Leaf



Tomato Leaf

Fig. 3.1 4 Types of leaf diseases

CHAPTER 4

METHODOLOGY

4.1 OVERALL ARCHITECTURES

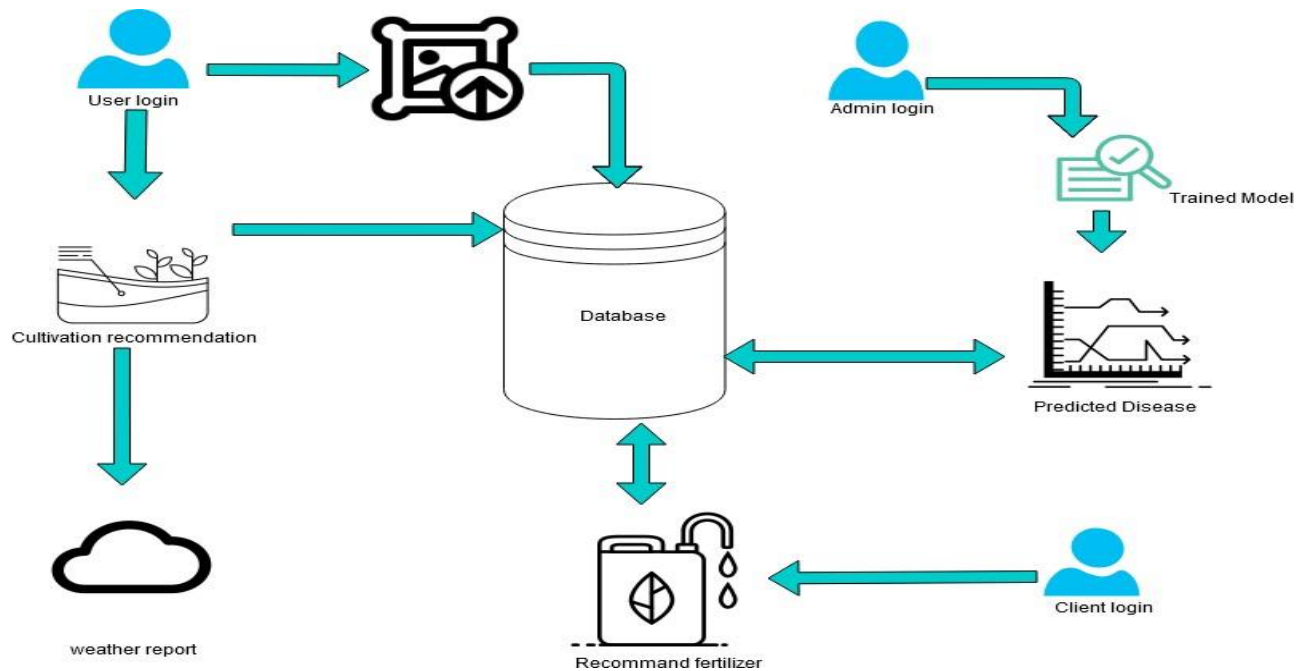


Fig. 4.0 ARCHITECTURE DIAGRAM

This diagram is the architecture of this project. Mainly user should login to upload the affected image of the leaf which is already uploaded in the MySQL Database. Then admin should login to predict disease of the image which will work through trained model and this trained model built by convolutional neural network algorithm which is basically a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. After detecting the diseases of the leaf user should check with recommendation of cultivation and weather report which is most important for finding the place where crops can grow. Then for better fertilizer client should login and search for required fertilizer for crops and plants leaf.

4.1.1 PLANT DEPLOYMENT

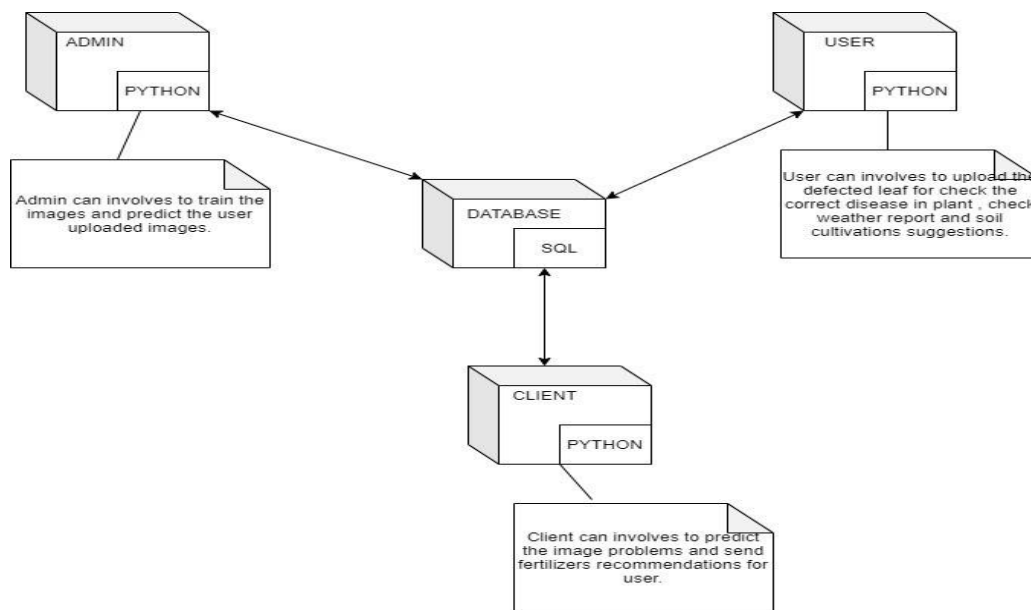
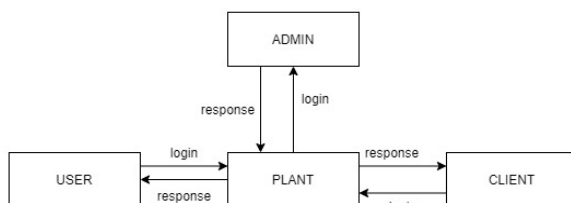


Fig. 4.1 PLANT DEPLOYMENT

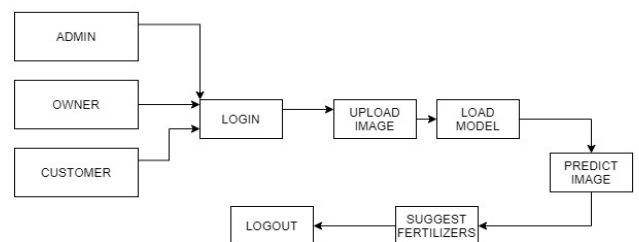
In this plant collaboration diagram where admin involves to train the images and predict what defected leaf image by the user which will be stored in MySQL database, in other hand user can also upload the defected leaf for checking the correct diseases present in the plant and for finding the better place to grow where there will be no defected leaf user can check weather report and soil cultivations suggestions. User request accomplished by client who can check what's the problem in the image and which is most import for agriculture is fertilizers, client can help user with fertilizer recommendations.

4.1.2 DATA FLOW DIAGRAM (DFD)

LEVEL 1



LEVEL 2



LEVEL 3



Fig. 4.2 DATA FLOW DIAGRAM

In this DFD diagram there are three levels in level 1 all three Admin, Client, User their login and response about the plant diseases which is very important to work in the next level in this diagram, in level 2 owner and customer login in the application through admin and they upload the defected images for predict the diseases in the leaf. In level 3 User, Client, Admin doing their final work to get final output, but in admin section we need user approval to make Train Neural Network to work and predict the defected leaf. Data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems. Data flow diagrams were popularized in the late 1970s, arising from the book *Structured Design*, by computing pioneers Ed Yourdon and Larry Constantine. They based it on the “data flow graph” computation models by David Martin and Gerald Estrin.

The structured design concept took off in the software engineering field, and the DFD method took off with it. It became more popular in business circles, as it was applied to business analysis, than in academic circles. Object Oriented Analysis and Design (OOAD), put forth by Yourdon and Peter Coad to analyze and design an application or system. Structured Systems Analysis and Design Method (SSADM), a waterfall method to analyze and design information systems. This rigorous documentation approach contrasts with modern agile approaches such as Scrum and Dynamic Systems Development Method (DSDM.) Three other experts contributing to this rise in DFD methodology were Tom DeMarco, Chris Gane and Trish Sarson. They teamed up in different combinations to be the main definers of the symbols and notations used for a data flow diagram. A Logical DFD visualizes the data flow that is essential for a business to operate. It focuses on the business and the information needed, not on how the system works or is proposed to work. However, a Physical DFD shows how the system is actually implemented now, or how it will be. For example, in a Logical DFD, the processes would be business activities, while in a Physical DFD, the processes would be programs and manual procedures. DFDs can be used to model a better, more efficient flow of data through a business process. BPR was pioneered in the 1990s to help organizations cut operational costs, improve customer service and better compete in the market. DFDs can be used to visualize and understand business and technical requirements and plan the next steps.

4.1.3 PLANT ENTITY RELATIONSHIP DIAGRAM

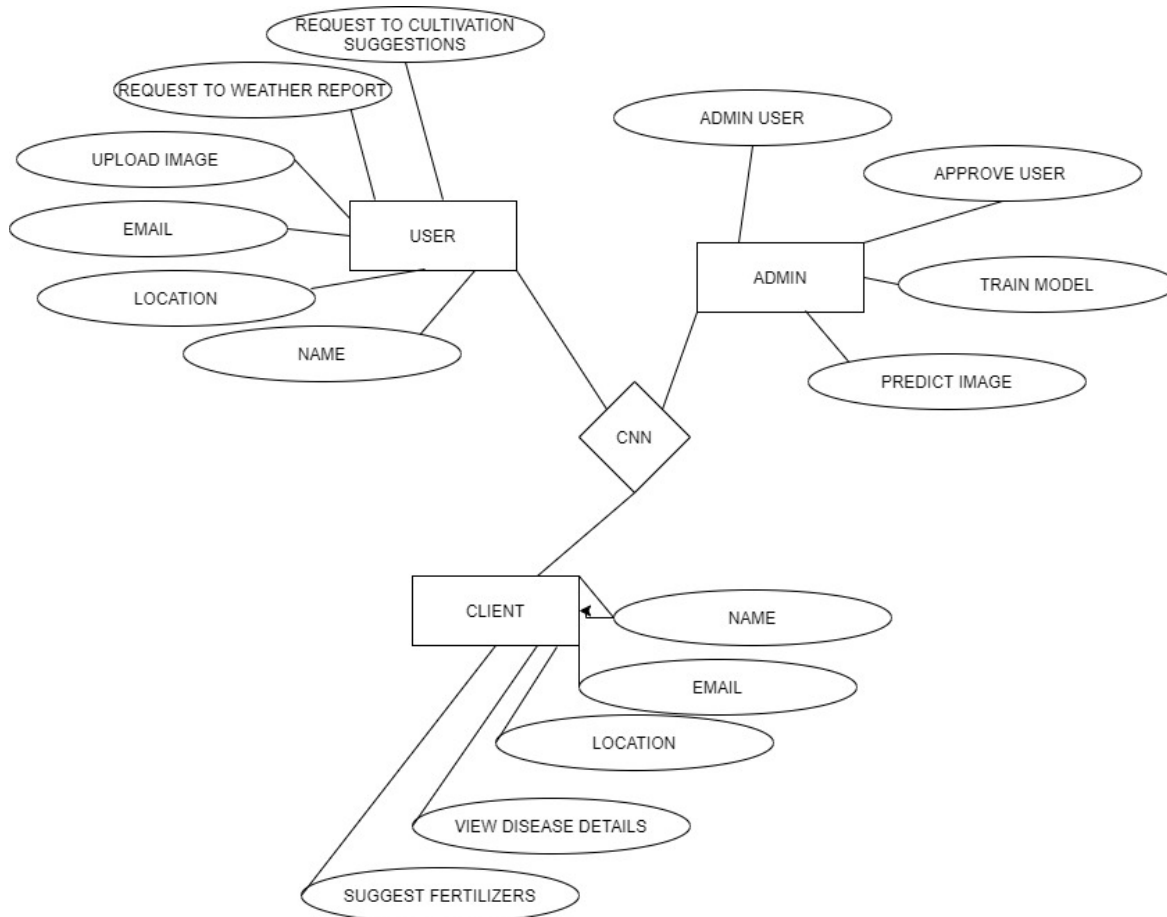


Fig. 4.3 ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems. ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.) In software engineering, an ER diagram is often an initial step in determining requirements for an information systems project. It's also later used to model a particular database or databases. A relational database has an equivalent relational table and can potentially be expressed that way as needed. ER diagrams are used to analyze existing databases to find and resolve problems in logic or deployment. Drawing the diagram should reveal where it's going wrong. The diagrams are used to design or analyze relational databases used in business processes. Any business process that uses fielded data involving entities, actions and interplay can potentially benefit from a relational database.

It can streamline processes, uncover information more easily and improve results. ER diagrams help in analysing databases used in business process re-engineering and in modelling a new database setup. Databases are today's method of storing relational information for educational purposes and later retrieval, so ER Diagrams can be valuable in planning those data structures. Since so much research focuses on structured data, ER diagrams can play a key role in setting up useful databases to analyze the data. How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines. Defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are one-to-one, one-to-many, and many-many.

4.1.4 PLANT USE CASE DIAGRAM

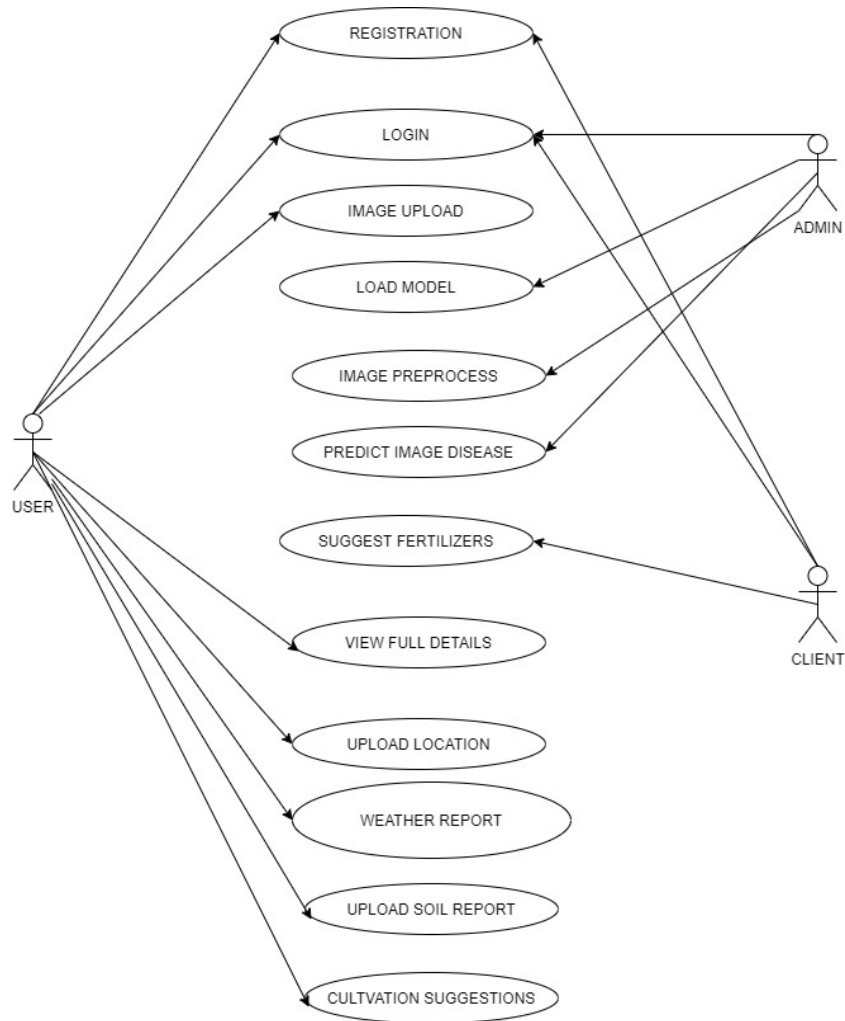


Fig. 4.4 USE CASE DIAGRAM

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow, however, use case diagram does not describe how those events are implemented. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site.

Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. There are a number of benefits with having a use case diagram over similar diagrams such as flowcharts. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components. The boundary, which defines the system of interest in relation to the world around it. The actors, usually individuals involved with the system defined according to their roles. The use cases, which are the specific roles played by the actors within and around the system. The relationships between and among the actors and the use cases.

A use case diagram looks something like a flowchart, where intuitive symbols represent the system elements. Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modeling the dependency between the two use cases. The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system. When a use case is depicted as using the functionality of another use case, the relationship between the use cases is named as include or uses relationship. A use case includes the functionality described in another use case as a part of its business process flow. A uses relationship from base use case to child use case indicates that an instance of the base use case will include the behaviour as specified in the child use case. An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at the base of the arrow. Use case granularity refers to the way in which information is organized within use case specifications, and to some extent, the level of detail at which they are written. Achieving the right level of use case granularity eases communication between stakeholders and developers and improves project planning

4.1.5 WORKING

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. In this project image segregation and identification is done using matrix factorization and machine learning. But we create a trained model using a Convolutional Neural Network where the user-given image is detected by converting and comparing the images which are in matrix form. In our application, the user image is compared with our trained model to identify the specifics of images and sent to the client where machine learning helps to find the right fertilizers for the crop based on disease recognized in the image given by the user in the first place.

Our proposed method is way efficient and effective compared to traditional image recognition. And our application use weather report for any given location and soil report - suggestion for the cultivation of specific crop for best productivity. Our application plays a vital role in recognizing the crop's leaves image and finding out the disease cause and fertilize suggestions after comparing with the trained model using Convolution Neural Network helps to differentiate each image by converting them by process of matrix convolution. CNN represents an interesting method for adaptive image processing and forms a link between general feed-forward neural networks and adaptive filters. Any user who grows different crops may not know all the diseases caused by fungus and bacteria. By uploading, their affected crops leave users to benefit by understanding the disease name and cause then our client can suggest fertilizer by analyzing the composition between modeled images and user images. Our application creates a good environment for fertilizer suggestions and two more important processes where the user can have weather and soil reports. Based on the location given by the user, a weather report is generated and a soil report contains the suggested crops to grow by the user based on the soil samples given by the user.

4.2 LANGUAGE DESCRIPTION

4.2.1 PYTHON PROGRAMMING

Python is an interpreted language and not a compiled one, although compilation is a step. Python code, written in **.py** file is first compiled to what is called byte code which is stored with a **.pyc or .pyo** format. Instead of translating source code to machine code like C++, Python code is translated to byte code. This byte code is a low-level set of instructions that can be executed by an interpreter. One popular advantage of interpreted languages is that they are platform-independent.

As long as the Python byte code and the Virtual Machine have the same version, Python byte code can be executed on any platform (Windows, macOS, etc). Dynamic typing is another advantage. In static-typed languages like C++, you have to declare the variable type and any discrepancy like adding a string and an integer is checked during compile time. In older programming languages, memory allocation was quite manual. Many times when you use variables that are no longer in use or referenced anywhere else in the program, they need to be cleaned from the memory. Garbage Collector does that for you.

➤ HOW PYTHON INTERPRETER WORKS

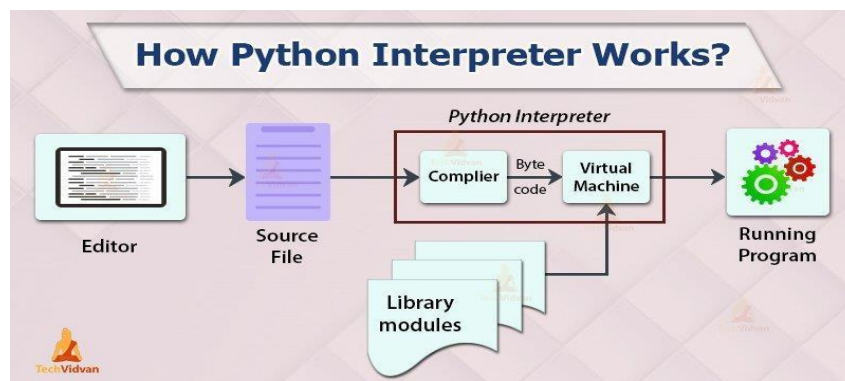


Fig 4.5 HOW PYTHON INTERPRETER WORKS

Python interpreter is **responsible for executing the Python scripts**. Python interpreter is a piece of software which works between the Python program and computer hardware. Here we are describing the series of ways to run Python scripts. The operating system command-line or terminal.

We write the python code in any text editor and save the same file using the “.py” extension in our system. Now, how will this code run? There must be some application or program like “python” or “python3” that must be installed in your system, and this is their duty to run this python code. This type of program is called Interpreter. Python does provide a python shell, which is used to execute a single command in python and display the result accordingly. The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line.

4.2.2 GUIDELINES FOR PYTHON CODE

One of Guido’s key insights is that code is read much more often than it is written. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code. As Python Enhancement Proposal says, “Readability counts”. A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important. However, know when to be inconsistent – sometimes style guide recommendations just aren’t applicable. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don’t hesitate to ask! In particular: do not break backwards compatibility just to comply with this PEP! For example, do not rely on CPython’s efficient implementation of in-place string concatenation for statements in the form `a += b` or `a = a + b`. This optimization is fragile even in CPython (it only works for some types) and isn’t present at all in implementations that don’t use refcounting. In performance sensitive parts of the library, the `"".join()` form should be used instead

This will ensure that concatenation occurs in linear time across various implementations. Comparisons to singletons like None should always be done with `is` or `is not`, never the equality operators. Also, beware of writing `if x` when you really mean `if x is not None` – e.g., when testing whether a variable or argument that defaults to None was set to some other value. The other value might have a type (such as a container) that could be false in a Boolean context!

The first form means that the name of the resulting function object is specifically `f` instead of the generic `<lambda>`. This is more useful for tracebacks and string representations in general. The use of the assignment statement eliminates the sole benefit a lambda expression can offer over an explicit `def` statement (i.e. that it can be embedded inside a larger expression) Derive exceptions from `Exception` rather than `Base Exception`. Direct inheritance from `Base Exception` is reserved for exceptions where catching them is almost always the wrong thing to do.

Design exception hierarchies based on the distinctions that code *catching* the exceptions is likely to need, rather than the locations where the exceptions are raised. Aim to answer the question “What went wrong?” programmatically, rather than only stating that “A problem occurred” Class naming conventions apply here, although you should add the suffix “Error” to your exception classes if the exception is an error. Non-error exceptions that are used for non-local flow control or other forms of signalling need no special suffix.

Use exception chaining appropriately. `raise X from Y` should be used to indicate explicit replacement without losing the original traceback. When deliberately replacing an inner exception (using `raise X from None`), ensure that relevant details are transferred to the new exception (such as preserving the attribute name when converting `Key Error` to `Attribute Error`, or embedding the text of the original exception in the new exception message). When catching exceptions, mention specific exceptions whenever possible instead of using a bare `except` clause. A bare `except:` clause will catch `System Exit` and `Keyboard Interrupt` exceptions, making it harder to interrupt a program with `Control-C`, and can disguise other problems. If you want to catch all exceptions that signal program errors, use `except Exception:` (bare `except` is equivalent to `except Base Exception:`).

A bare except: clause will catch System Exit and Keyboard Interrupt exceptions, making it harder to interrupt a program with Control-C, and can disguise other problems. If you want to catch all exceptions that signal program errors, use except Exception: (bare except is equivalent to except Base Exception:). If the exception handler will be printing out or logging the traceback; at least the user will be aware that an error has occurred. If the code needs to do some cleanup work, but then lets the exception propagate upwards with raise. try...finally can be a better way to handle this case.

When catching operating system errors, prefer the explicit exception hierarchy introduced in Python 3.3 over introspection of errno values. Be consistent in return statements. Either all return statements in a function should return an expression, or none of them should. If any return statement returns an expression, any return statements where no value is returned should explicitly state this as return None, and an explicit return statement should be present at the end of the function.

4.2.3 CHARACTERISTICS OF PYTHON

We briefly list some of C's characteristics that define the language and also have led to its popularity as a programming language. Naturally we will be studying many of these aspects throughout the course.

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming.

In Python, we don't need to declare the type of variable because it is a dynamically typed language.

➤ **Characteristics of Python:**

There are many features in Python, some of which are discussed below –

- Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc...
- One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

- Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.
- Python is an **Extensible** language. We can write us some Python code into C or C++ language and also, we can compile that code in C/C++ language.
- Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
- Python is also an integrated language because we can easily integrated python with other languages like c, c++, etc.
- Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called **bytecode**.
- Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.
- Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

4.3 MYSQL DATABASE

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

It is developed, marketed, and supported by **MySQL AB, a Swedish company**, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is ***My Ess Que Ell***. *However, you can pronounce it in your way.* Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
- It provides the Referential Integrity between rows or columns of various tables. It allows us to update the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users. It is very important to understand the database before learning MySQL. A database is an application that stores the organized collection of records. It can be accessed and managed by the user very easily. It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly.

4.4 WORKING OF MYSQL

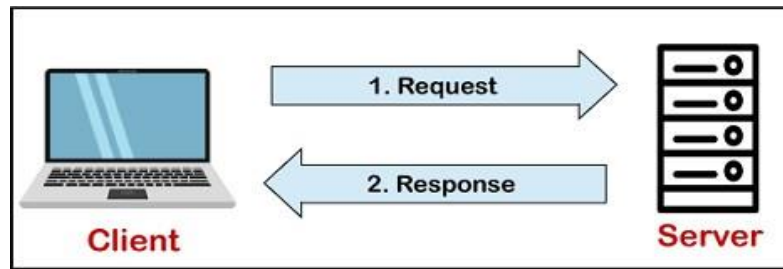


Fig 4.6 WORKING OF MYSQL

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.

The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

- MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
- Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
- Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

A client can use any MySQL GUI. But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier. Some of the most widely used MySQL GUIs are MySQL Workbench, SequelPro, DB Visualizer, and the Navicat DB Admin Tool. Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

4.5 CHARACTERISTICS OF MYSQL

MySQL is a system that helps store and manage data efficiently. Database generally stores data in a structured fashion. It is written in C and C++, and it has been tested with a variety of compilers to check for bugs and inconsistencies.

Let us understand some of the main features associated with MySQL:

- MySQL is open-source, which means this software can be downloaded, used and modified by anyone. It is free-to-use and easy-to-understand. The source code of MySQL can be studied, and changed based on the requirements.
- It uses GPL, i.e. GNU General Public license which defines rules and regulations regarding what can and can't be done using the application. MySQL stores data efficiently in the memory ensuring that data is consistent, and not redundant.
- Hence, data access and manipulation using MySQL is quick. Scalability refers to the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines, and so on. MySQL server was developed to work with large databases.
- It contains multiple data types such as unsigned integers, signed integers, float (FLOAT), double (DOUBLE), character (CHAR), variable character (VARCHAR), text, blob, date, time, datetime, timestamp, year, and so on.
- It supports different character sets, and this includes latin1 (cp1252 character encoding), German, Ujis, other Unicode character sets and so on.

- It provides a secure interface since it has a password system which is flexible, and ensures that it is verified based on the host before accessing the database. The password is encrypted while connecting to the server.
- It comes with support for large databases, which could contain about 40 to 50 million records, 150,000 to 200,000 tables and up to 5,000,000,000 rows.
- MySQL server also comes with many client and utility programs. This includes Command line programs such as 'mysqladmin' and graphical programs such as 'MySQL Workbench'.

4.6 ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include the acquisition of information and rules for using the information, using rules to reach approximate or definite conclusions, and self-correction. Artificial Intelligence has the capability to find patterns in big data to learn and reveal hidden information or deliver solutions to deliver complex problems. It is an automated process.

artificial intelligence (AI), the ability of a digital computer or computer- controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing chess—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge.

4.6.1 COMPONENTS OF ARTIFICIAL INTELLIGENCE

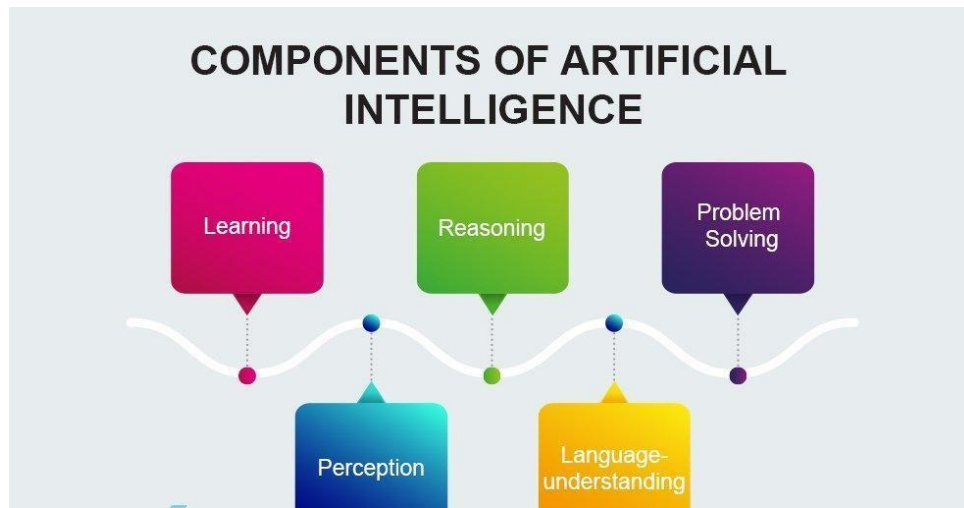


Fig 4.7 COMPONENTS OF ARTIFICIAL INTELLIGENCE

- **Learning**

Similar to humans, computer programs also learn in different manners. Talking of AI, learning by this platform is further segregated into a varied number of forms. One of the essential **components of ai**, learning for AI includes the trial-and- error method. The solution keeps on solving problems until it comes across the right results.

This way, the program keeps a note of all the moves that gave positive results and stores it in its database to use the next time the computer is given the same problem. The learning component of AI includes memorizing individual items like different solutions to problems, vocabulary, foreign languages, etc., also known as rote learning. This learning method is later implemented using the generalization method.

- **Reasoning**

The art of reasoning was something that was only limited to humans until five decades ago. The ability to differentiate makes Reasoning one of the essential **components of artificial intelligence**. To reason is to allow the platform to draw inferences that fit with the provided situation. Further, these inferences are also categorized as either inductive or deductive.

The difference is that in an inferential case, the solution of a problem provides guarantees of conclusion. In contrast, in the inductive case, the accident is always a result of instrument failure. The use of deductive interferences by programming computers has provided them with considerable success. However, reasoning always involves drawing relevant inferences from the situation at hand.

- **Problem Solving**

In its general form, the AI's problem-solving ability comprises data, where the solution needs to find x. AI witnesses a considerable variety of problems being addressed in the platform. The different methods of 'Problem-solving' count for essential **artificial intelligence components** that divide the queries into special and general purposes.

In the situation of a special-purpose method, the solution to a given problem is tailor-made, often exploiting some of the specific features provided in the case where a suggested problem is embedded. On the other hand, a general-purpose method implies a wide variety of vivid issues. Further, the problem-solving component in AI allows the programs to include step-by-step reduction of difference, given between any goal state and current state. The reflex agent of AI directly maps states into action. Whenever these agents fail to operate in an environment where the state of mapping is too large and not easily performed by the agent, then the stated problem dissolves and sent to a problem-solving domain which breaks the large stored problem into the smaller storage area and resolves one by one. The final integrated action will be the desired outcomes. On the basis of the problem and their working domain, different types of problem-solving agent defined and use at an atomic level without any internal state visible with a problem-solving algorithm.

- **Perception**

In using the 'perception' component of Artificial Intelligence, the element scans any given environment by using different sense-organs, either artificial or real. Further, the processes are maintained internally and allow the perceiver to analyze other scenes in suggested objects and understand their relationship and features. This analysis is often complicated as one, and similar items might pose considerable amounts of different appearances over different occasions, depending on the view of the suggested angle.

At its current state, perception is one of those **components of artificial intelligence** that can propel self-driving cars at moderate speeds. FREDDY was one of the robots at its earliest stage to use perception to recognize different objects and assemble different artifacts.

- **Language Understanding**

In simpler terms, language can be defined as a set of different system signs that justify their means using convention. Occurring as one of the widely used **artificial intelligence components**, language understanding uses distinctive types of language over different forms of natural meaning, exemplified overstatements.

One of the essential characteristics of languages is humans' English, allowing us to differentiate between different objects. Similarly, AI is developed in a manner that it can easily understand the most commonly used human language, English. This way, the platform allows the computers to understand the different computer programs executed over them easily.

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English . Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc. After register in user login through user email id and password after that upload the defected leaf image

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 SCREENSHOTS

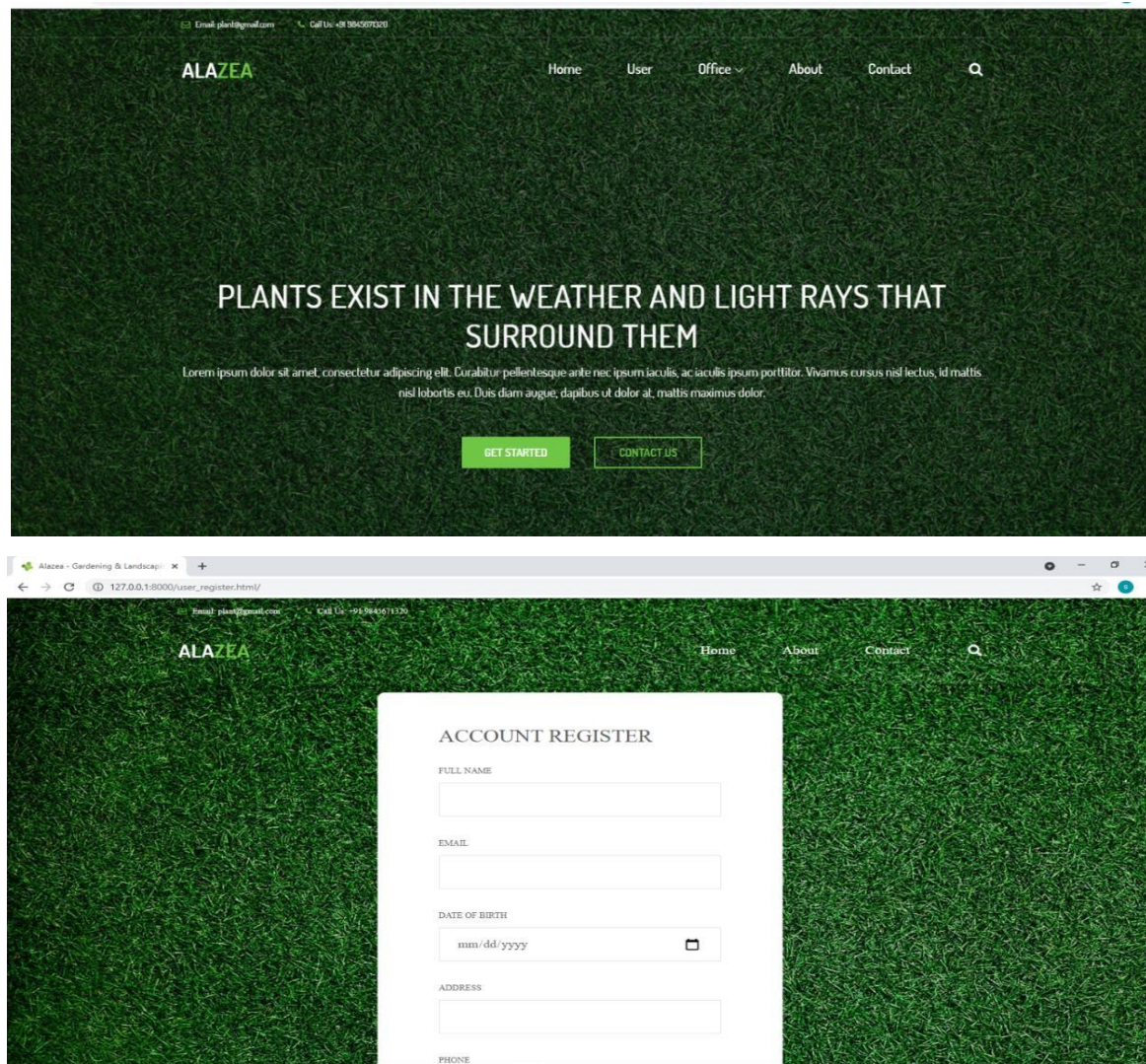
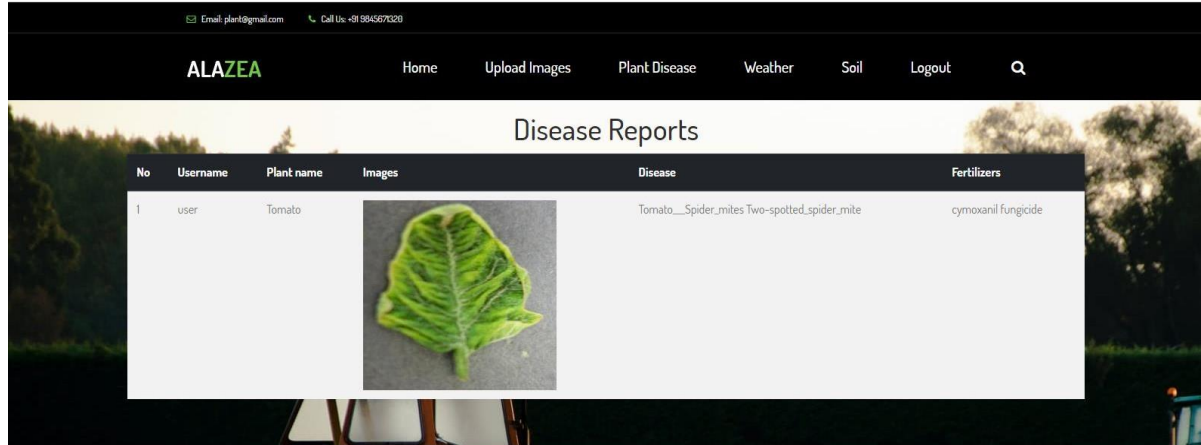


Fig:5.1 Registration for Uploading Image

In this project first we have to connect MySQL database with PyCharm through WampServer, then when we run it in PyCharm TensorFlow server will run then from PyCharm it will automatically redirect to this ALZEA application, where user, client and admin should register with their details to find the defected leaf and cure for it.



ABOUT US

We are leading in the plants service fields.

Quality Products

Inbiam eu sagittis est. at commodo lacini

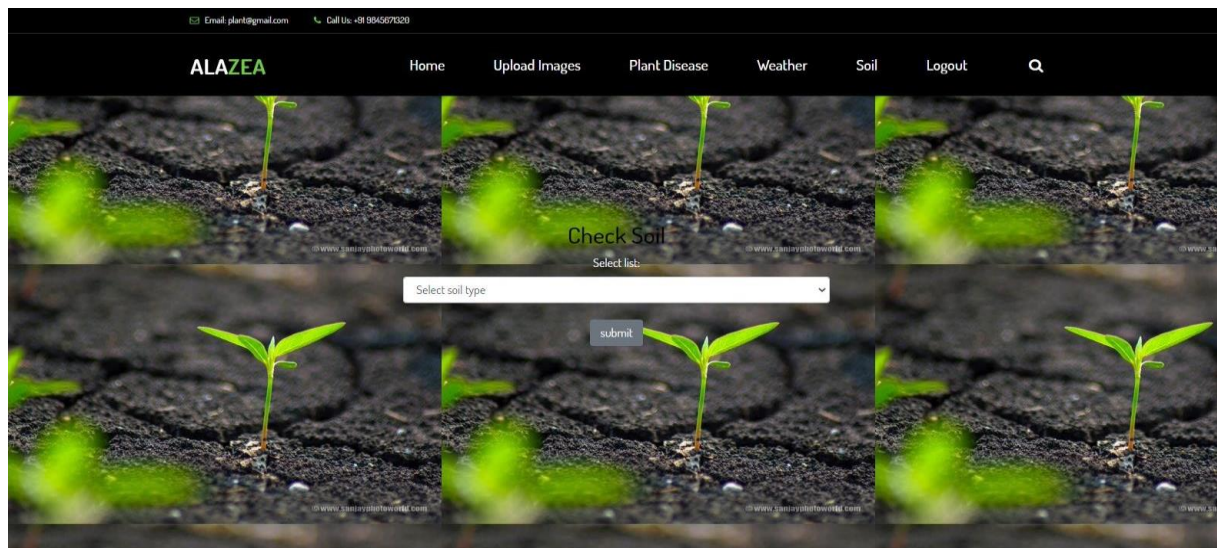


Fig:5.2 Uploading Image and Soil report check

After register in user login through registered user email id and password after that upload the defected leaf image for predict the disease and then we can check weather report and soil recommendation.

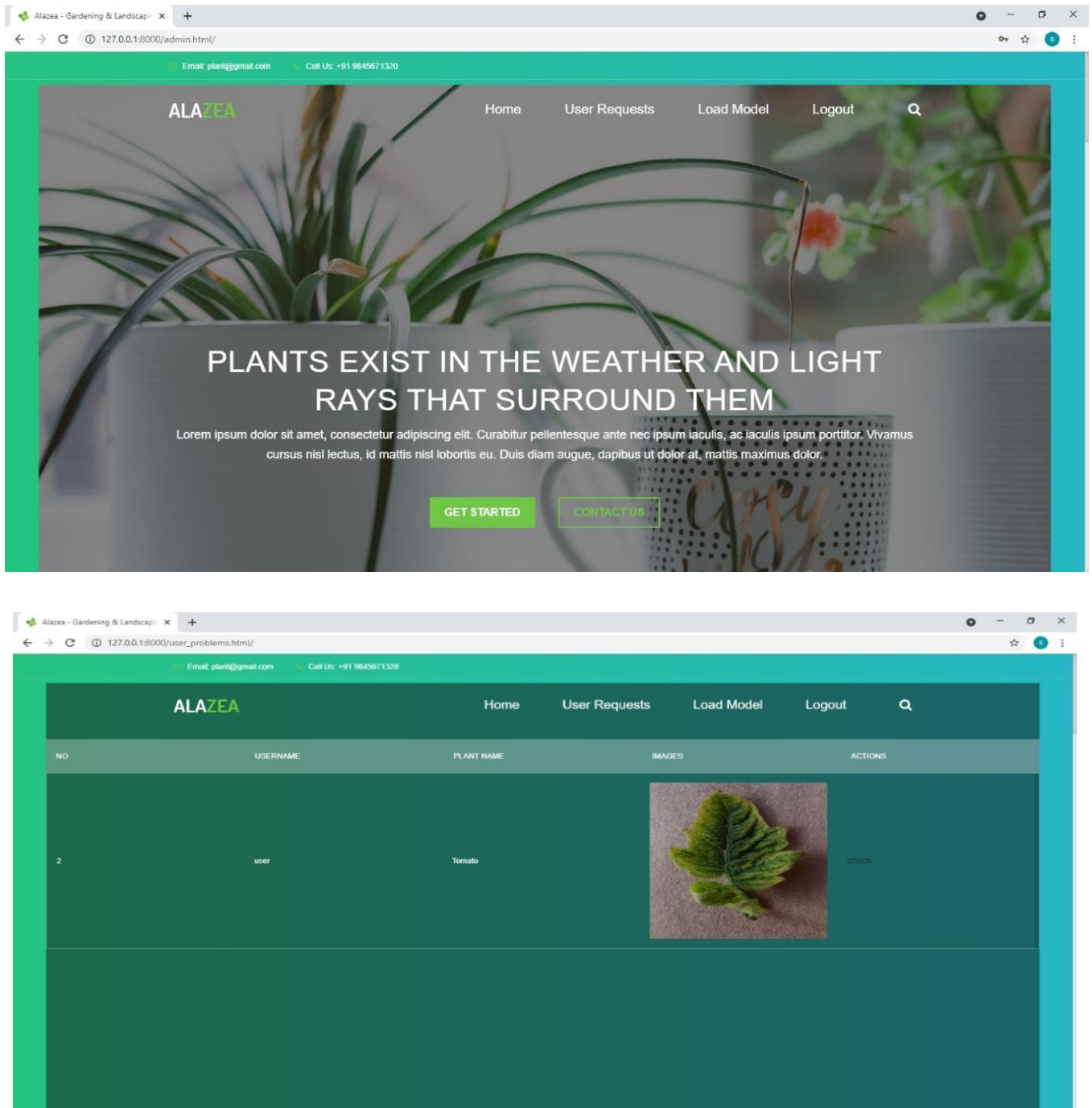


Fig 5.3 User Request

When we finished the uploading the image then logout from the user section and then register in client with new details it should be different from user the login with the registered client email id and password, then click on user request where leaf disease name will show. In this output after getting the result in client section logout from client

5.2 OUTPUT SCREEN

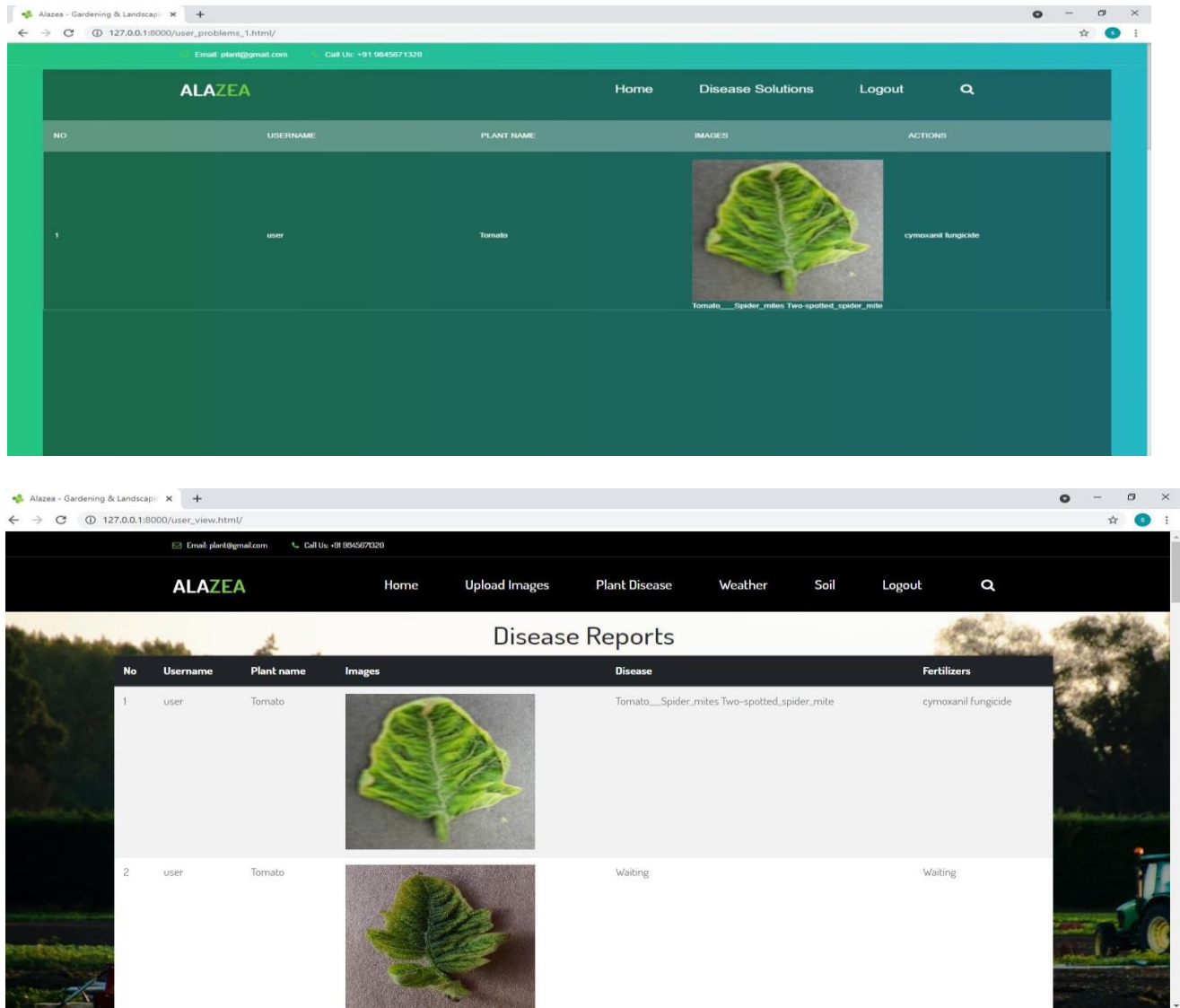


Fig 5.4 Output Screen

In this output after getting the result in client section logout from client logout from that page then click on admin and sign-up again with the registered admin user name and password to check disease name of the leaf and what action need to for cure of that predicted disease leaf.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this paper, we present Image recognition using a convolution neural network where the user's uploaded leaves are matched with trained data using CNN, where pre-processing is much lesser compared to other processes like Distance metric learning and matrix factorization. Even though CNN has high accuracy in image recognition problems, CNN does not encode the position and orientation of the object. Lack of ability to be spatially invariant to the input data and lots of training data is required. We leave these constraints for future enhancement where the application should encode the position and or orientation of the object and work efficiently with minimal trained data.

6.2 FUTURE WORK

Our application plays a vital role in recognizing the crop's leaves image and finding out the disease cause and fertilize suggestions after comparing with the trained model using Convolution Neural Network helps to differentiate each image by converting them by process of matrix convolution. CNN represents an interesting method for adaptive image processing and forms a link between general feed-forward neural networks and adaptive filters. Any user who grows different crops may not know all the diseases caused by fungus and bacteria.

By uploading, their affected crops leave users to benefit by understanding the disease name and cause then our client can suggest fertilizer by analysing the composition between modeled images and user images. Our application creates a good environment for fertilizer suggestions and two more important processes where the user can have weather and soil reports. Based on the location given by the user, a weather report is generated and a soil report contains the suggested crops to grow by the user based on the soil samples given by the user. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

REFERENCES

- [1] Ahmed, K, Shahidi, T.R., Irfanul Alam, S.M., Momen, S., 2019. Rice leaf disease detection using Machine Learning techniques, 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI). IEEE, Dhaka, Bangladesh.
- [2] Taskeen Ashraf, Yasir Niaz Khan., 2020. Weed density classification in rice crop using computer vision. *Compute, Electron, Agric.*175.
- [3] Sanga, S.L., Machuve, D., Jomanga, K., 2020. Mobile-based deep learning models for Banana Disease detection. *Technol. Appl. Sci. Res.* 10(3).
- [4] Tewari, V.K., Pareek, C.M., Lal, G., Dhruv, L.K., Singh, N., 2020. Image processing based real-time variable-rate chemical spraying system for diseases control in paddy. *Artific. Intellig. Agric.*
- [5] Ramesh, S., Vydeki, D., 2020. Recognition and classification of paddy leaf diseases using optimized deep neural network with Jaya algorithm. *Inform. Proc. Agric.*
- [6] Sethy, P.K, Barpanda, N.K, Rath, A.K, Behera, S.K. Deep feature-based rice leaf diseases identification using support vector machine. *Comput. Electron. Agric.* 2020.
- [7] Li, Y.; Yang, J. Few-shot cotton pest recognition and terminal realization. *Comput. Electron. Agric.* 2020, 169, 105240.
- [8] Lee, S.H.; Goeau, H.; Bonnet, P.; Joly, A. New perspectives on plant disease characterization based on deep learning. *Comput Electron. Agric.* 2020, 170, 105220.
- [9] Rangarajan Arvind, K.; Rajan, P. Automated disease classification in (Selected) agriculture crops using transfer learning. *Autom, Caspois Autom. Mjer. Elektron .Ra cunarstvo Komun.* 2020.
- [10] Arora, J.; Agrawal, U. Classification of Maize leaf disease from healthy leaves using Deep Forest. *J.Artif. Intell, Syst.* 2020

- [11] Li, Y.; Nie, J.; Chao, X. Do we really need Deep CNN for plant diseases identification? Comput, Electron, Agric. 2020.
- [12] Oyewola, D.O.; Dada, E.G.; Misra, S.; Damasevicius, R. Detecting cassava mosaic disease using a deep residual convolutional neural network with block processing. PreeJ. Comput, Sci. 2021.
- [13] Zeng, W.; Li, M. Crop leaf disease recognition based on Self-Attention convolutional neural network. Comput, Electron. Agric. 2020.
- [14] Zhao, Y.; Chen Z.; Gao, X.; Song, W.; Xiong, Q.; Zhang, Z. Plant Disease Detection using Generated Leaves Based on DoubleGAN. IEEE/ACM Trans.Comput. Biol. Bioinform.2021.
- [15] J. Abdulridha, Y. Ampatzidis, S. C. Kakarla, and P.Roberts, "Detection of target spot and bacterial spot diseases in tomato using UAV-based and benchtop-based hyperspectral imaging techniques , " *Precis. Agricult.*, vol. 21, no. 5, pp .oct. 2020.
- [16] Agarwal, M. et al. (2020) 'Toled,;." Tomato Leaf Disease Detection using Convo- lutional Neural Network", *Procedia Computer Science*. 2020
- [17] AL-Sadi, A.(2017) 'Impact of Plant Disease on Human Health',*International Journal of Nutrition, Pharmacology, Neurological Diseases*. 2021.
- [18] Ault, R. (2020) 'Optimization Study of an Image Classification Deep Neural Network'.
- [19] Abbas, A.; Jain, S.; Gour, M.; Vankudothu, S. Tomato plant disease detection using transfer learning with C-GAN synthetic images, *Comput. Electron. Agric*. 2021.
- [20] Maeda-Gutierrez, V.; Galvan-Tejada, C.E.; Zanella-Calzada, L.A.; Celaya-Padilla, J.M.; Galcan-Tejada, J.L.; Gamboa-Rosales, H.; Luna-Garcia, H.; Magallanes-Quintanar, R.; Guerrero Mendez, C.A.; Olvera-Olvera, C.A. Comparison of Convolutional Neural Network Architecture for Classification of Tomato Plant Diseases. *April.Sci*.2020.

- [21] R. I. Hasan, S. M. Yusuf, and L. Alzubaidi, "Review of the state of the art of deep learning for plant diseases: a board analysis and discussion," *Plants*, 2020.
- [22] V. V. Adit, C. V. Rubesh, S. S. Bharathi, G. Santhiya, and R. Anuradha, "A Comparison of Deep Learning Algorithm for Plant Disease Classification," Springer, Singapore, 2020.
- [23] M.Loey, A. ElSawy, and M. Afify, "Deep learning in plant diseases detection for agricultural crops," *International Journal of Service Science*, 2020.
- [24] S. Hernandez and J. L. Lopez, "Uncertainty quantification for plant disease detection using Bayesian deep learning," *Applied Soft Computing*. 2020.
- [25] M. H. Saleem, S. Khanchi, J. Potgieter, and K. M. Arif, "Image-based plant disease identification by deep learning meta-architecture,". 2020.
- [26] U. Atila, M. Ucar, K.Akyol, and E. Ucar, "Plant leaf disease classification using EfficientNet deep learning model," *Ecological Informatics*, 2021.
- [27] J. G. M. Esgario, R. A. Krohling, and J. A. Ventura, "Deep learning for classification and severity estimation of coffee leaf biotic stress," *Computers and Electronics in Agriculture*, 2020.
- [28] R. Chalapathy, and S. Chawla, "Deep learning for anomaly detection a survey," 2020.
- [29] Y. Guo, J. Zhang, C. Yin., "Plant disease identification based on deep learning algorithm in smart farming," *Discrete Dynamics in Nature and Society*, 2020.
- [30] J. S. H. Al-bavati and B. B. Ustundag, "Evolutionary feature optimization for plant leaf disease detection by deep neural networks," *International Journal of Computational Intelligence Systems*, 2020.

APPENDIX

A. ARDUINO CODE:

- **Setting.py**

```
"""
Django settings for plant project.
Generated by 'django-admin startproject' using Django 3.2.5.
For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

import os
from pathlib import Path
from .email import *
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(_file_).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-m%s7ly(tv)o724txy_p85o@+ywl#q%-umi9x+0%=w4@1(9^jk5'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'projectadmin',
    'client',
    'user',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
```

```

'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'plant.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [(os.path.join(BASE_DIR, 'assets/templates'))],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.media',
            ],
        },
    },
]

WSGI_APPLICATION = 'plant.wsgi.application'
# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'plant',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '',
        'port': ''

    }
}
# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {

```

```

        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/
STATIC_URL = '/static/'
STATICFILES_DIRS=[os.path.join(BASE_DIR,'assets/static')]
MEDIA_URL='/media/'
MEDIA_ROOT=os.path.join(BASE_DIR,'assets/media')
# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
EMAIL_USE_TLS = SET_EMAIL_USE_TLS
EMAIL_HOST = SET_EMAIL_HOST
EMAIL_HOST_USER = SET_EMAIL_HOST_USER
EMAIL_HOST_PASSWORD = SET_EMAIL_HOST_PASSWORD
EMAIL_PORT = SET_EMAIL_PORT
EMAIL_BACKEND = SET_EMAIL_BACKEND
DEFAULT_FROM_EMAIL = SET_DEFAULT_FROM_EMAIL

```

- **Views.py**

```
from django.shortcuts import render, redirect, Http404,
HttpResponse
from django.contrib import messages
import os
from django.conf import settings
from .models import *
from django.core.files.storage import FileSystemStorage
from .whether import weather as wea

# Create your views here.
def register(request):
    if request.method == "POST":
        use = False
        username = request.POST.get('username')
        try:
            User.objects.get(username=username)
            use = True

        except User.DoesNotExist:
            pass
        if use:
            messages.error(request, "username already exists")
            return redirect('/user_registration.html')
        else:
            user = User()
            user.username = username
            user.name = request.POST.get('name')
            user.date = request.POST.get('dob')
            user.email = request.POST.get('email')
            user.password = request.POST.get('pass')
            user.phone = request.POST.get('phone')
            user.address = request.POST.get('address')
            user.save()
            messages.success(request, "registered successfully")
            return redirect('/user_login.html')
        return render(request, "register.html")
def login(request):
    if "user" in request.session:
        messages.success(request, 'already logged in')
        return redirect('/user_page.html')
    else:
        if request.method == 'POST':
            username = request.POST.get('username')
            password = request.POST.get('pass')
            print(0)
            try:
```

```

        user = User.objects.get(username=username)
        print(1)
        if user.password != password:
            print(2)
            messages.success(request, 'password is wrong')
            return redirect('/user_login.html')
        request.session['user'] = user.username
        return redirect('/user.html')
    except User.DoesNotExist:
        messages.success(request, 'username does not exist')
        return redirect('/user_login.html')
    return render(request, "login.html", {'name': 'user'})

def page(request):
    if 'user' in request.session:
        return render(request, 'user.html')
    else:
        messages.success(request, 'session already loggedout')
        return redirect('/user_login.html')

def logout(request):
    if 'user' in request.session:
        request.session.pop('user', None)
        messages.success(request, 'logout already successfully')
        return redirect('/')
    else:
        messages.success(request, 'session loggedout')
        return redirect('/user_login.html')

def upload(request):
    if 'user' in request.session:
        if request.method == "POST":
            dis = Disease()
            dis.user = request.session['user']
            img = request.FILES['image']
            dis.plant_name = request.POST.get('username')
            dis.image = img
            dis.save()
            return render(request, 'asdf.html')
        else:
            messages.success(request, 'session loggedout')
            return redirect('/user_login.html')

```

```

def view(request):
    if 'user' in request.session:
        dis = Disease.objects.filter(user=request.session['user'])
        print(request.session['user'])
        return render(request, 'user_view.html', {'dis': dis})
    else:
        messages.success(request, 'session loggedout')
        return redirect('/user_login.html')

def weather(request):
    if 'user' in request.session:
        if request.method == "POST":
            city = request.POST.get('city')
            i, j, k, l = wea(city)

            return render(request, 'weather.html', {'i': i, 'j': j, "k": k, 'l': l})
            return render(request, 'weather.html')
        else:
            messages.success(request, 'session loggedout')
            return redirect('/user_login.html')
    soil_rep={'Alluvial':['Tobacco', 'Cotton', 'Rice', 'Wheat', 'Bajra',
'Sorghum', 'Pea', 'Pigeon Pea', 'Chicpea', 'Black Gram', 'Green
Gram', 'soybean', 'Groundnut', 'Mustard', 'Linseed', 'Sesame',
'Barley', 'Jute', 'Maize', 'any oilseeds', 'vegetables', 'fruits'],
            'Black':['Rice', 'Sugarcane', 'Wheat', 'jowar', 'linseed',
'sunflower', 'ceral crops', 'citrus fruits', 'vegetables', 'tobacco',
'groundnut', 'millets'],
            'red and yellow':['Rice', 'Wheat', 'sugarcane', 'maize/corn',
'groundnut', 'ragi', 'potato', 'oilseeds', 'pulses', 'millets',
'mango', 'orange', 'citrus'],
            'Laterite':['Cotton', 'Rice', 'wheat', 'pulses', 'Tea', 'Growing
coffee', 'rubber', 'coconut', 'cashews'],
            'Arid':['Wheat', 'cotton', 'corn', 'millets', 'pulses', 'barley'], 'forest
and mountain':['Tea', 'Spices', 'Wheat',
'maize', 'barley', 'coffee', 'tropical fruits'],
            }
def soil(request):
    if 'user' in request.session:
        if request.method == "POST":
            soil = request.POST.get('soil')
            i= soil_rep[soil]
            return render(request, 'soil_1.html', {'i': i, 'soil': soil})
            return render(request, 'soil.html')
        else:
            messages.success(request, 'session loggedout')
            return redirect('/user_login.html')

```

plagiarism report

ORIGINALITY REPORT

3%

0%

2%

0%

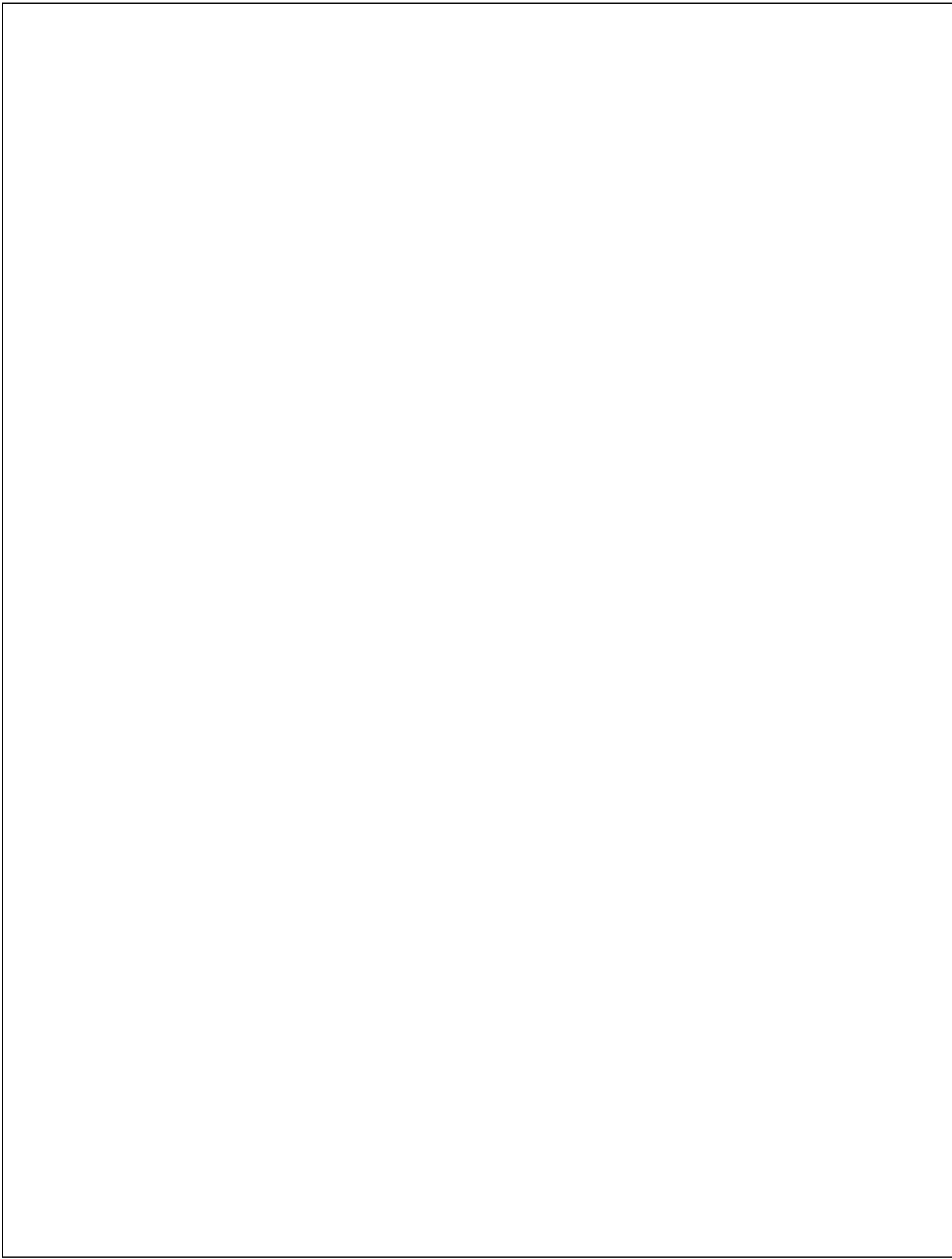
SIMILARITY INDEX

**INTERNET
SOURCES**

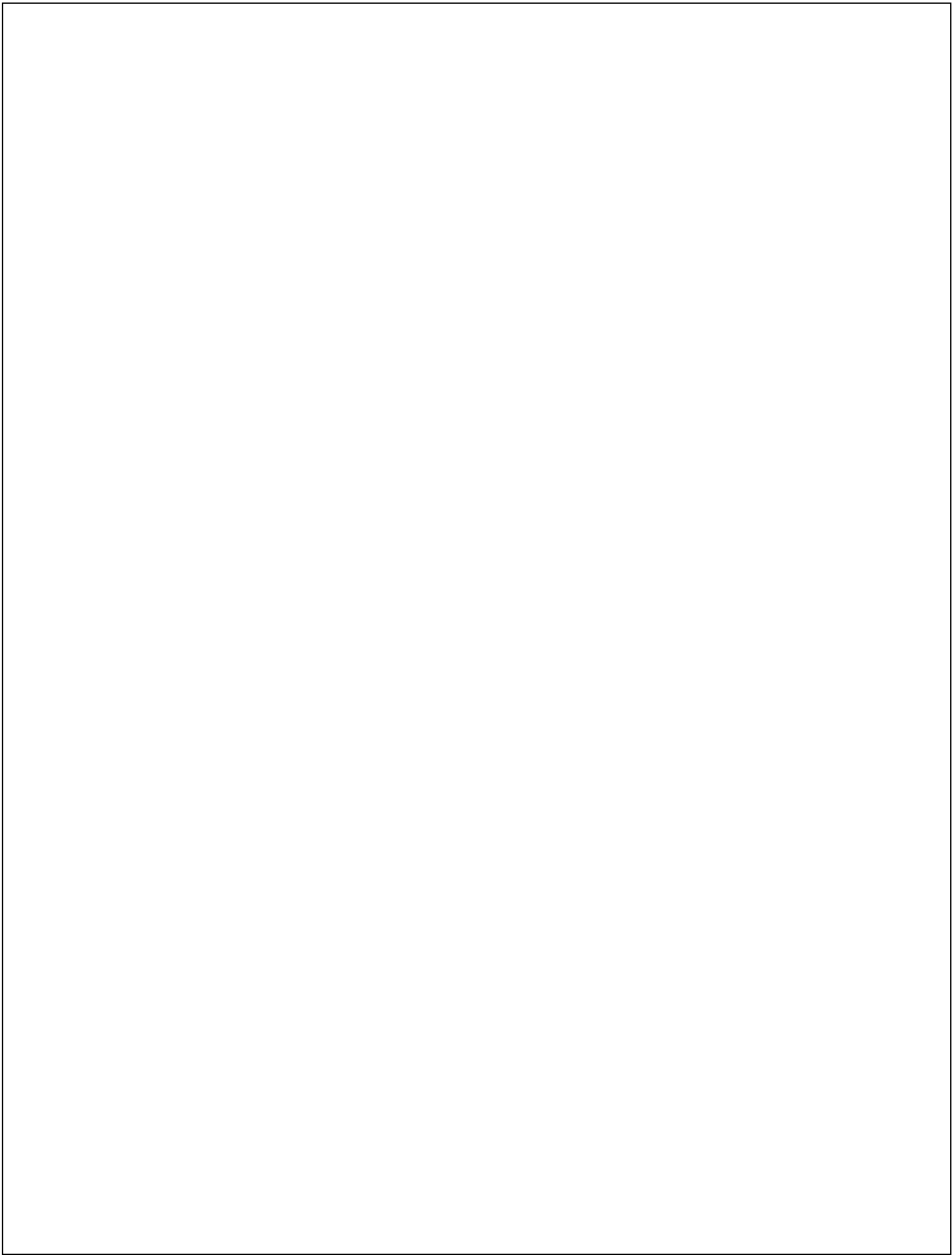
PUBLICATION

**STUDENT
PAPERS**

3% Plagiarism



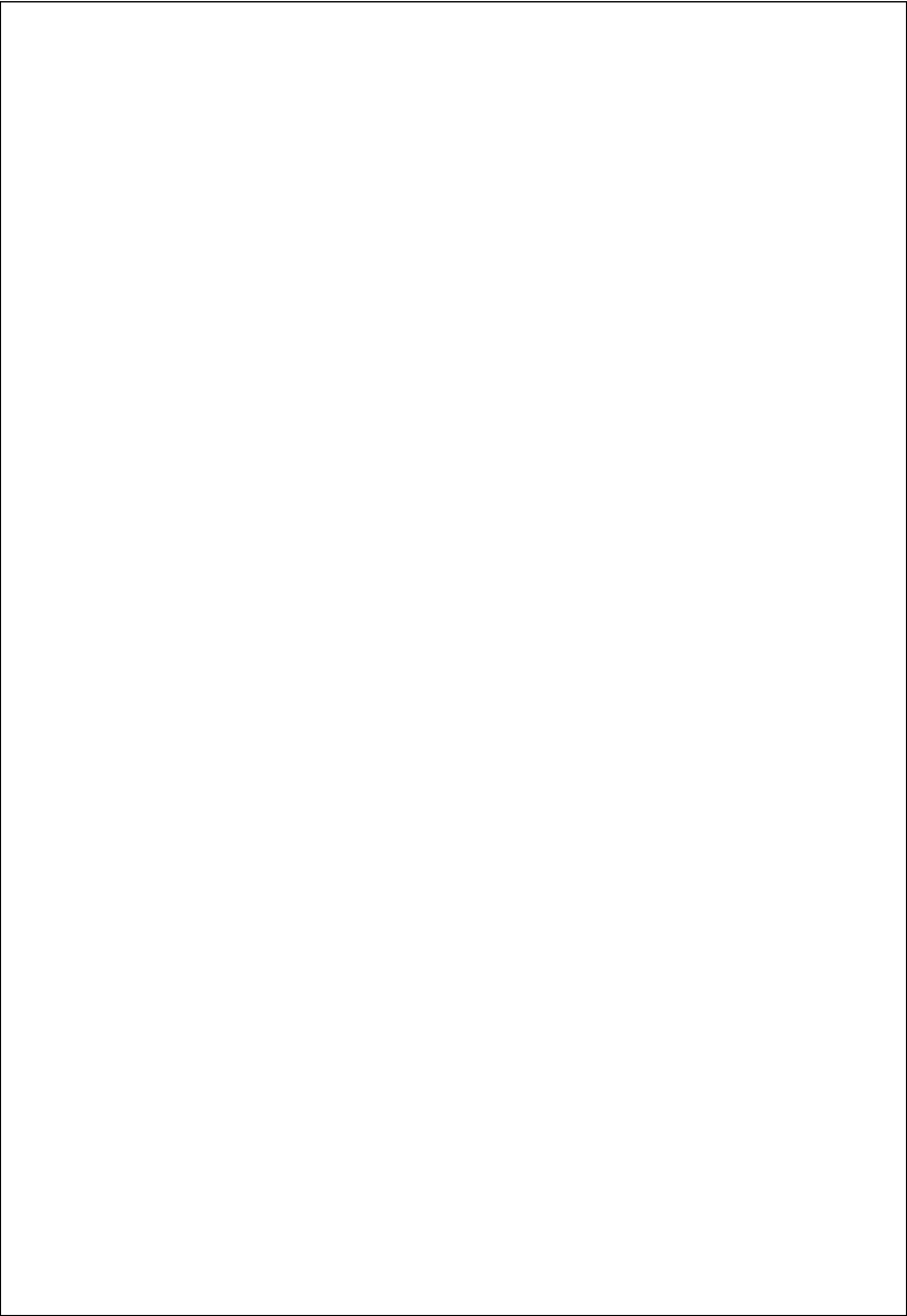


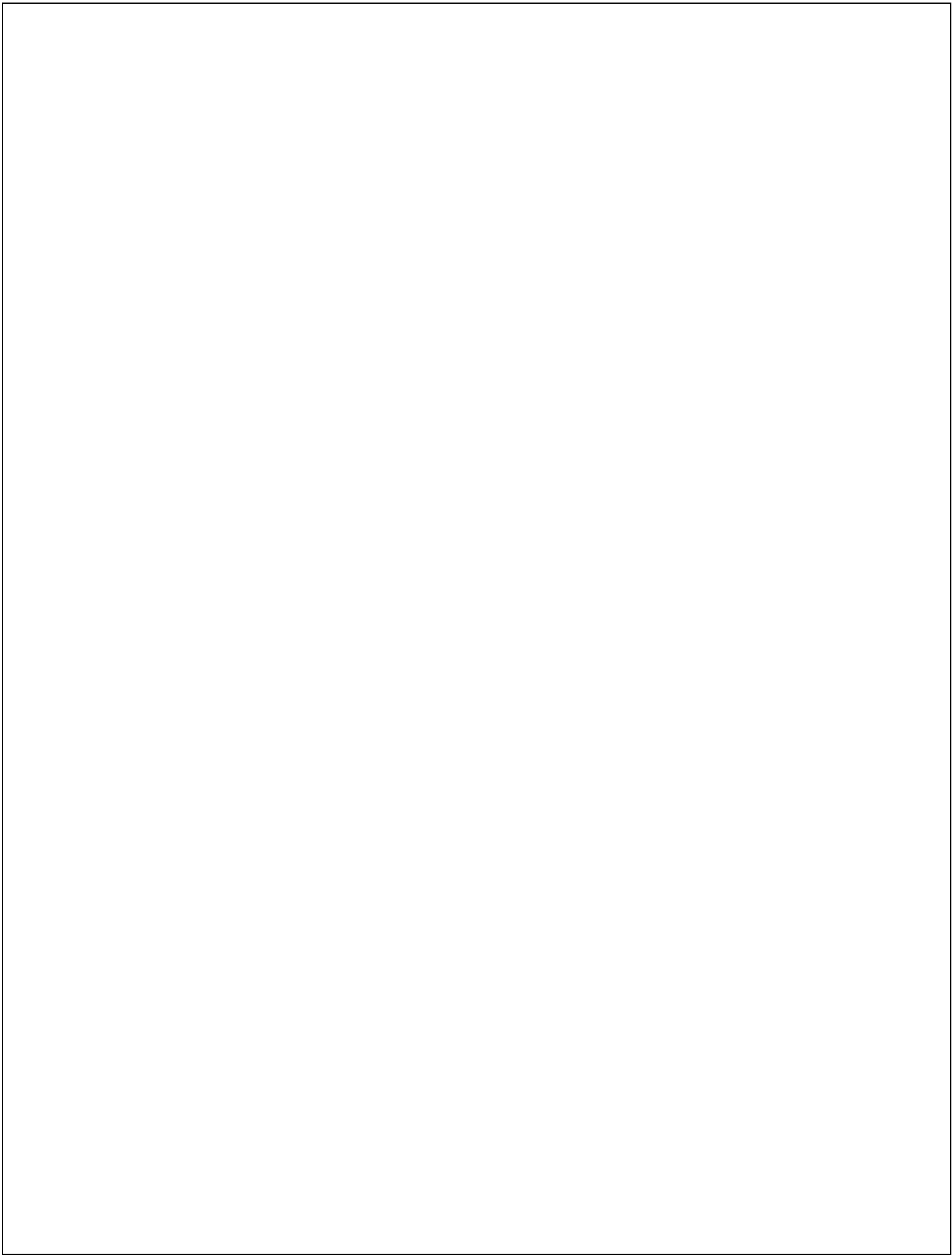


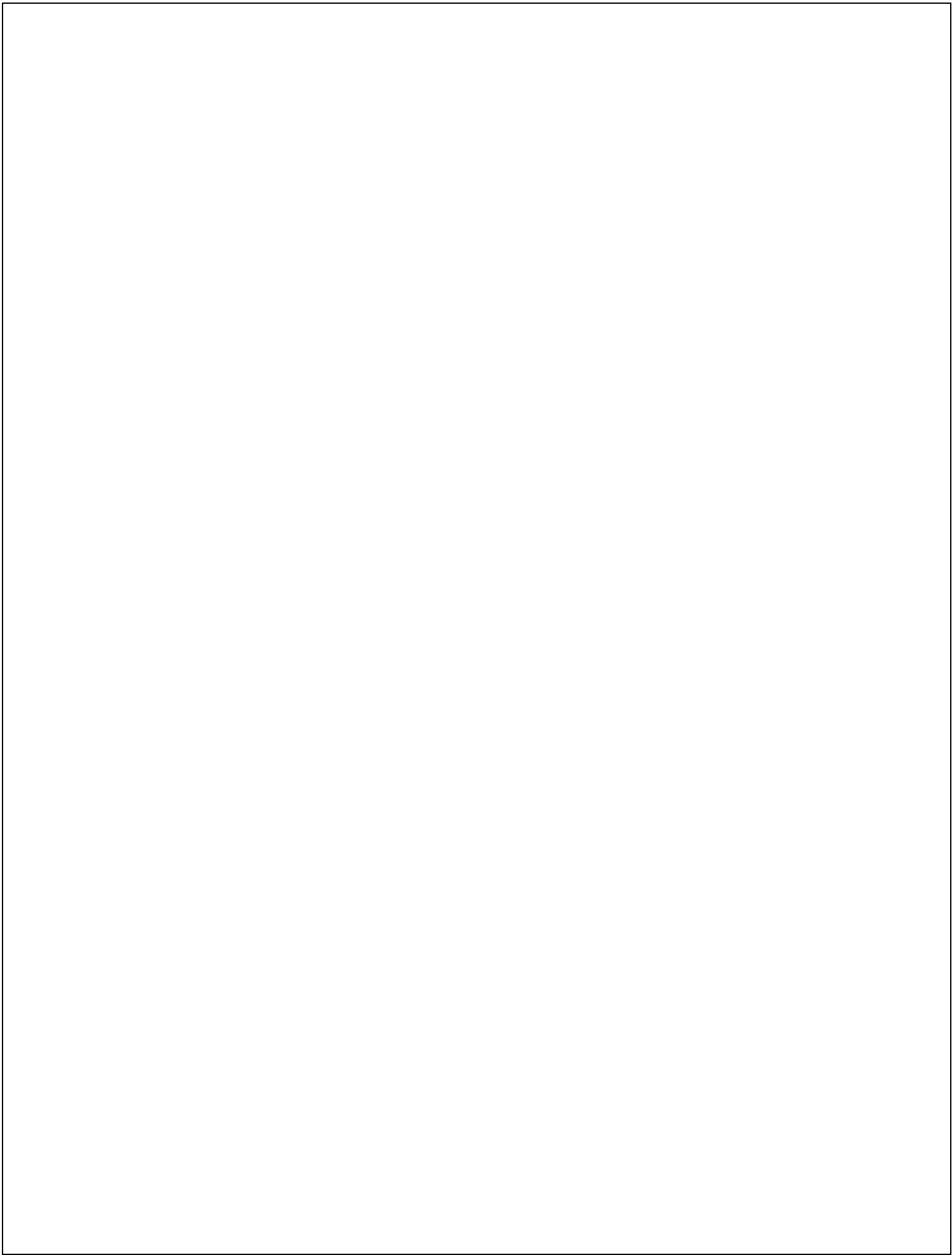


.









.

